# Sparkflows Documentation

***Release 0.0.1***

**Sparkflows**

**Jan 27, 2021**

# Contents


**1 Architecture & Deployment** **3**
1.1 Architecture & Deployment . . . . . 3

**2 Installation** **7**
2.1 Installation . . . . . 7

**3 Configuration** **35**
3.1 Configuration . . . . . 35

**4 Authentication** **75**
4.1 Authentication . . . . . 75

**5 Security** **89**
5.1 Security . . . . . 89

**6 Operating Fire Insights** **97**
6.1 Operating Guide . . . . . 97

**7 Quick Start Guide** **107**
7.1 Quickstart Guide . . . . . 107

**8 User Guide** **117**
8.1 User Guide . . . . . 117

**9 Web App User Guide** **147**
9.1 Analytical Apps User Guide . . . . . 147

**10 Data Science** **159**
10.1 Machine Learning User Guide . . . . . 159

**11 Time Series** **183**
11.1 Time Series Analysis . . . . . 183

**12 Tutorials** **193**
12.1 Tutorials . . . . . 193

**13 Troubleshooting** **393**
13.1 Troubleshooting . . . . . 393

**14 Frequently Asked Questions** **401**
14.1 FAQ . . . . . 401

**15 Administration** **409**
15.1 Administration Guide . . . . . 409

**16 Databricks Integration** **411**
16.1 Databricks Guide . . . . . 411

**17 AWS Integration** **435**
17.1 AWS Guide . . . . . 435

**18 AZURE Integration** **469**
18.1 AZURE Guide . . . . . 469

**19 Load Balancer Integration** **479**
19.1 Load Balancer . . . . . 479

**20 Superset** **485**
20.1 Superset . . . . . 485

**21 Python** **491**
21.1 Python Integration . . . . . 491

**22 Performance** **501**
22.1 Performance Tuning . . . . . 501

**23 Developer Guide** **503**
23.1 Developer Guide . . . . . 503

**24 Processors** **515**
24.1 Processors . . . . . 515

**25 Release Notes** **707**
25.1 Release Notes . . . . . 707

**26 REST API Authentication** **719**
26.1 REST API Authentication . . . . . 719

**27 REST API's using Python** **727**
27.1 REST API Examples using Python . . . . . 727

**28 REST API's using Java** **733**
28.1 REST API Examples using Java . . . . . 733

**29 REST API's using curl** **735**
29.1 REST API Examples using curl . . . . . 735

**30 Third Party Acknowledgements** **773**
30.1 Third Party Acknowledgements . . . . . 773

**31 Indices and tables** **779**

Fire Insights makes it incredibly fast and easy to do Self-Serve Data Preparation and Advanced Analytics.

With the power of Fire Insights at your hands, seamlessly find value from your data and scale to Petabytes of data.

Install on the cloud, on-premise or even on your laptop. Fire Insights seamlessly integrates with the most complex of Enterprise Environments.

CHAPTER 1

# Architecture & Deployment

## 1.1 Architecture & Deployment

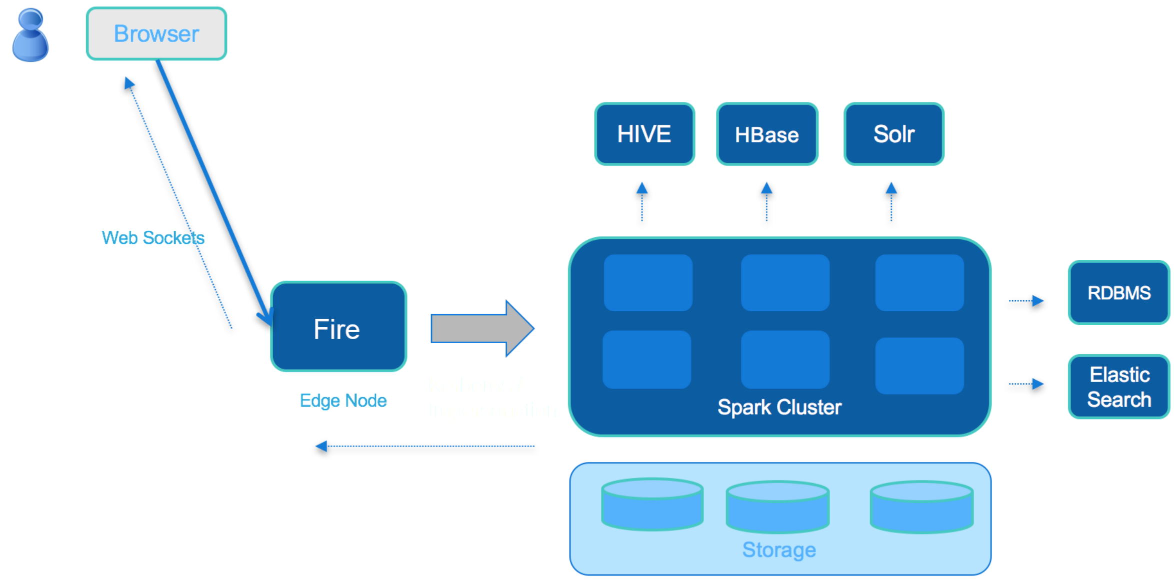
Sparkflows can be installed in one of two ways:

- On a standalone machine. In this case all the processing would happen within the single process.
  - This can be used to run Sparkflows on your laptop/desktop.
- On the Edge node of a Hadoop/Spark Cluster.
  - In this case, the jobs for processing would be submitted to the Hadoop/Spark Cluster.

### 1.1.1 Fire Architecture

Fire consists of three core components:

- **Web Browser** for defining end-to-end workflows for building data products and applications
  - Users interact with the web based drag and drop user interface for creating Datasets and Workflows
  - Workflows leverage the exhaustive set of functional and operational nodes such as Data Profiling, Data Cleaning, ETL, NLP, OCR, Machine Learning etc. displayed in the user interface.
- **Web Server** running on an Edge node in a Apache Spark Cluster
  - For running the workflows, they are submitted to the web server. The web server submits the workflow to the Apache Spark cluster as a spark job using spark-submit. The results of the workflow execution are streamed back and displayed in the Browser.
  - Web Server provides a host of other features likes interactive execution, schema inference and propagation, user permissions and roles, LDP integration etc.
- **Apache Spark** cluster on which the workflows are executed as Spark jobs
  - Workflows are saved in a JSON string.
  - Workflows can also be submitted on the spark cluster through **spark-submit** via a command line interface




### 1.1.2 Fire Deployment Options

Fire Insights can easily be deployed:

- On an Apache Hadoop/Apache Spark Cluster or
- On a standalone machine

#### Deployment on an Apache Hadoop/Apache Spark Cluster

The clusters could be based on the Apache Hadoop distribution from Cloudera, Hortonworks, MapR or any other Hadoop Cluster distributors.

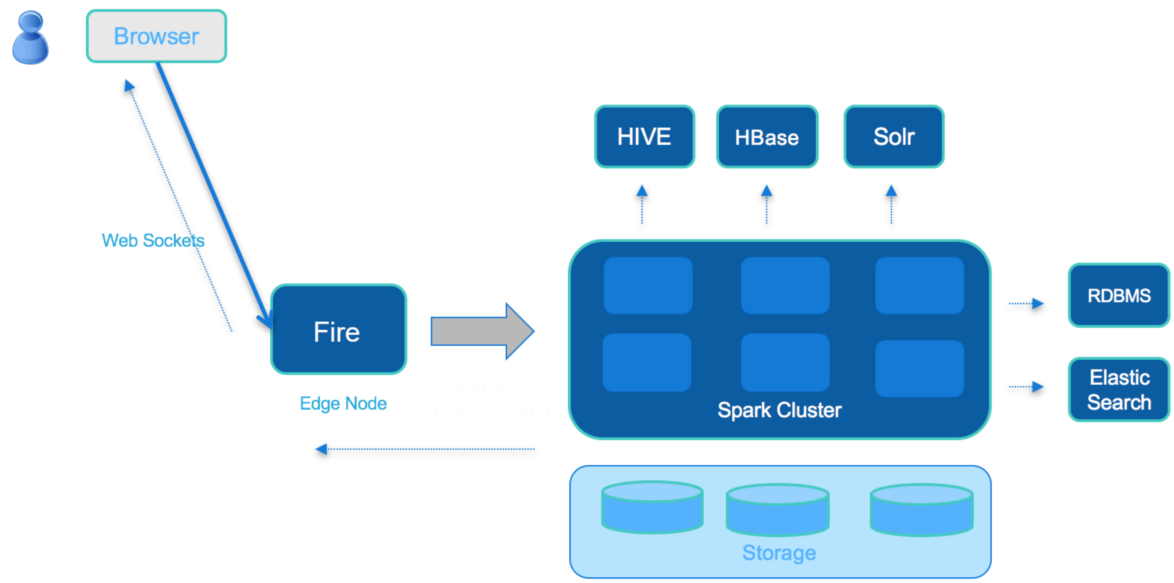
The cluster can be on-premise or on the cloud.

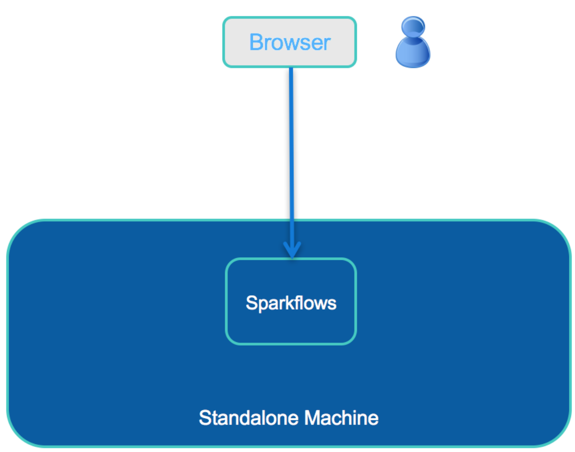
#### Deployment on a Standalone Machine

In this mode, Fire is installed on a mac/windows/linux machine. All the executions happen on that machine, in the web server.

This mode can be used for:

- Designing Workflows to be finally deployed on a larger Apache Spark Cluster
- For analyzing smaller sets of data


Browser
Web Sockets
Fire
Edge Node
HIVE
HBase
Solr
Spark Cluster
RDBMS
Elastic Search
Storage


Browser
Sparkflows
Standalone Machine

# CHAPTER 2

# Installation

## 2.1 Installation

### 2.1.1 Installer for laptop/desktop

You can download and use the Installer for installing/upgrading Fire Insights on your laptop or desktop.

This is not recommended to use on the server, where you need better control over the Installation process.

#### Prerequisites

- JDK 1.8

Java 8 can be downloaded and installed from here : https://www.oracle.com/java/technologies/javase-jdk8-downloads.html

You may have to set JAVA_HOME after the installation.

#### Download

Download the installer from : https://www.sparkflows.io/download

#### Execute

Execute the installer with :

java -jar sparkflows-installer-1.0.jar

Default port for sparkflows is : 8080

You can also change the port number while installing or starting the server.

**When you finish**

- Browse to http://<system-ip>:port
- Login with below credentials :
- Username : admin
- Password : admin

### 2.1.2 Linux/Mac OS Installation Prerequisites

Below are the Prerequisites for installing Fire Insights on a mac or linux machine:

```
- JDK 1.8+ installed.
- java and jar have to be in the PATH
- 8 GB+ of RAM.
- Python 3.6+ (when running Python and PySpark, otherwise not needed)
```

If Fire would be connected to an Apache Spark Cluster:

```
- Spark 2.X is needed on the cluster
- Fire has to be installed on an Edge node of the Spark Cluster
```

If using Python and PySpark (not needed for the core features of Fire Insights)

```
- Python 3.X can be set up with the Python virtual environment and activated
```

#### Downloading and Installing Java 8

Java 8 can be downloaded and installed from here : https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html

You may have to set JAVA_HOME after the installation.

There are various ways for Installing Java 8 on Linux. Some are listed below.

#### Using Linux RPM Package

- Download the Linux x64 RPM Package
- yum localinstall jdk-8u202-linux-x64.rpm (this has to be run as the root user)

Update `.bash_profile` to add the below:

```
export JAVA_HOME=/usr/java/jdk1.8.0_202-amd64/
export PATH=$PATH:$JAVA_HOME/bin
```

#### Download OpenJDK

- https://openjdk.java.net/install/
- Install OpenJDK on Ubuntu

https://docs.datastax.com/en/jdk-install/doc/jdk-install/installOpenJdkDeb.html

### 2.1.3 Linux/Mac OS Installation Steps

Fire can run independently on any machine, since we package Apache Spark along with or it can be connected to a Spark cluster.

If Sparkflows Fire needs to be connected to a Spark Cluster, install it on an edge node of the cluster. The edge node has the hadoop binaries and spark configs.

#### Quick Installation Steps of Fire with H2 DB

- Download the fire tgz file from:
  - https://www.sparkflows.io/download OR
  - https://www.sparkflows.io/archives
- Unpack it:

```
tar xvf fire-x.y.z.tgz
```

- Create H2 DB:

```
cd <fire install_dir>
./create-h2-db.sh
```

- Launch Fire Server:

```
cd <fire install_dir>
./run-fire-server.sh start
```

- Open your web browser and navigate to:

```
<machine_name>:8080
```

- Login with:

```
admin/admin or test/test
```

#### Detailed Installation Steps

- Glossary
  - `<install_dir>` : location where you unzipped fire tgz file. For example this can be your home directory.
  - `<machine_name>` : hostname where your installed Fire
  - `#` : used for comments and documentation
- Download the fire tgz file from:
  - https://www.sparkflows.io/download OR
  - https://www.sparkflows.io/archives
- Unzip it:

```
tar xvf fire-x.y.z.tgz
```

- Set up H2 or MySQL DB

Fire can be configured to run with H2 db or MySQL. H2 is very easy to set up with Fire. For production deployments MySQL is recommended.

- ../database/h2-db
- ../database/mysql-db
- Launch Fire:

```
cd <fire install_dir>
./run-fire.sh start
```

- Launch Fire Server:

```
cd <fire install_dir>
./run-fire-server.sh start
```

- Test by opening your web browser and going to:

```
localhost:8080

OR

<machine_name>:8080
```

- Login with:

```
username: admin and password: admin.
```

**Note:** Two user accounts come preconfigured with Fire.

- admin/admin
- test/test

You may change these usernames and passwords in Fire under the menu Administration/Users

### Stopping Fire

Stop Fire with the below:

```
./run-fire.sh stop
```

### Stopping the Fire Server

Stop the Fire Server with the below:

```
./run-fire-server.sh stop
```

### Connecting to Apache Spark Cluster

Now that you have Fire installed, you may want to connect it to your Apache Spark Cluster.

- *Connecting to Apache Spark Cluster*

## 2.1.4 Windows Installation Prerequisites

Below are the Prerequisites for installing Fire Insights on a windows machine:

```
- JDK 1.8 installed.
- java and jar have to be in the PATH
- 8+ GB of RAM on the machine.
- Python 3.6+ (when running Python and PySpark, otherwise not needed)
```

### Check JDK 1.8 is installed

- Check the JDK version installed on your machine:

```
Open the command window
Type the following command to check your java version : java -version
```

- If JDK 1.8 is not installed, follow the JDK installation steps mentioned below.

### Install JDK 1.8

- Download JDK 1.8 for windows using the link below:
  - https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Install java by double clicking on the downloaded exe file
- After installation, make sure that java 1.8 is in the path:

```
Open a new command window
Type the following command to check your java version : java -version
```

**Note:** If you have multiple versions of Java installed on you system, you can update the PATH using the steps outlined in either of the links below:

- https://javatutorial.net/set-java-home-windows-10
- https://www.java.com/en/download/help/path.xml

**Note:** With the above steps, you would have Fire Insights running locally on your laptop. It would not be able to submit jobs to a Spark cluster. For that, Fire Insights has to be installed on the edge node of the cluster.

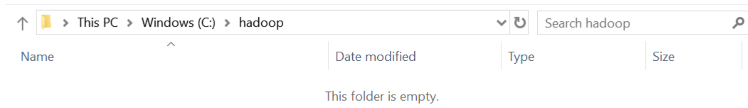
### winutils.exe

winutils.exe is needed for running Apache Spark/Hadoop on windows machines. Follow the below steps to setup winutils.exe on your machine:

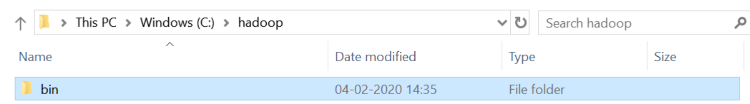
```
- Download winutils.exe from https://github.com/steveloughran/winutils
```

- winutils.exe can be directly downloaded from link below:

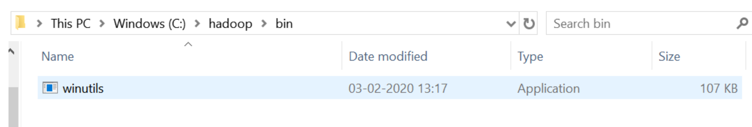
- https://github.com/steveloughran/winutils/blob/master/hadoop-2.7.1/bin/winutils.exe
- Create hadoop folder in Your System : `C:\hadoop`.




- Create bin folder in hadoop directory : `C:\hadoop\bin`.




- Copy the downloaded winutils.exe to the bin directory : `C:\hadoop\bin\winutils.exe`.




- Add a new Environment Variable.
  - `HADOOP_HOME = C:\hadoop`.
- Update the System Environment Variable PATH by adding : `%HADOOP_HOME%\bin`.
- Guide to setting Environment Variables on Windows

  https://www.architectryan.com/2018/08/31/how-to-change-environment-variables-on-windows-10/

### Troubleshooting

#### Running into an exception when saving files

> org.apache.spark.SparkException: Job aborted due to stage failure: Task 1 in stage 33.0 failed 1 times, most recent failure: Lost task 1.0 in stage 33.0 (TID 131, localhost): java.io.IOException: (null) entry in command string: null chmod 0644

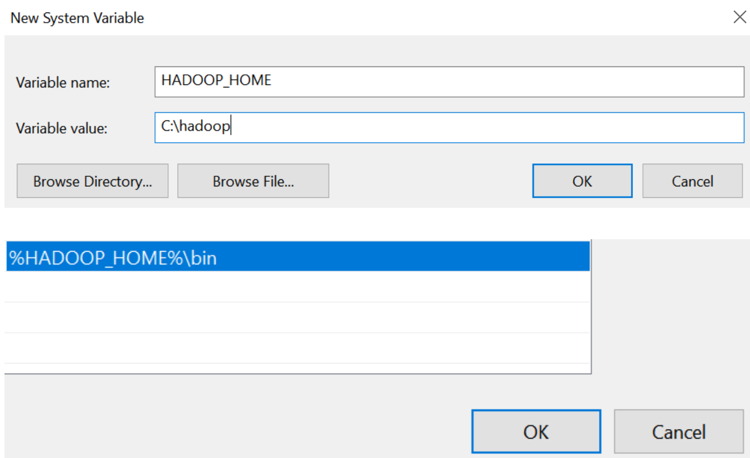
If you run into an exception like above, then there is problem with the setup of `winutils.exe`.

## 2.1.5 Windows Installation Steps

Fire Insights can be installed to run independendly on Windows.

### Installation Steps of Fire Insights with H2 DB

- Download the fire tgz file from:
  - https://www.sparkflows.io/download OR




  - https://www.sparkflows.io/archives

- Unpack the downloaded tgz file. Below are some tools which can be used for it:

```
WinRar : https://www.rarlab.com/download.htm
WinZip : https://www.winzip.com
7-Zip : https://www.7-zip.org/download.html
```

- Create H2 DB:

```
cd <fire install_dir>
.\create-h2-db.bat
```

- Launch Fire Server:

```
cd <fire install_dir>
.\run-fire-server.bat start
```

- Open your web browser and navigate to:

```
<machine_name>:8080
```

- Login with:

```
admin/admin or test/test
```

**Note:** Two user accounts come preconfigured with Fire Insights.

- admin/admin
- test/test

You may change these usernames and passwords in Fire under the menu Administration/Users

### Stopping the Fire Server

Stop the Fire Server with the below:

```
.\run-fire-server.bat stop
```

#### Stopping Fire Helper Processes

Stop Fire helper processes with the below:

```
.\run-fire.bat stop
```

### 2.1.6 Python Installation on Linux - Redhat/CentOS

Python is only needed if you need to use Python and the PySpark engine in Fire Insights. Python modules in Fire Insights use Python 3.7+.

#### Check if Python 3.7+ is Installed

Use the below commands:

```
python --version
python3.7 --version
```

#### Install Python 3.7 (if not installed)

Some References for Installing Python:

- CentOS : https://tecadmin.net/install-python-3-7-on-centos/

#### Prerequisites

Python installation requires the GCC compiler to be available on the machine. Use the following command to install the prerequisites for installing Python.

> yum install gcc openssl-devel bzip2-devel libffi-devel zlib-devel

#### Download and extract the downloaded package

- **Download python from below Link**
  - https://www.python.org/downloads/
  - https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz

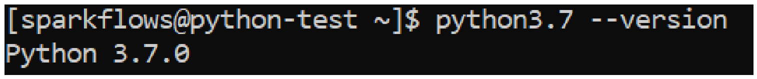
Download and untar:

```
wget https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz
tar xzf Python-3.7.0.tgz
```

### Compile Python source code

Compile the Python source code on your system using altinstall:

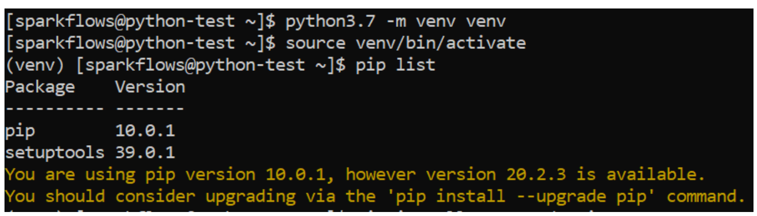
```
cd Python-3.7.0
./configure --enable-optimizations
make altinstall
python3.7 --version
```

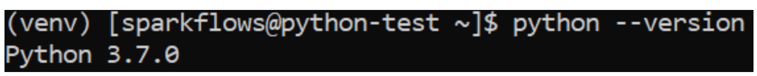



### Create Python virtual environment & Activate it

Create Python virtual environment & Activate it:

```
python3.7 -m venv venv
source venv/bin/activate
python --version
```




### Upgrade pip version

Upgrade pip version with 20.0 or above:

```
pip install pip --upgrade
```

### Install dependency for fbprophet package (CentOS 7)

### Run below command with sudo privilege

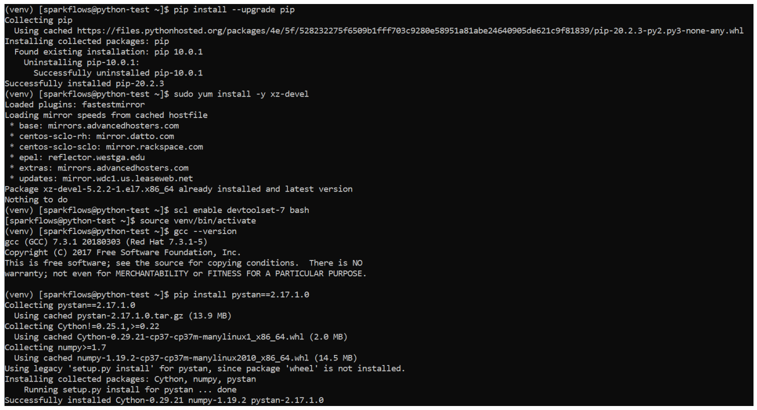
- Install development tool:

  ```
  yum install -y xz-devel
  ```

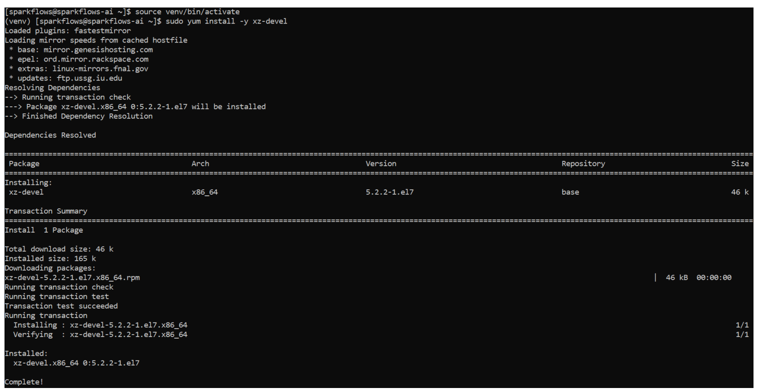
- Install the CentOS SCL release file:

  ```
  yum install centos-release-scl
  ```

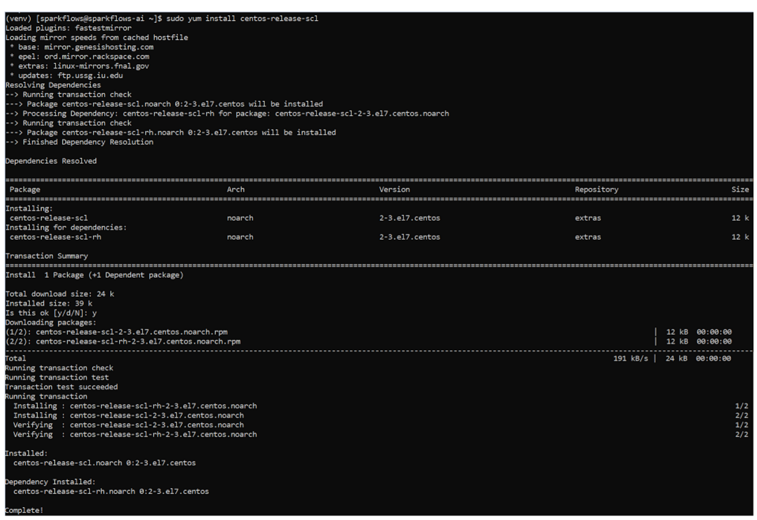
- Install Developer Toolset version 7:



```
(venv) [sparkflows@python-test ~]$ pip install --upgrade pip
Collecting pip
  Using cached https://files.pythonhosted.org/packages/4e/5f/528232275f6509b1fff703c9280e58951a81abe24640905de621c9f81839/pip-20.2.3-py2.py3-none-any.whl
Installing collected packages: pip
  Found existing installation: pip 10.0.1
    Uninstalling pip-10.0.1:
      Successfully uninstalled pip-10.0.1
Successfully installed pip-20.2.3
(venv) [sparkflows@python-test ~]$ sudo yum install -y xz-devel
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.advancedhosters.com
 * centos-sclo-rh: mirror.datto.com
 * centos-sclo-sclo: mirror.rackspace.com
 * epel: reflector.westga.edu
 * extras: mirrors.advancedhosters.com
 * updates: mirror.wdc1.us.leaseweb.net
Package xz-devel-5.2.2-1.el7.x86_64 already installed and latest version
Nothing to do
(venv) [sparkflows@python-test ~]$ scl enable devtoolset-7 bash
[sparkflows@python-test ~]$ source venv/bin/activate
(venv) [sparkflows@python-test ~]$ gcc --version
gcc (GCC) 7.3.1 20180303 (Red Hat 7.3.1-5)
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

(venv) [sparkflows@python-test ~]$ pip install pystan==2.17.1.0
Collecting pystan==2.17.1.0
  Using cached pystan-2.17.1.0.tar.gz (13.9 MB)
Collecting Cython!=0.25.1,>=0.22
  Using cached Cython-0.29.21-cp37-cp37m-manylinux1_x86_64.whl (2.0 MB)
Collecting numpy>=1.7
  Using cached numpy-1.19.2-cp37-cp37m-manylinux2010_x86_64.whl (14.5 MB)
Using legacy 'setup.py install' for pystan, since package 'wheel' is not installed.
Installing collected packages: Cython, numpy, pystan
    Running setup.py install for pystan ... done
Successfully installed Cython-0.29.21 numpy-1.19.2 pystan-2.17.1.0
```

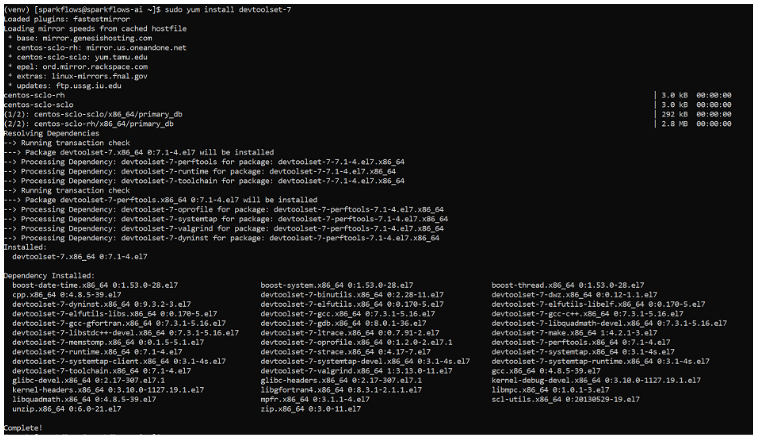


```
[sparkflows@sparkflows-ai ~]$ source venv/bin/activate
(venv) [sparkflows@sparkflows-ai ~]$ sudo yum install -y xz-devel
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.genesishosting.com
 * epel: ord.mirror.rackspace.com
 * extras: linux-mirrors.fnal.gov
 * updates: ftp.ussg.iu.edu
Resolving Dependencies
--> Running transaction check
---> Package xz-devel.x86_64 0:5.2.2-1.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package              Arch              Version              Repository    Size
================================================================================
Installing:
 xz-devel             x86_64            5.2.2-1.el7          base          46 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 46 k
Installed size: 165 k
Downloading packages:
xz-devel-5.2.2-1.el7.x86_64.rpm                          |  46 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : xz-devel-5.2.2-1.el7.x86_64                                  1/1
  Verifying  : xz-devel-5.2.2-1.el7.x86_64                                  1/1

Installed:
  xz-devel.x86_64 0:5.2.2-1.el7

Complete!
```



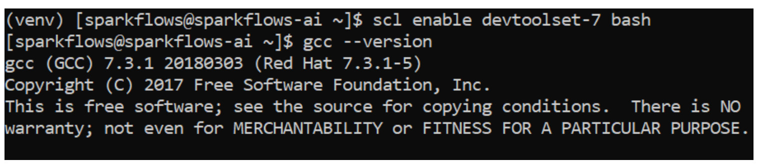
```
(venv) [sparkflows@sparkflows-ai ~]$ sudo yum install centos-release-scl
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.genesishosting.com
 * epel: ord.mirror.rackspace.com
 * extras: linux-mirrors.fnal.gov
 * updates: ftp.ussg.iu.edu
Resolving Dependencies
--> Running transaction check
---> Package centos-release-scl.noarch 0:2-3.el7.centos will be installed
--> Processing Dependency: centos-release-scl-rh for package: centos-release-scl-2-3.el7.centos.noarch
--> Running transaction check
---> Package centos-release-scl-rh.noarch 0:2-3.el7.centos will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                   Arch         Version              Repository    Size
================================================================================
Installing:
 centos-release-scl        noarch       2-3.el7.centos       extras        12 k
Installing for dependencies:
 centos-release-scl-rh     noarch       2-3.el7.centos       extras        12 k

Transaction Summary
================================================================================
Install  1 Package (+1 Dependent package)

Total download size: 24 k
Installed size: 39 k
Is this ok [y/d/N]: y
Downloading packages:
(1/2): centos-release-scl-2-3.el7.centos.noarch.rpm        |  12 kB  00:00:00
(2/2): centos-release-scl-rh-2-3.el7.centos.noarch.rpm     |  12 kB  00:00:00
--------------------------------------------------------------------------------
Total                                            191 kB/s  |  24 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : centos-release-scl-rh-2-3.el7.centos.noarch                  1/2
  Installing : centos-release-scl-2-3.el7.centos.noarch                     2/2
  Verifying  : centos-release-scl-2-3.el7.centos.noarch                     1/2
  Verifying  : centos-release-scl-rh-2-3.el7.centos.noarch                  2/2

Installed:
  centos-release-scl.noarch 0:2-3.el7.centos

Dependency Installed:
  centos-release-scl-rh.noarch 0:2-3.el7.centos

Complete!
```

```
yum install devtoolset-7
```



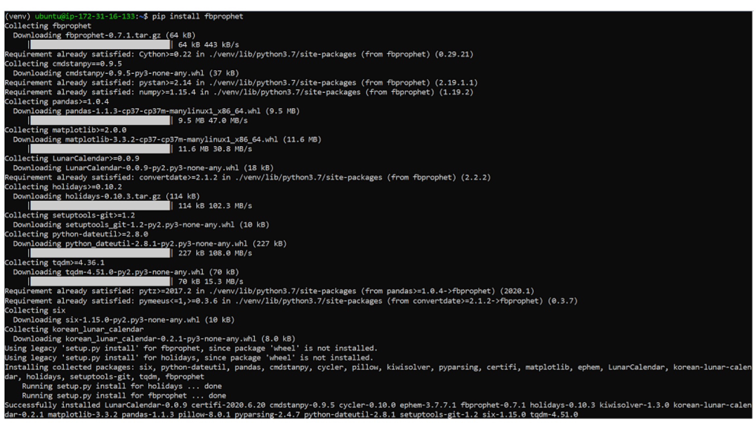
- launch a new shell instance using the Software Collection scl tool & Check GCC version:

```
scl enable devtoolset-7 bash
gcc --version``
```




- Install fbprophet package:

```
pip install fbprophet
```



- Check pip list:

```
pip list
```

## Reference

```
(venv) [sparkflows@python-test ~]$ pip list
Package               Version
--------------------- ---------
certifi               2020.6.20
cmdstanpy             0.9.5
convertdate           2.2.2
cycler                0.10.0
Cython                0.29.21
ephem                 3.7.7.1
fbprophet             0.7.1
holidays              0.10.3
kiwisolver            1.2.0
korean-lunar-calendar 0.2.1
LunarCalendar         0.0.9
matplotlib            3.3.2
numpy                 1.19.2
pandas                1.1.3
Pillow                7.2.0
pip                   20.2.3
PyMeeus               0.3.7
pyparsing             2.4.7
pystan                2.17.1.0
python-dateutil       2.8.1
pytz                  2020.1
setuptools            39.0.1
setuptools-git        1.2
six                   1.15.0
tqdm                  4.50.0
```

#### Links

- https://linuxize.com/post/how-to-install-gcc-compiler-on-centos-7/

#### Install Other Packages

Install the required packages:

```
cd fire-x.y.x/dist/fire
pip install -r requirements.txt
```

`requirements.txt` file is available in the installation directory of fire insights:

```
fire-x.y.x/dist/fire/requirements.txt
```

### Reference

#### Links

- https://docs.aws.amazon.com/cli/latest/userguide/install-linux-python.html
- https://aws.amazon.com/premiumsupport/knowledge-center/ec2-linux-python3-boto3/
- https://blog.teststation.org/centos/python/2016/05/11/installing-python-virtualenv-centos-7/

#### Delete a venv

To delete a virtual environment, follow below steps:

```
source venv/bin/activate
pip freeze > requirements.txt
pip uninstall -r requirements.txt -y
deactivate
rm -r venv/
```

#### Installing pip & wheel

- yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
- yum install python-pip
- yum install python-wheel

#### Add below in .bash_profile

- export PYSPARK_PYTHON=/usr/bin/python3
- export PYSPARK_DRIVER_PYTHON=/usr/bin/python3

#### For Ubuntu

- Ubuntu : https://docs.python-guide.org/starting/install3/linux/

### 2.1.7 Python Installation on MacOS

Python is only needed if you need to use Python and the PySpark engine in Fire Insights. Python modules in Fire Insights use Python 3.6+.

#### Check if Python is Installed

- python –version
- python3 –version

#### Install Python 3 (if not already there)

- **One way to install Python 3 on macOS is by installing Anaconda** https://docs.anaconda.com/anaconda/install/mac-os/
- Use *brew install python3*

#### Add below in .bash_profile

- alias python='python3'
- export PYSPARK_PYTHON=/usr/bin/python3
- export PYSPARK_DRIVER_PYTHON=/usr/bin/python3

**Sometimes a soft link to Pythons's executables is broken for some reason.** sudo ln -s /usr/bin/python3.x /usr/bin/python

#### Install Other Packages

Install the required python packages for Fire Insights:

- pip install -r requirements.txt

`requirements.txt` file is available in the installation directory of Fire Insights.

- fire-x.y.x/dist/fire/requirements.txt

### 2.1.8 Python Installation on Windows

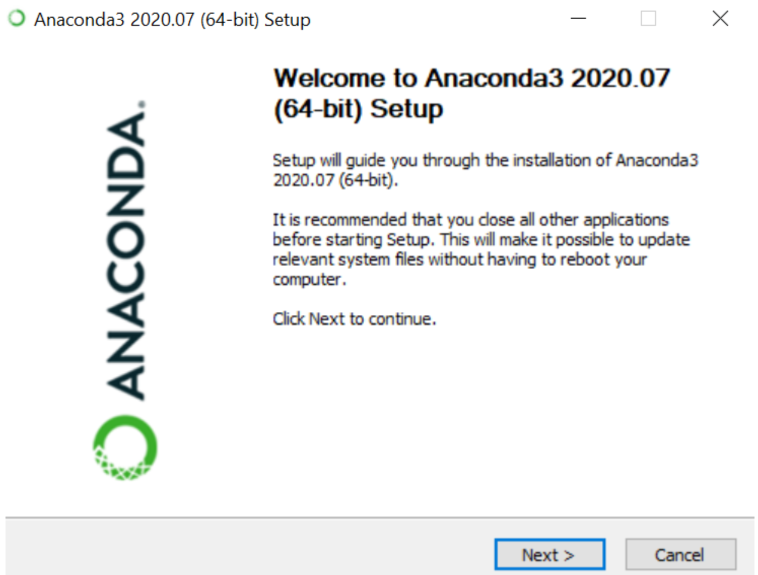
Python is only needed if you need to use Python and the PySpark engine in Fire Insights. Python modules in Fire Insights use Python 3.6+.

Below are steps for installing Anaconda.

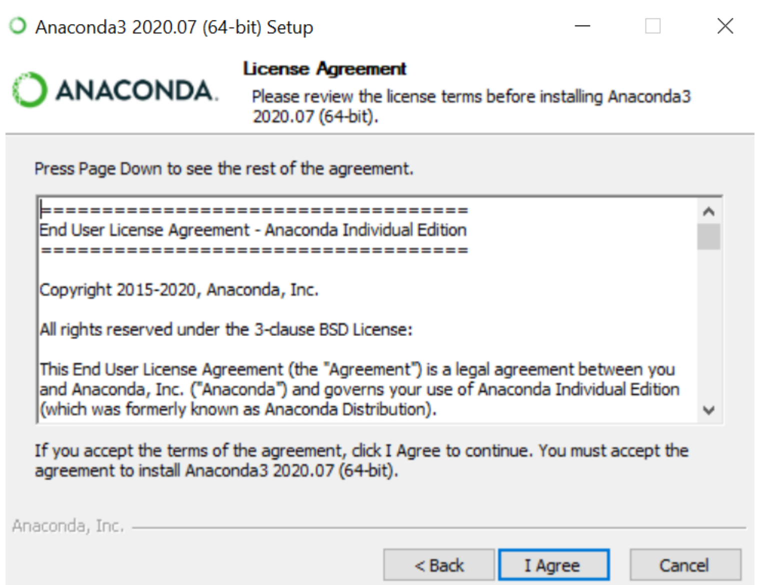
- **Download Anaconda from the below Link**
  - https://www.anaconda.com/products/individual
  - https://www.anaconda.com/products/individual#Downloads

Once the download completes, run the .exe installer

### Click Next to confirm the installation

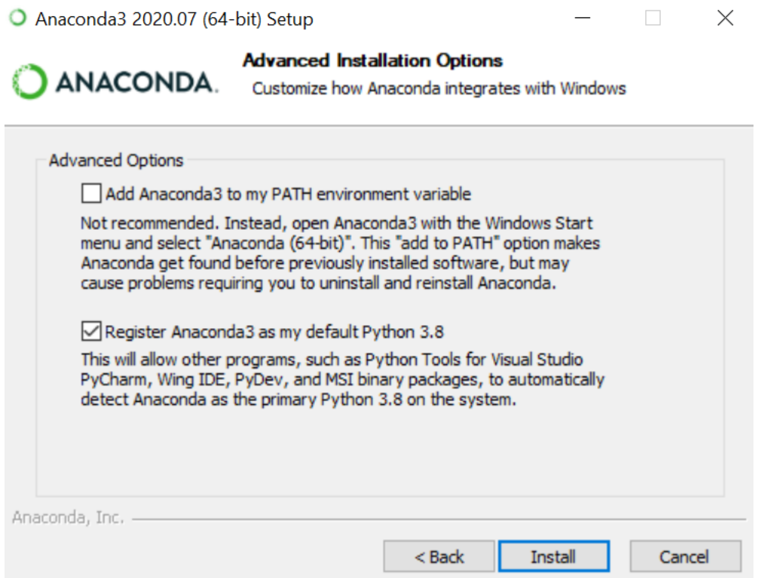



### Agree to the License



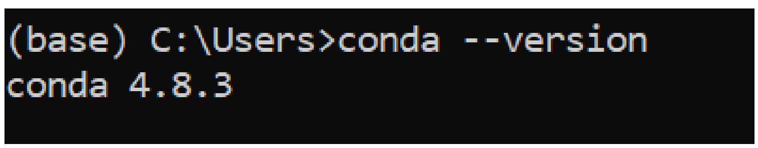

### Advanced Installation Options screen

It is recommended to not check "Add Anaconda to my PATH environment variable"




### Open the Anaconda Prompt from the Windows start menu

At the Anaconda prompt, check the `conda --version`




### Reference Link

- https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/

### Create virtual environment using conda

Run below command to Create virtual environment using conda.

- `conda create --name venv python=3.7`

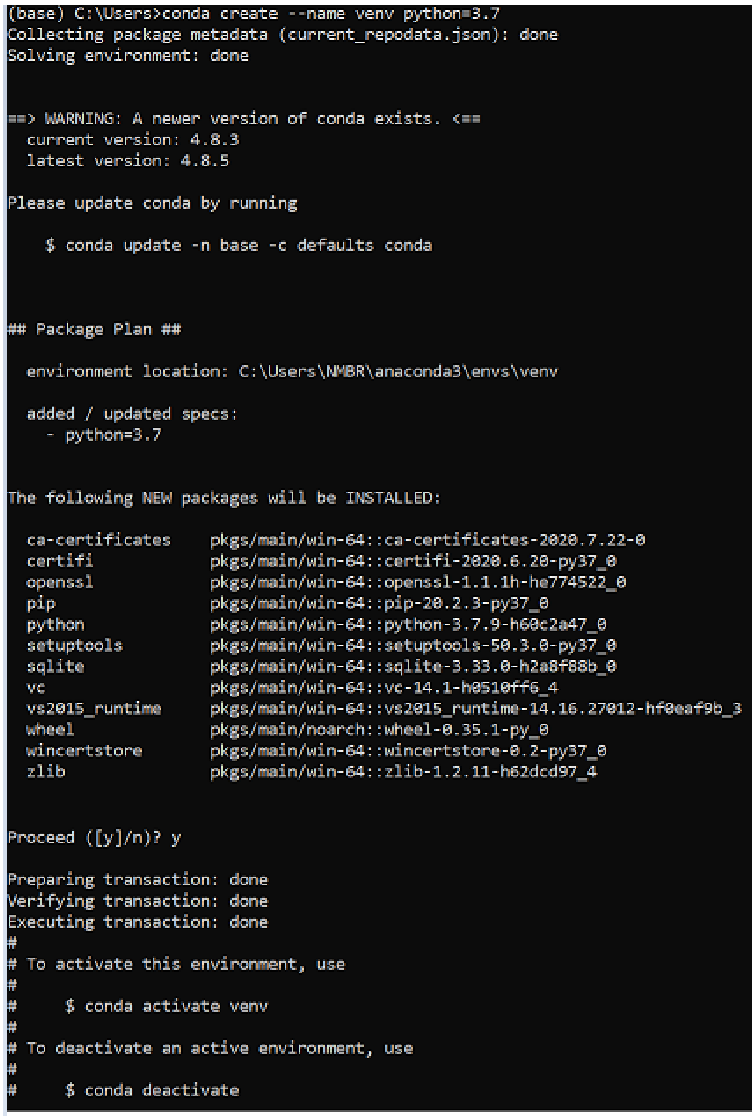
### Activate Virtual environment and Check list of python package

Run Below command to activate and check list of python package available by default.

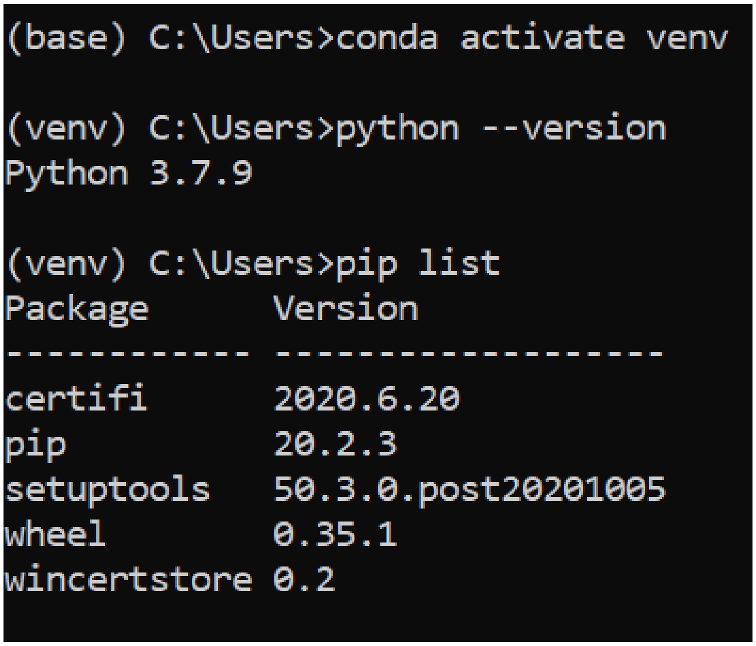
- `conda activate venv`
- `python --version`
- `pip list`

### Install Other Dependent Packages

Install the other required packages:

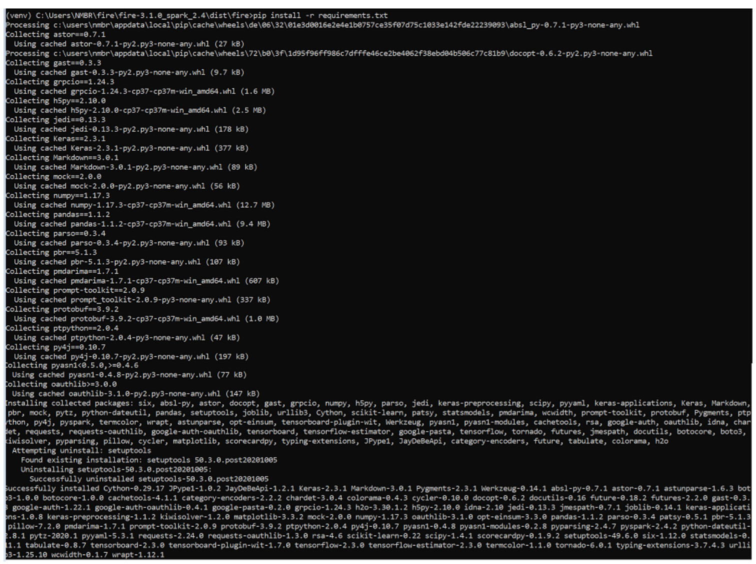


```
(base) C:\Users>conda create --name venv python=3.7
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.8.3
  latest version: 4.8.5

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: C:\Users\NMBR\anaconda3\envs\venv

  added / updated specs:
    - python=3.7


The following NEW packages will be INSTALLED:

  ca-certificates    pkgs/main/win-64::ca-certificates-2020.7.22-0
  certifi            pkgs/main/win-64::certifi-2020.6.20-py37_0
  openssl            pkgs/main/win-64::openssl-1.1.1h-he774522_0
  pip                pkgs/main/win-64::pip-20.2.3-py37_0
  python             pkgs/main/win-64::python-3.7.9-h60c2a47_0
  setuptools         pkgs/main/win-64::setuptools-50.3.0-py37_0
  sqlite             pkgs/main/win-64::sqlite-3.33.0-h2a8f88b_0
  vc                 pkgs/main/win-64::vc-14.1-h0510ff6_4
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.16.27012-hf0eaf9b_3
  wheel              pkgs/main/noarch::wheel-0.35.1-py_0
  wincertstore       pkgs/main/win-64::wincertstore-0.2-py37_0
  zlib               pkgs/main/win-64::zlib-1.2.11-h62dcd97_4


Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate venv
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```




- pip install -r requirements.txt

`requirements.txt` file is available in the installation directory of Fire Insights : fire-x.y.x/dist/fire/requirements.txt



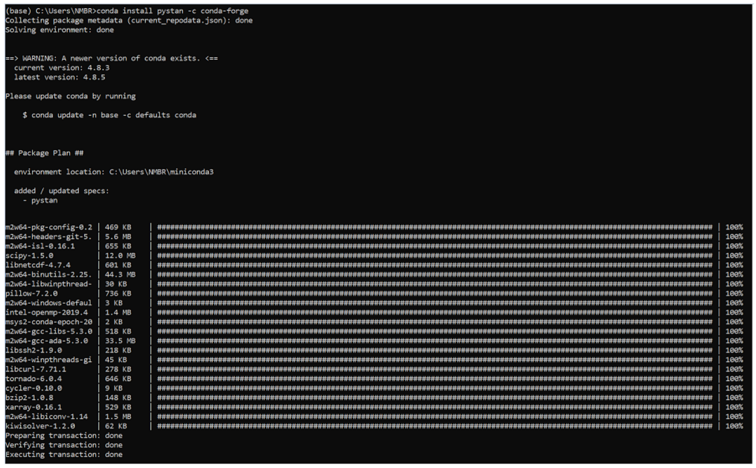
### Install dependency for fbprophet package (Windows 10)

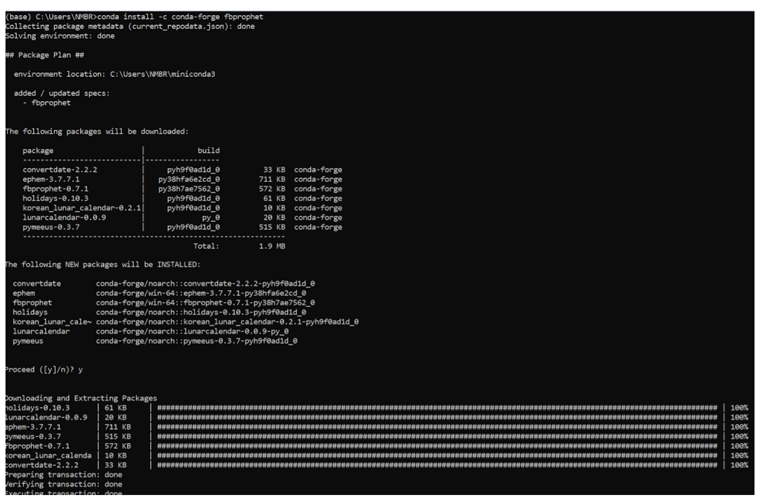
Install pystan:

- `conda install pystan -c conda-forge`

Install fbprophet:

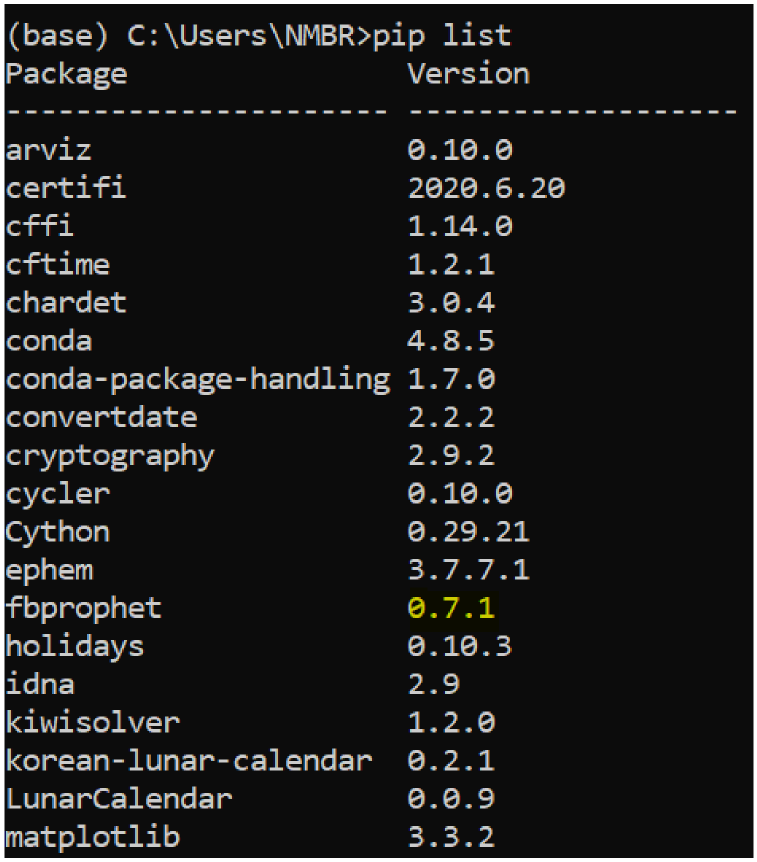
- `conda install -c conda-forge fbprophet`


(base) C:\Users\NMBR>conda install pystan -c conda-forge
Collecting package metadata (current_repodata.json): done
Solving environment: done
==> WARNING: A newer version of conda exists. <==
current version: 4.8.3
latest version: 4.8.5
Please update conda by running
$ conda update -n base -c defaults conda
## Package Plan ##
environment location: C:\Users\NMBR\miniconda3
added / updated specs:
- pystan
m2w64-pkg-config-0.2 | 469 KB | 100%
m2w64-headers-git-5. | 5.6 MB | 100%
m2w64-isl-0.16.1 | 655 KB | 100%
scipy-1.5.0 | 12.0 MB | 100%
libnetcdf-4.7.4 | 601 KB | 100%
m2w64-binutils-2.25. | 44.3 MB | 100%
m2w64-libwinpthread- | 30 KB | 100%
pillow-7.2.0 | 736 KB | 100%
m2w64-windows-defaul | 3 KB | 100%
intel-openmp-2019.4 | 1.4 MB | 100%
msys2-conda-epoch-20 | 2 KB | 100%
m2w64-gcc-libs-5.3.0 | 518 KB | 100%
m2w64-gcc-ada-5.3.0 | 33.5 MB | 100%
libssh2-1.9.0 | 218 KB | 100%
m2w64-winpthreads-gi | 45 KB | 100%
libcurl-7.71.1 | 278 KB | 100%
tornado-6.0.4 | 646 KB | 100%
cycler-0.10.0 | 9 KB | 100%
bzip2-1.0.8 | 148 KB | 100%
xarray-0.16.1 | 529 KB | 100%
m2w64-libiconv-1.14 | 1.5 MB | 100%
kiwisolver-1.2.0 | 62 KB | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done


(base) C:\Users\NMBR>conda install -c conda-forge fbprophet
Collecting package metadata (current_repodata.json): done
Solving environment: done
## Package Plan ##
environment location: C:\Users\NMBR\miniconda3
added / updated specs:
- fbprophet
The following packages will be downloaded:
package | build
convertdate-2.2.2 | pyh9f0ad1d_0 33 KB conda-forge
ephem-3.7.7.1 | py38hfa6e2cd_0 711 KB conda-forge
fbprophet-0.7.1 | py38h7ae7562_0 572 KB conda-forge
holidays-0.10.3 | pyh9f0ad1d_0 61 KB conda-forge
korean_lunar_calendar-0.2.1 | pyh9f0ad1d_0 10 KB conda-forge
lunarcalendar-0.0.9 | py_0 20 KB conda-forge
pymeeus-0.3.7 | pyh9f0ad1d_0 515 KB conda-forge
Total: 1.9 MB
The following NEW packages will be INSTALLED:
convertdate conda-forge/noarch::convertdate-2.2.2-pyh9f0ad1d_0
ephem conda-forge/win-64::ephem-3.7.7.1-py38hfa6e2cd_0
fbprophet conda-forge/win-64::fbprophet-0.7.1-py38h7ae7562_0
holidays conda-forge/noarch::holidays-0.10.3-pyh9f0ad1d_0
korean_lunar_cale~ conda-forge/noarch::korean_lunar_calendar-0.2.1-pyh9f0ad1d_0
lunarcalendar conda-forge/noarch::lunarcalendar-0.0.9-py_0
pymeeus conda-forge/noarch::pymeeus-0.3.7-pyh9f0ad1d_0
Proceed ([y]/n)? y
Downloading and Extracting Packages
holidays-0.10.3 | 61 KB | 100%
lunarcalendar-0.0.9 | 20 KB | 100%
ephem-3.7.7.1 | 711 KB | 100%
pymeeus-0.3.7 | 515 KB | 100%
fbprophet-0.7.1 | 572 KB | 100%
korean_lunar_calenda | 10 KB | 100%
convertdate-2.2.2 | 33 KB | 100%
Preparing transaction: done
Verifying transaction: done

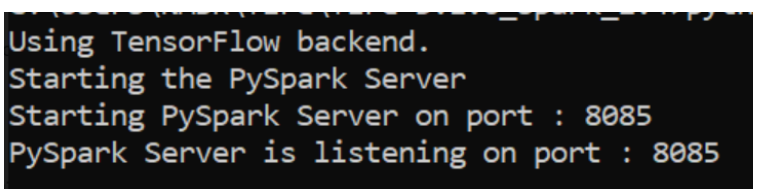
Check the version of fbprophet Installed:

- `pip list`




Once the above steps have completed successfully, run the below command to ensure everything was setup correctly.

- `python ./dist/__main__.py`

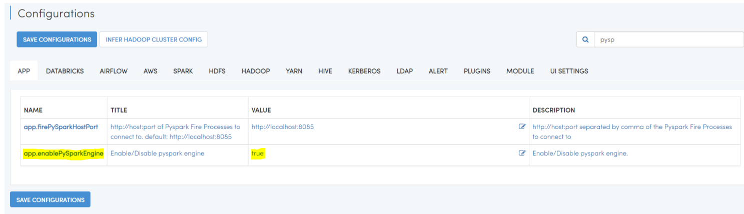



### Enable PySpark Engine in Fire Insights

Login to Fire Insights application and go to configurations and set `app.enablePySparkEngine` to `true` and save the changes. Now you can start using PySpark engine in Fire Insights.

### Removing Conda virtual Environment

- `conda deactivate`
- `conda env remove --name name of virtual environment`
- Delete those package from exact location.




## 2.1.9 Python Installation on Ubuntu

Python is only needed if you need to use Python and the PySpark engine in Fire Insights. Python modules in Fire Insights use Python 3.7+.

### Check if Python 3.7+ is Installed

Use the below commands:

```
python --version
python3.7 --version
```

### Install Python 3.7 (if not installed)

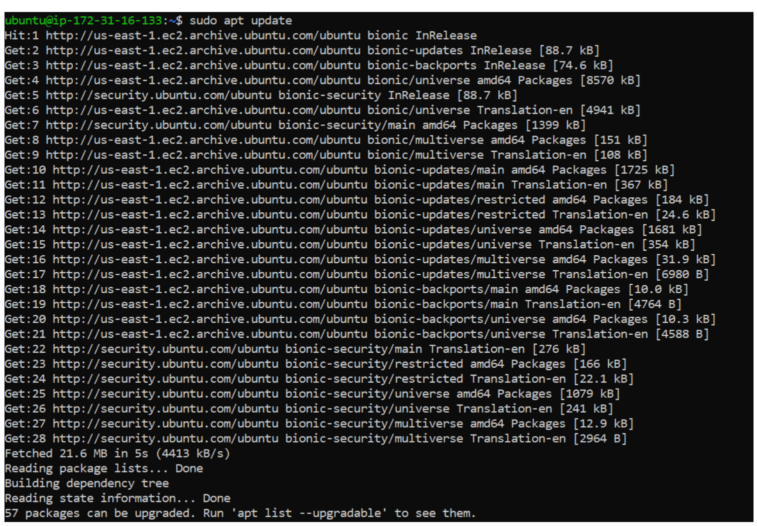
Some References for Installing Python:

- Ubuntu : https://linuxize.com/post/how-to-install-python-3-7-on-ubuntu-18-04/

### Prerequisites

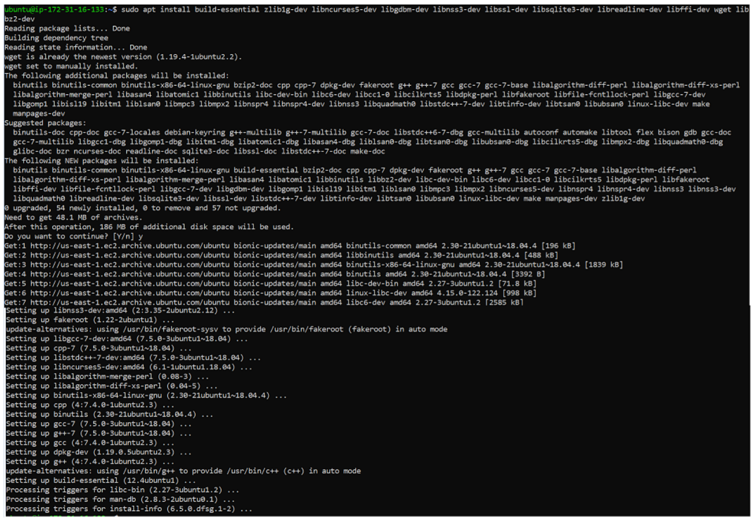
update the packages list and install the packages necessary to build Python source:

```
sudo apt update
```




- Install needed dependency:

```
sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-
↪dev libssl-dev libsqlite3-dev libreadline-dev libffi-dev wget libbz2-dev``
```

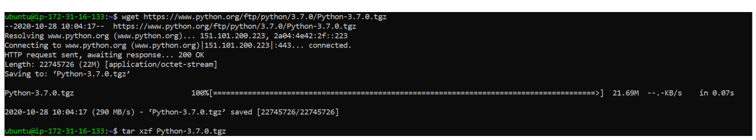


### Download and extract the downloaded package

- **Download python from below Link**
  - https://www.python.org/downloads/
  - https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz

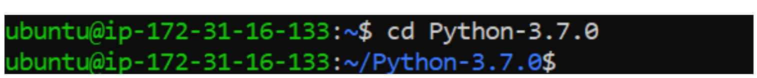
Download and untar:

```
wget https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz
tar xzf Python-3.7.0.tgz
```




Next, navigate to the Python source directory and run the configure script which will perform a number of checks to make sure all of the dependencies on your system are present:

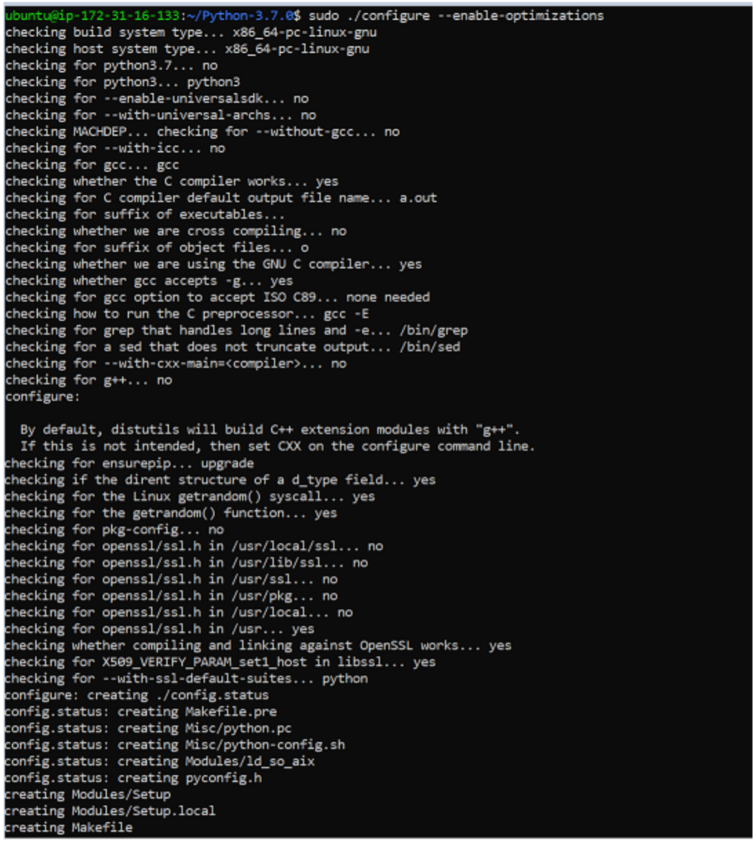
```
cd Python-3.7.0
```




- Build & compile:

  ```
  ./configure --enable-optimizations
  ```

- Install the Python binaries by running the following command:



```
ubuntu@ip-172-31-16-133:~/Python-3.7.0$ sudo ./configure --enable-optimizations
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for python3.7... no
checking for python3... python3
checking for --enable-universalsdk... no
checking for --with-universal-archs... no
checking MACHDEP... checking for --without-gcc... no
checking for --with-icc... no
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /bin/grep
checking for a sed that does not truncate output... /bin/sed
checking for --with-cxx-main=<compiler>... no
checking for g++... no
configure:

  By default, distutils will build C++ extension modules with "g++".
  If this is not intended, then set CXX on the configure command line.
checking for ensurepip... upgrade
checking if the dirent structure of a d_type field... yes
checking for the Linux getrandom() syscall... yes
checking for the getrandom() function... yes
checking for pkg-config... no
checking for openssl/ssl.h in /usr/local/ssl... no
checking for openssl/ssl.h in /usr/lib/ssl... no
checking for openssl/ssl.h in /usr/ssl... no
checking for openssl/ssl.h in /usr/pkg... no
checking for openssl/ssl.h in /usr/local... no
checking for openssl/ssl.h in /usr... yes
checking whether compiling and linking against OpenSSL works... yes
checking for X509_VERIFY_PARAM_set1_host in libssl... yes
checking for --with-ssl-default-suites... python
configure: creating ./config.status
config.status: creating Makefile.pre
config.status: creating Misc/python.pc
config.status: creating Misc/python-config.sh
config.status: creating Modules/ld_so_aix
config.status: creating pyconfig.h
creating Modules/Setup
creating Modules/Setup.local
creating Makefile
```

```
make altinstall
```

Note: Do not use the standard make install as it will overwrite the default system python3 binary.

Verify it by typing:

```
python3.7 --version
```

### Create Python virtual environment & Activate it

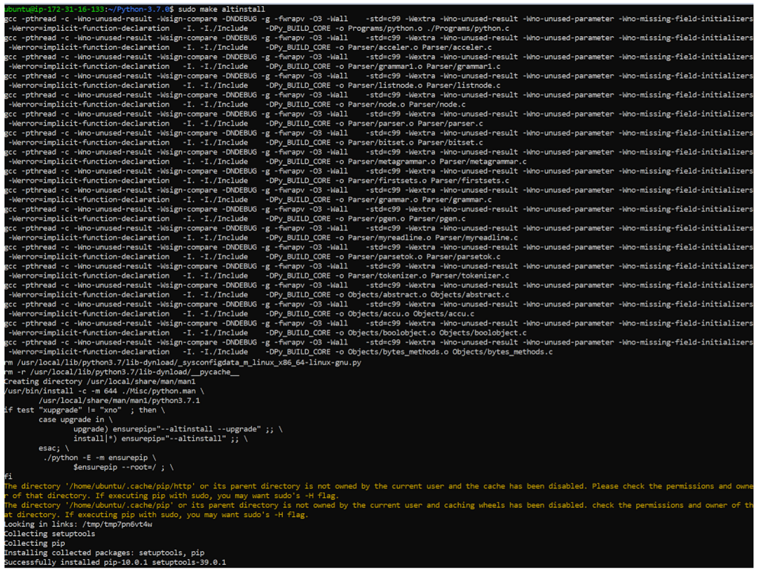
Create Python virtual environment & Activate it:

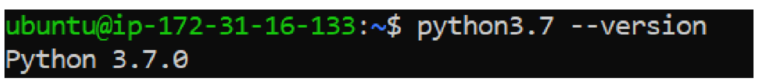
```
python3.7 -m venv venv
source venv/bin/activate
python --version
```

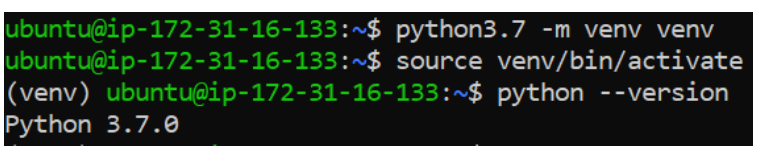
### Upgrade pip version

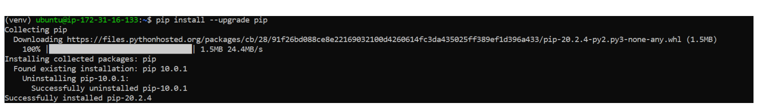
Upgrade pip version with 20.0 or above:

```
pip install pip --upgrade
```


ubuntu@ip-172-31-16-133:~/Python-3.7.0$ sudo make altinstall
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Programs/python.o ./Programs/python.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/acceler.o Parser/acceler.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/grammar1.o Parser/grammar1.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/listnode.o Parser/listnode.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/node.o Parser/node.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/parser.o Parser/parser.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/bitset.o Parser/bitset.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/metagrammar.o Parser/metagrammar.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/firstsets.o Parser/firstsets.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/grammar.o Parser/grammar.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/pgen.o Parser/pgen.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/myreadline.o Parser/myreadline.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/parsetok.o Parser/parsetok.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Parser/tokenizer.o Parser/tokenizer.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Objects/abstract.o Objects/abstract.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Objects/accu.o Objects/accu.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Objects/boolobject.o Objects/boolobject.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -Wall -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-field-initializers
-Werror=implicit-function-declaration -I. -I./Include -DPy_BUILD_CORE -o Objects/bytes_methods.o Objects/bytes_methods.c
rm /usr/local/lib/python3.7/lib-dynload/_sysconfigdata_m_linux_x86_64-linux-gnu.py
rm -r /usr/local/lib/python3.7/lib-dynload/__pycache__
Creating directory /usr/local/share/man/man1
/usr/bin/install -c -m 644 ./Misc/python.man \
/usr/local/share/man/man1/python3.7.1
if test "xupgrade" != "xno" ; then \
case upgrade in \
upgrade) ensurepip="--altinstall --upgrade" ;; \
install|*) ensurepip="--altinstall" ;; \
esac; \
./python -E -m ensurepip \
$ensurepip --root=/ ; \
fi
The directory '/home/ubuntu/.cache/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please check the permissions and owne
r of that directory. If executing pip with sudo, you may want sudo's -H flag.
The directory '/home/ubuntu/.cache/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. check the permissions and owner of th
at directory. If executing pip with sudo, you may want sudo's -H flag.
Looking in links: /tmp/tmp7pn6vt4w
Collecting setuptools
Collecting pip
Installing collected packages: setuptools, pip
Successfully installed pip-10.0.1 setuptools-39.0.1


ubuntu@ip-172-31-16-133:~$ python3.7 --version
Python 3.7.0

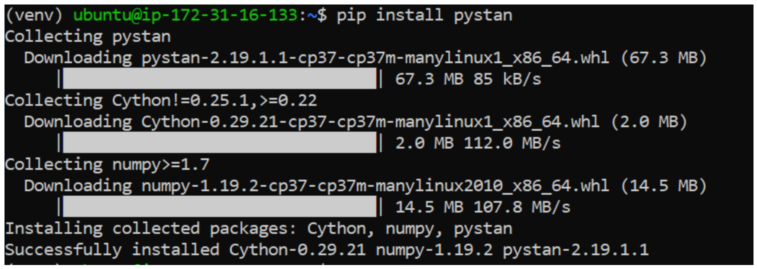

ubuntu@ip-172-31-16-133:~$ python3.7 -m venv venv
ubuntu@ip-172-31-16-133:~$ source venv/bin/activate
(venv) ubuntu@ip-172-31-16-133:~$ python --version
Python 3.7.0


(venv) ubuntu@ip-172-31-16-133:~$ pip install --upgrade pip
Collecting pip
Downloading https://files.pythonhosted.org/packages/cb/28/91f26bd088ce8e22169032100d4260614fc3da435025ff389ef1d396a433/pip-20.2.4-py2.py3-none-any.whl (1.5MB)
100% | | 1.5MB 24.4MB/s
Installing collected packages: pip
Found existing installation: pip 10.0.1
Uninstalling pip-10.0.1:
Successfully uninstalled pip-10.0.1
Successfully installed pip-20.2.4

### Install dependency for fbprophet package (Ubuntu 18.04)

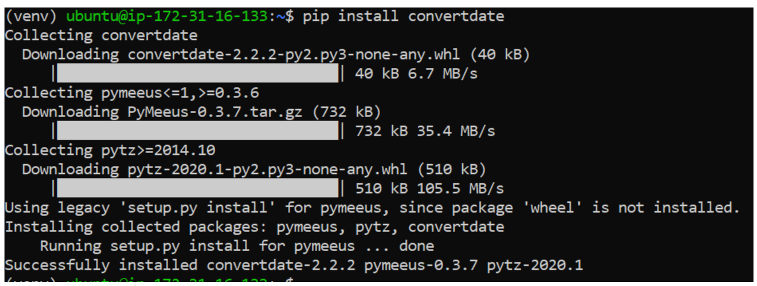
- pystan dependency:

```
pip install pystan
```



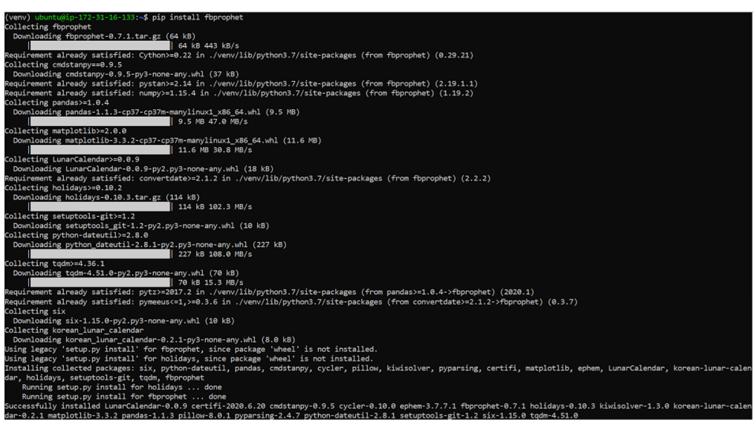

- convertdate dependency:

```
pip install convertdate
```




- fbprophet dependency:

```
pip install fbprophet
```

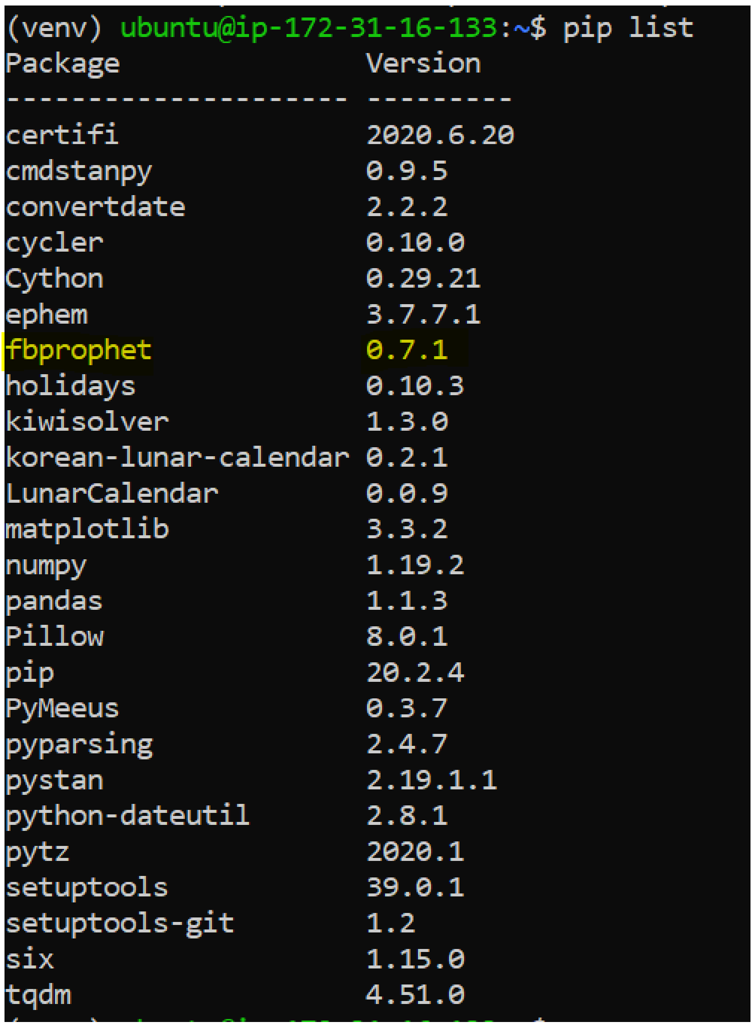


- Check pip list:

```
pip list
```

### Install Other Packages

Install the required packages:



```
(venv) ubuntu@ip-172-31-16-133:~$ pip list
Package               Version
--------------------- ---------
certifi               2020.6.20
cmdstanpy             0.9.5
convertdate           2.2.2
cycler                0.10.0
Cython                0.29.21
ephem                 3.7.7.1
fbprophet             0.7.1
holidays              0.10.3
kiwisolver            1.3.0
korean-lunar-calendar 0.2.1
LunarCalendar         0.0.9
matplotlib            3.3.2
numpy                 1.19.2
pandas                1.1.3
Pillow                8.0.1
pip                   20.2.4
PyMeeus               0.3.7
pyparsing             2.4.7
pystan                2.19.1.1
python-dateutil       2.8.1
pytz                  2020.1
setuptools            39.0.1
setuptools-git        1.2
six                   1.15.0
tqdm                  4.51.0
```

```
cd fire-x.y.x/dist/fire
pip install -r requirements.txt
```

`requirements.txt` file is available in the installation directory of fire insights:

```
fire-x.y.x/dist/fire/requirements.txt
```

#### Delete a venv

To delete a virtual environment, follow below steps:

```
source venv/bin/activate
pip freeze > requirements.txt
pip uninstall -r requirements.txt -y
deactivate
rm -r venv/
```

### 2.1.10 Running Diagnostics

#### Linux

Fire Insights needs jdk 1.8 to be available

- java -version

java version "1.8.0_101"

#### Mac OS

Fire Insights needs jdk 1.8 to be available

- java -version

java version "1.8.0_101"

#### Windows

Fire Insights needs jdk 1.8 to be available

- java -version

java version "1.8.0_101"

# CHAPTER 3

# Configuration

## 3.1 Configuration

### 3.1.1 Database Setup

**Fire stores metadata in a Relational Database. These include:**

- Applications
- Dataset Definitions
- Workflows
- Users
- Groups
- Roles
- Dashboards

**Below are the details for setting up a database for Fire:**

#### H2 Database

Fire can be setup up to easily run with H2 database. Fire runs H2 in embedded mode. The H2 database is used for storing the metadata of the DataSets, Workflows, Dashboards, Users, Groups, Roles etc.

More details of the H2 database can be found here : http://www.h2database.com/html/main.html

If you are want to run multiple instances of Fire for high availability, configure Fire to run with MySQL.

#### Creating/Upgrading the H2 database

Execute the following steps on your CLI:

- Mac/Linux:

```
cd <install_dir>/fire-x.y.z
./create-h2-db.sh
```

- Windows:

```
cd <install_dir>\fire-x.y.z
.\create-h2-db.bat
```

If you would like to use different values for the db, username, password, update them in `<install_dir>/fire-1.4.0/conf/db.properties`:

```
spring.datasource.url = jdbc:h2:file:~/firedb

spring.datasource.username = fire

spring.datasource.password = fire

spring.datasource.driverClassName = org.h2.Driver
```

**Note:** firedb is created in the users home directory and the name is firedb.mv.db

### Recreating H2 database

If you need to recreate the H2 database, follow the steps below to create a new empty H2 DB:

```
Stop the running Fire server

Move the existing firedb files to another temp location on your disk

Recreate the H2 DB using the steps in the above section for creating a brand new␣
↪empty H2 DB for Fire
```

### MySQL Database

Fire can easily be setup up to run with MySQL

More details of the MySQL database can be found here : https://www.mysql.com/

### Install MySQL

- Install MySQL on a machine.
- It might be easier to install it on the same machine you are installing Fire on.

### Create the DB for Fire in MySQL

- Create the database for Fire in MySQL
- Let us call it `firedb`:

```
create database firedb;
```

### Create the User for Fire in MySQL and grant it Permissions

Create the User for Fire in MySQL:

```
CREATE user 'fire'@'%' IDENTIFIED BY 'fire';

GRANT ALL PRIVILEGES ON firedb.* TO 'fire'@'%' WITH GRANT OPTION;
```

- In `CREATE` user, the user we are creating is `fire` who is allowed to access the database from anywhere `%` and his password is `fire`.
- Next, this user has been granted all `permissions`. This, of course can be further restricted based on your use case.

### Configure Fire to connect to MySQL

- Copy `db.properties.mysql` file into the `conf` directory as `db.properties`:

```
cd   fire-x.y.z
cp   conf.orig/db.properties.mysql   conf/db.properties
```

- Update the following fields in `conf/db.properties` based on the values you used in creating the DB for fire. The below assumes that the database name you created for Fire is `firedb`. It also assumes that MySQL has been installed on the `same machine` as Fire:

```
# Connection url for the database "firedb"

spring.datasource.url=jdbc:mysql://localhost:3306/firedb

spring.datasource.driverClassName=com.mysql.jdbc.Driver

spring.jpa.database=MYSQL

# Username and password

spring.datasource.username=fire

spring.datasource.password=fire
```

### Install the MySQL Connector Jar file

- Download the MySQL JDBC driver from http://www.mysql.com/downloads/connector/j/5.1.html
- Extract the `JDBC driver JAR file` from the downloaded file. For example:

```
tar zxvf mysql-connector-java-8.0.11.tar.gz
```

- just copy the path location for `` `JDBC driver JAR file ``
- copy the mysql JDBC driver JAR file to the `fire-server-lib` directory of `fire-x.y.z`:

```
cd fire-x.y.z
cp /pathlocation of jdbc jar file/mysql-connector-java.jar      fire-server-lib
```

### Create the Tables for Fire in MySQL

- Create the tables for Fire in MySQL by executing the `create-mysql-db.sh` script:

```
cd fire-x.y.z

./create-mysql-db.sh
```

### Troubleshooting

MySQL has a problem where one of the default users in the user table is `'' @ localhost`, which winds up denying all localhost users later in the table. If you are accessing mysql from localhost, assuming Fire and MySQL have been installed on the same machine, then you need to delete this entry in `mysql.user` table:

```
select user, host from user where user = ''

#you should see an entry for this and host equals localhost.

DELETE from user WHERE user = '' AND host = 'localhost';

flush privileges;

#this reloads privileges - important step. otherwise you will get access denied error␣
↪even though you log in with the correct user.
```

Here is a link on stackoverflow that talks about this:

> http://stackoverflow.com/questions/1412339/cannot-log-in-with-created-user-in-mysql

### Microsoft SQL Server Database

Fire can easily be setup up to run with Microsoft SQL Server.

More details of the Microsoft SQL Server database can be found here : https://www.microsoft.com/en-us/sql-server/default.aspx

### Install Microsoft SQL Server

- Install Microsoft SQL Server on a machine.
- It might be easier to install it on the same machine you are installing Fire on.

### Create the DB for Fire in Microsoft SQL Server

- Create the database for Fire in Microsoft SQL Server
- Let us call it `firedb`:

```
CREATE DATABASE firedb;
```

### Create the User for Fire in Microsoft SQL Server and grant it Permissions

Create the User for Fire in Microsoft SQL Server and give it Permissions.

### Configure Fire to connect to Microsoft SQL Server

- Copy `db.properties.sqlserver` file into the `conf` directory as `db.properties`:

```
cd   fire-x.y.z
cp   conf.orig/db.properties.sqlserver   conf/db.properties
```

- Update the following fields in `conf/db.properties` based on the values you used in creating the DB for fire. The below assumes that the database name you created for Fire is `firedb`. It also assumes that Microsoft SQL Server has been installed on the `same machine` as Fire:

```
# Connection url for the database "firedb"

spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=firedb

spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver

spring.jpa.database=SQLSERVER

# Username and password

spring.datasource.username=fire

spring.datasource.password=fire

spring.jpa.hibernate.dialect=org.hibernate.dialect.SQLServer2008Dialect
```

### Install the Microsoft SQL Server Connector Jar file

- Download the Microsoft SQL Server JDBC driver from https://www.microsoft.com/en-us/download/details.aspx?id=11774
- Untar the file `sqljdbc_6.0.8112.200_enu.tar.gz`
- You will get JDBC jar file on untaring `sqljdbc42.jar`
- Copy the Microsoft SQL Server JDBC driver JAR file to the `fire-server-lib` directory of `fire-x.y.z`

### Create the Tables for Fire in Microsoft SQL Server

- Tables in Microsoft SQL Server can be created by using the DDL script : `db/sqlserver/fire-schema.sqlserver.sql`
- They can also be created by executing the `create-sqlserver-db.sh` script:

```
cd fire-x.y.z

./create-sqlserver-db.sh
```

### Aurora MySQL Database

Fire can easily be setup up to run with Aurora MySQL

More details of the Aurora MySQL database can be found here : https://aws.amazon.com/rds/aurora/

#### Create Aurora MySQL database on AWS

- Login to AWS.
- Create Aurora MySQL Database which is accessible from machine where Fire is running.

#### Create the DB for Fire in Aurora MySQL

- Create the database for Fire in Aurora MySQL
- Let us call it `firedb`:

```
create database firedb;
```

#### Create the User for Fire in Aurora MySQL and grant it Permissions

Create the User for Fire in MySQL:

```
CREATE user 'fire'@'%' IDENTIFIED BY 'fire';

GRANT ALL PRIVILEGES ON firedb.* TO 'fire'@'%' WITH GRANT OPTION;
```

- In `CREATE` user, the user we are creating is `fire` who is allowed to access the database from anywhere `%` and his password is `fire`.
- Next, this user has been granted all `permissions`. This, of course can be further restricted based on your use case.

#### Configure Fire to connect to Aurora MySQL

- Copy `db.properties.mysql` file into the `conf` directory as `db.properties`:

```
cd   fire-x.y.z
cp   conf.orig/db.properties.mysql   conf/db.properties
```

- Update the following fields in `conf/db.properties` based on the values you used in creating the DB for fire. The below assumes that the database name you created for Fire is `firedb`. It also assumes that MySQL has been installed on the `same machine` as Fire:

```
# Connection url for the database "firedb"

spring.datasource.url=jdbc:mysql://Endpoint:3306/firedb

spring.datasource.driverClassName=com.mysql.jdbc.Driver

spring.jpa.database=MYSQL

# Username and password

spring.datasource.username=fire

spring.datasource.password=fire
```

### Install the MySQL Connector Jar file

- Download the MySQL JDBC driver from http://www.mysql.com/downloads/connector/j/5.1.html
- Extract the `JDBC driver JAR file` from the downloaded file. For example:

```
tar zxvf mysql-connector-java-8.0.11.tar.gz
```

- just copy the path location for `JDBC driver JAR file
- copy the mysql JDBC driver JAR file to the `fire-server-lib` directory of `fire-x.y.z`:

```
cd fire-x.y.z
cp /pathlocation_of_jdbc_jar_file/mysql-connector-java.jar  fire-server-lib
```

### Create the Tables for Fire in Aurora

- Create the tables for Fire in MySQL by executing the `create-mysql-db.sh` script:

```
cd fire-x.y.z

./create-mysql-db.sh
```

### Troubleshooting

MySQL has a problem where one of the default users in the user table is `'' @ localhost`, which winds up denying all localhost users later in the table. If you are accessing mysql from localhost, assuming Fire and MySQL have been installed on the same machine, then you need to delete this entry in `mysql.user` table:

```
select user, host from user where user = ''

#you should see an entry for this and host equals localhost.

DELETE from user WHERE user = '' AND host = 'localhost';

flush privileges;

#this reloads privileges - important step. otherwise you will get access denied error␣
↪even though you log in with the correct user.
```

Here is a link on stackoverflow that talks about this:

http://stackoverflow.com/questions/1412339/cannot-log-in-with-created-user-in-mysql

## 3.1.2 Connecting to Apache Spark Cluster

### Overview

Fire can be configured to submit the spark jobs to run on an Apache Spark Cluster.

- **Install Fire on an edge node of your Apache Spark Cluster.**
  - The edge node has the hadoop/hive/spark configuration files set up.
  - Make sure that you are already able to run your spark jobs from this node using **spark-submit**.
- **Update the below configurations under the menu, ''Administration/Configuration''**

---

**Note:** In order for Fire to connect to the Apache Spark Cluster, it needs to be installed as a user which can impersonate other users. More details are below in the page. For the rest of the documentation on this page, we assume that it has been installed as the user `sparkflows`.

---

### Fire User Setup

The user with which Fire is running has to be a proxy user in HDFS. That way it can impersonate the logged in user.

Below are the steps for setting the Fire user to be a proxy user on HDFS.

### Update core-site.xml of Hadoop to allow Fire user to impersonate

https://www.cloudera.com/documentation/enterprise/5-8-x/topics/admin_hdfs_proxy_users.html

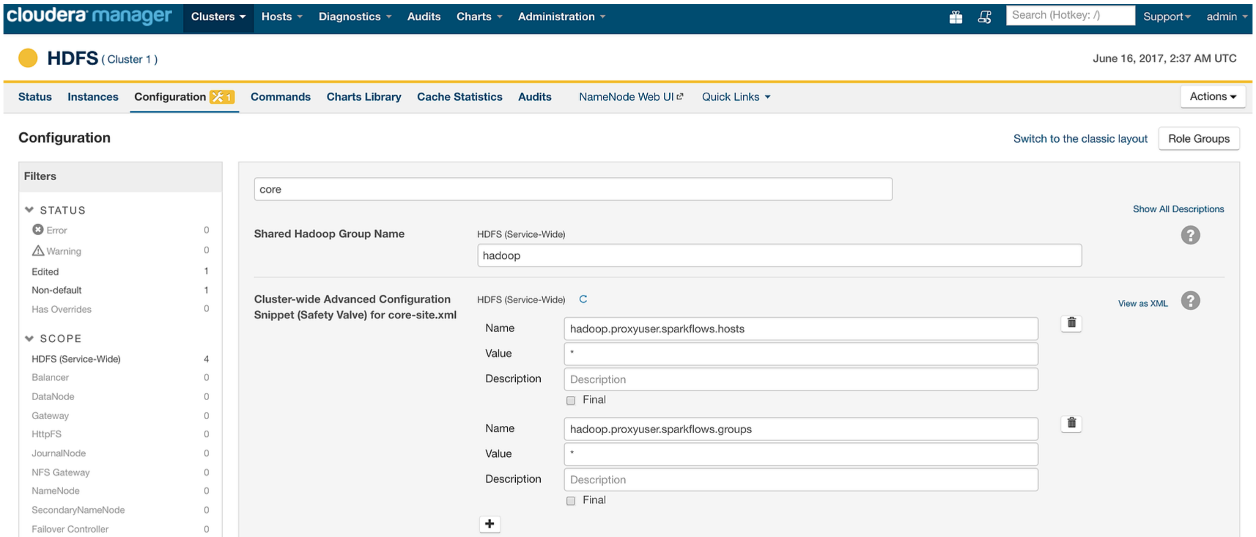
- In your core-site.xml file for Hadoop, allow sparkflows user to impersonate other users. Without impersonation enabled for this user, your Sparkflows application users trying to run jobs against a hadoop cluster would not be able to do so.
- Also, allow the appropriate groups that the sparkflows users will be able to impersonate belong to.
- In the example below, user `sparkflows` is allowed to impersonate users from hosts `host1` and `host2`. The users being impersonated belong to the groups `hive,hfs,hadoop,spark`. Your permissions are likely going to be different and more restrictive.

Below is an example:

```
<property>
   <name>hadoop.proxyuser.sparkflows.hosts</name>
   <value>host1,host2</value>
</property>

<property>
   <name>hadoop.proxyuser.sparkflows.groups</name>
   <value>hive,hfs,hadoop,spark</value>
 </property>
```

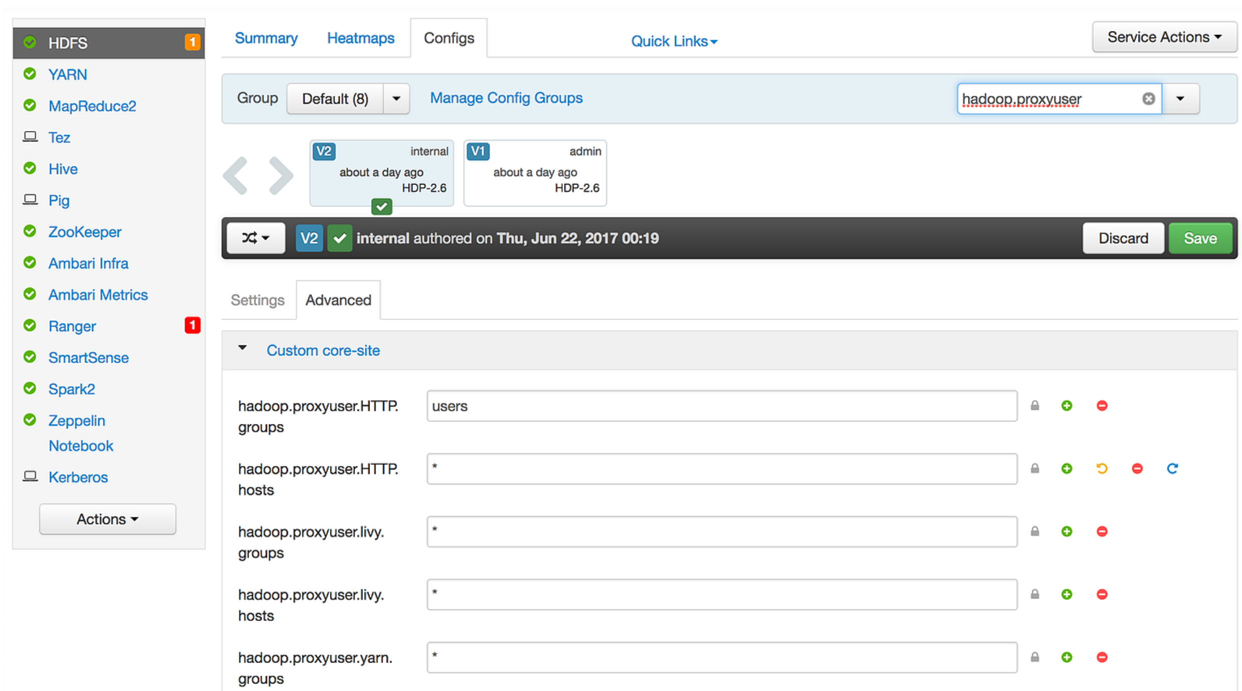
### Cloudera Manager

If you are using Cloudera Manager, you can set the above settings for impersonation in `HDFS/Configuration`.



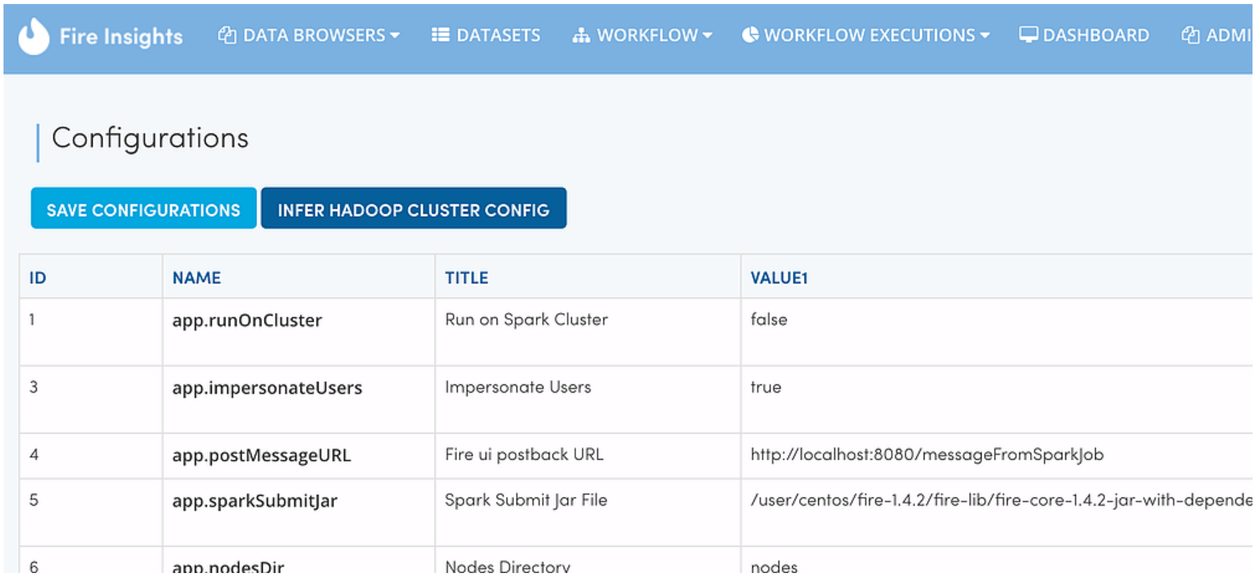

### Ambari

If you are using Ambari, you can set the above settings for impersonation in `HDFS/Configuration under Custom core-site`



### Infer Hadoop Configs

**Infer Hadoop Configs** button under Administration/Configuration automatically infers some of the configurations of the cluster from the hadoop config files on the edge node to help with the process. Use it to get the initial set of configurations.




### Fire Configurations for connecting to an Apache Spark Cluster

Below are the configuration details for connecting Fire to an Apache Spark Cluster.

| Parameter | Value | Description |
|---|---|---|
| app.runOnCluster | true | Indicate to run on the spark cluster. By default it is set to false |
| app.postMessageURL | http://localhost:8080/messageFromSparkJob | Indicate the URL on fire server which receives messages from the spark jobs running on the cluster. Set localhost to the machine name on which Fire is running. Replace 8080 with the port number on which Fire is running. |
| app.sparkSubmitJar | /home/centos/fire-2.1.0/fire-lib/fire-core-2.1.0-jar-with-dependencies.jar | fire-lib directory of the Sparkflows install contains the fire core jar used in submitting the workflows to the Spark cluster. Set it correctly to be the absolute path of the fire core jar. |
| hdfs.namenodeURI | hdfs://localhost:8020 | Update the hdfs namenode URI. Set localhost to the machine on which the namenode is running. |
| hdfs.namenodeURI | file:/// | Set it to file:// when the files are on the local filesystem. This can be the case when HDFS is not there. |
| hdfs.namenodeURI | maprfs:/// | Set it to maprfs:/// for mapr. |
| hive.JDBC_DB_URL | jdbc:hive2://localhost:10000 | Update the hive JDBC DB URL if you would be accessing HIVE from Sparkflows. This is the URL of the HiveServer 2 server. |
| spark.sql-context | HIVEContext | Set it to either HIVEContext or SQLContext based on whether you want to use HIVEContext or SQLContext in your job. Use HIVEContext if you would be accessing the HIVE tables. |
| spark.master | yarn | Set it to yarn for connecting to a spark cluster running YARN |
| spark.master | spark://spark_master_hostname:port | Set it to the spark master URL when connecting to a spark cluster running in standalone mode. Port is normally 7077. |
| spark.spark-submit | spark-submit | Spark Submit command for submitting the Spark jobs to the cluster. It can be spark2-submit for Spark2 CDH clusters. Make sure to provide the full path or spark-submit should be in the path. |

### Create New Users in Fire

Fire allows creating multiple users. Create the users in Fire under `Administration/Users` who would be building and running workflows.

```
These users have to exist on HDFS. So ensure that these users Home Directory
are created on HDFS
```

Also create the home directory for the users on HDFS. The example code below creates the home directory for the user `xyz` onto HDFS. It also changes the permission of the directory.

- su - hdfs
- hadoop fs -mkdir /user/xyz
- hadoop fs -chown xyz:hadoop /user/xyz

### Setting up PySpark

If running with PySpark the following might need to be added to point PYSPARK to the right version of python on the cluster machines. Below is an example where python is at `/home/ec2-user/venv/bin/python`

It is also important that all the users are able to execute the python executable.

spark-env.sh:

```
export PYSPARK_PYTHON=/home/ec2-user/venv/bin/python
export PYSPARK_DRIVER_PYTHON=/home/ec2-user/venv/bin/python
```

spark-defaults.conf:

```
spark.yarn.appMasterEnv.PYSPARK_PYTHON=/home/ec2-user/venv/bin/python
spark.yarn.appMasterEnv.PYSPARK_DRIVER_PYTHON=/home/ec2-user/venv/bin/python
```

### 3.1.3 Customizing Fire Installation

Below are the details of Configuring Fire for various requirements:

#### Configuring Max Upload File Size

Fire allows users to upload files into HDFS through their Browser.

The settings which controls it is in `conf/application.properties`:

```
# max file size
multipart.maxFileSize: 10Mb
multipart.maxRequestSize: 10Mb
```

#### Increasing Memory of Fire Server

By default, when the Fire web server is started with `run-fire-server.sh`, it is given `1.5 GB of memory`.

Below is from run-fire-server.sh:

```
nohup ${JAVA} -server -Djava.ext.dirs=./user-lib/ -Xmx1548m -Xms1356m -
↪XX:+CMSClassUnloadingEnabled -XX:PermSize=512m -XX:MaxPermSize=512m -jar ./app/fire-
↪ui-1.3.0.war --spring.config.name=application,db,log4j --spring.config.
↪location=file:./conf/ &
```

- In order to increase the amount of memory for the Fire web server, `increase the value of -Xmx` based on the amount of memory available on your server.
- For example, you could raise it to 5 GB or 10 GB or more up to 25 GB.
  - -Xmx5g
  - -Xmx10g
  - -Xmx25g
- The increased memory size, if available, allows Fire to handle more requests and return results faster. Of course, when connected to an Apache Spark cluster, the full jobs are submitted to the Spark cluster through spark-submit, allowing it to be very scalable and not dependent on the Fire web server.
- The interactive execution of the workflows in the workflow editor, is run within Fire on a small subset of the data. These interactive executions would benefit from increased memory.

### 3.1.4 Configuring HTTPS for Fire Server

You can choose to run the Fire Server either on http or https connection.

The ports for http and https are configured in the file `conf/application.properties`:

```
http.port=8080
https.port=8443
```

#### HTTP

http://hostname:8080/login

#### HTTPS

https://hostname:8443/login

#### keystore.jks

Fire Server comes with a pre-configured keystore in the `conf` folder of the install.

- conf/keystore.jks
- conf/keystore.properties : Stores the keystore password

#### Generating New Keystore

You can use the following command for generating a new keystore:

```
keytool -genkeypair -alias sparkflows -keyalg RSA -validity 365 -keystore keystore.jks
```

You will be prompted with the following questions and enter something similar to the SAMPLE answers:

```
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  John Smith
What is the name of your organizational unit?
  [Unknown]:  BigData
What is the name of your organization?
  [Unknown]:  MyOrg
What is the name of your City or Locality?
  [Unknown]:  San Francisco
What is the name of your State or Province?
  [Unknown]:  California
What is the two-letter country code for this unit?
  [Unknown]:  CA
Is CN=John Smith, OU=BigData, O=MyOrg, L=San Francisco, ST=California, C=CA correct?
  [no]:  yes
Enter key password for <sparkflows>
        (RETURN if same as keystore password): Press the return key or Type and note
↪down the password
```

#### Copy the keystore into the Fire installation directory

- Copy the generated `keystore.jks` file into the `conf` folder of your installation.
- Update `keystore.properties` with the new password.

**Note:** When the keystore is updated, the password also has to be updated in case it changes.

The Fire web server would also have to be restarted for the changes to take effect.

#### Use keytool commands

#### Listing entries in Keystore

List entries in keystore:

```
keytool -list -keystore keystore.jks
```

#### Importing a Certificate to an existing Keystore

Importing a Certificate to an existing Keystore:

```
keytool -import -trustcacerts -alias <Name of Cert> -file <Absolute Path to .crt File>
↪ -keystore <Absolute Path to Desired Keystore> -storepass <KEYSTORE_PASSWORD>
```

### 3.1.5 HTTPS : Importing Self-Signed Certificates

Fire Insights comes with a self-signed certificate. It is contained in conf/keystore.jks.

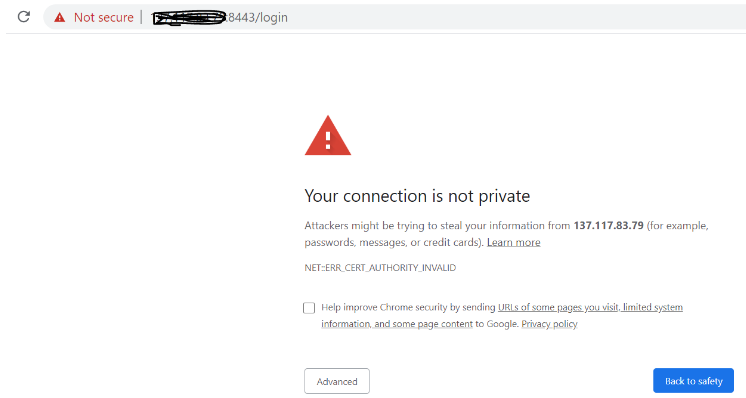
When using the self-signed certificate, the Browser will complain as it has not been issued by a Certificate Authority.

This warning message can be supressed by importing the self-signed certificate into the Browser inside `Trusted Root Certification Authorities`.

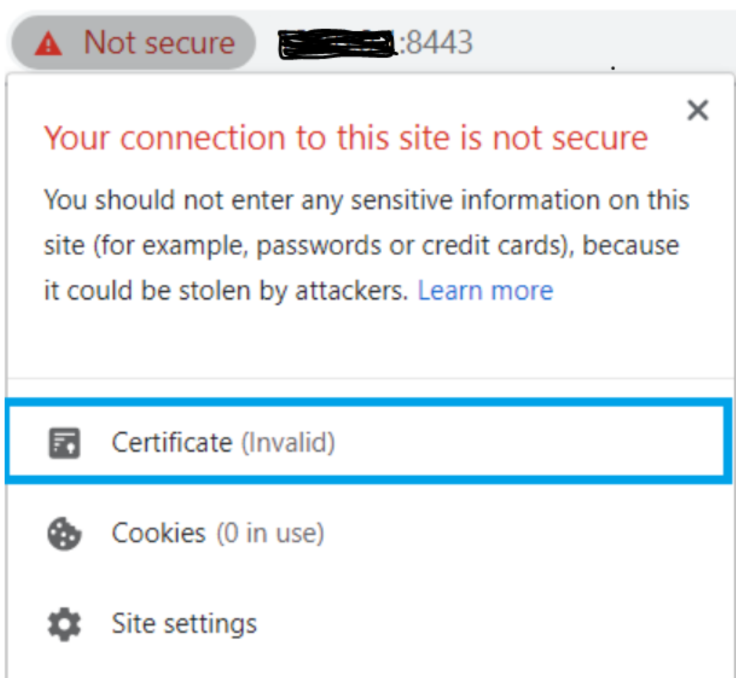
Below are the steps for importing self-signed certificate into your Browser.

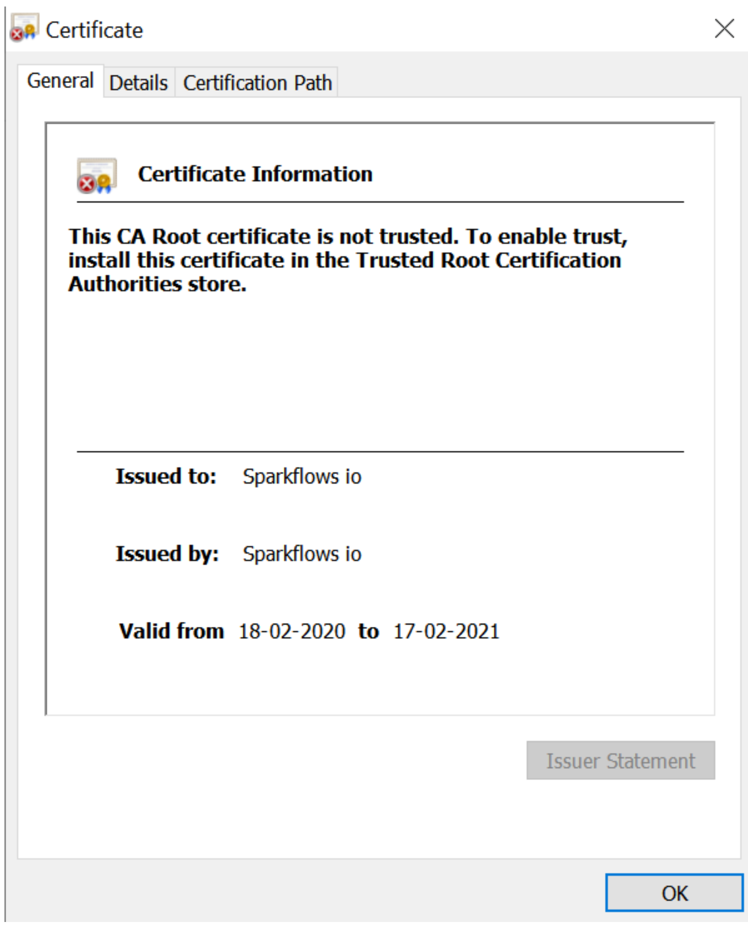
#### Export the certificate to your machine

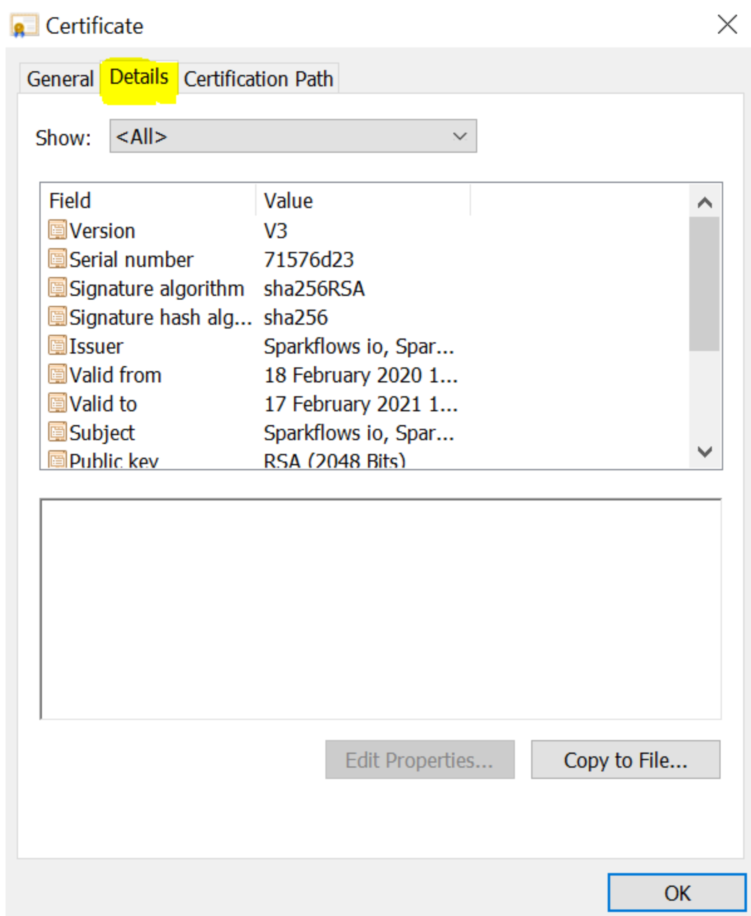
- **Got to `URL` for the `HTTPS` port.**
  - https://privateip:8443/login
- Click on `Not secure` option.
- Click on `Certificate`.
- View `Certificate`.
- Click on `Details` option to see detail information of certificate.
- Click on `copy to file` option to copy certificate to `local machine`.
- Select below option and press `Next`.
- Select the `Name & file location` of certificate.

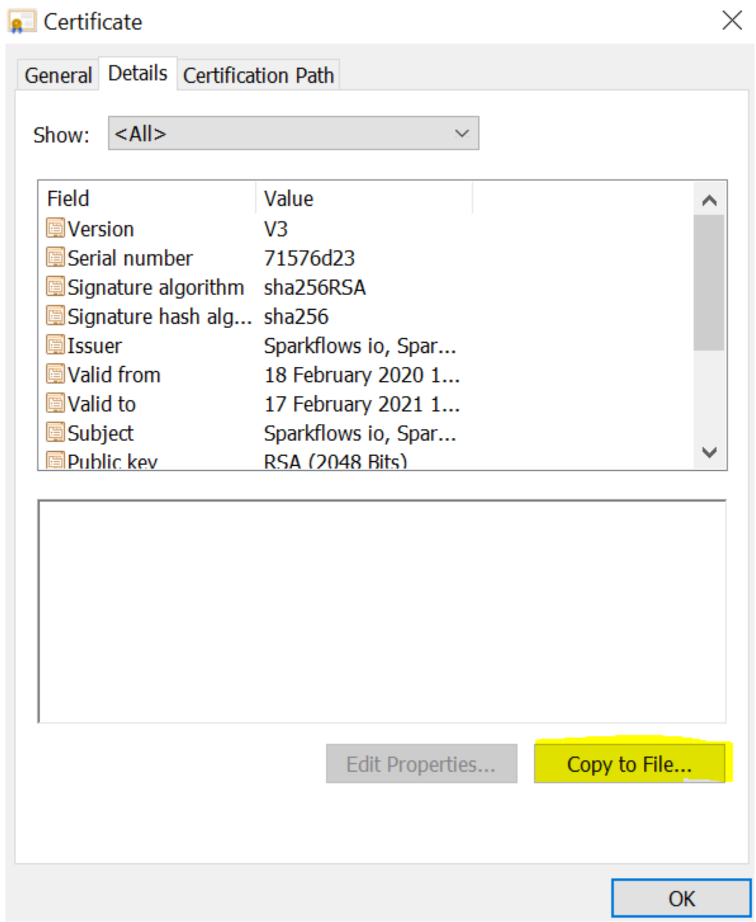

Not secure | [redacted]8443/login
Your connection is not private
Attackers might be trying to steal your information from 137.117.83.79 (for example, passwords, messages, or credit cards). Learn more
NET::ERR_CERT_AUTHORITY_INVALID
Help improve Chrome security by sending URLs of some pages you visit, limited system information, and some page content to Google. Privacy policy
Advanced
Back to safety

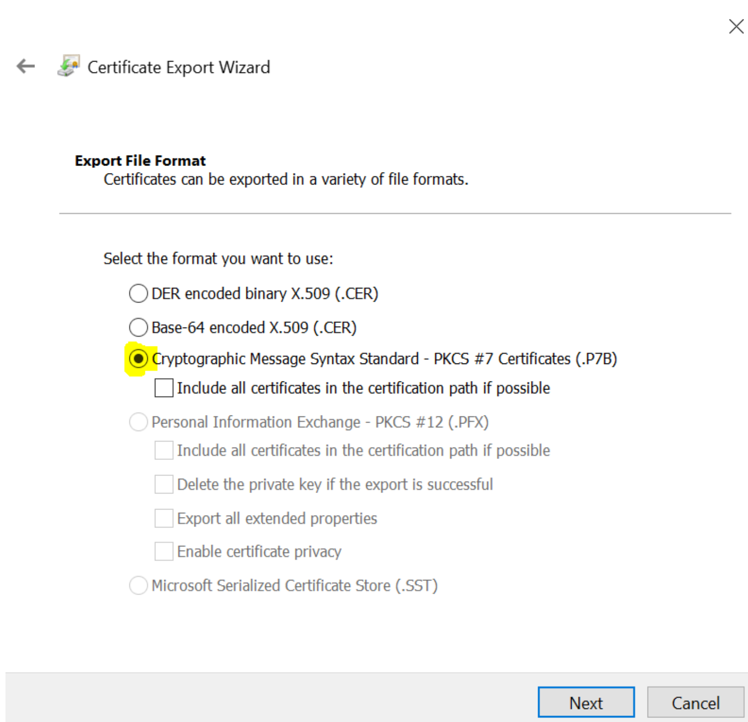

Not secure | [redacted]:8443/login

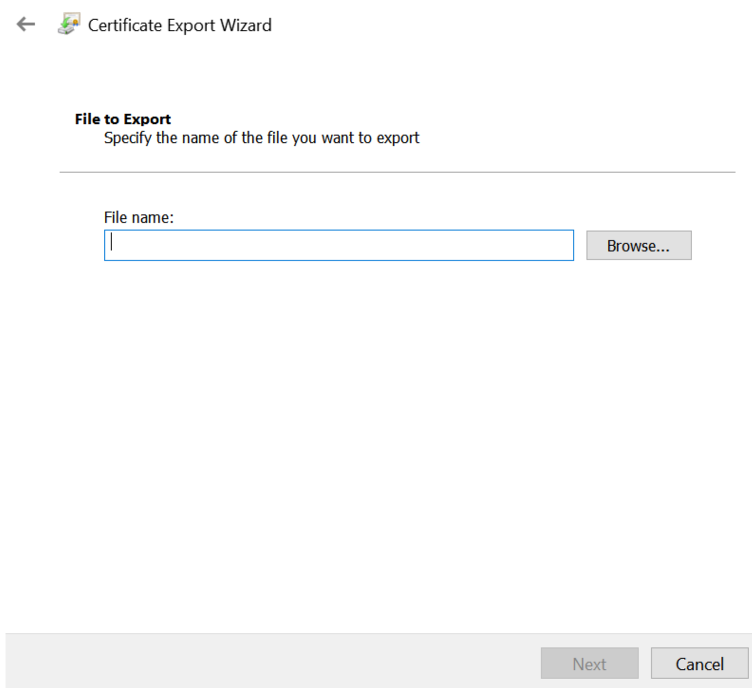

Not secure
[redacted]:8443
Your connection to this site is not secure
You should not enter any sensitive information on this site (for example, passwords or credit cards), because it could be stolen by attackers. Learn more
Certificate (Invalid)
Cookies (0 in use)
Site settings


Certificate
General
Details
Certification Path
Certificate Information
This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.
Issued to: Sparkflows io
Issued by: Sparkflows io
Valid from 18-02-2020 to 17-02-2021
Issuer Statement
OK

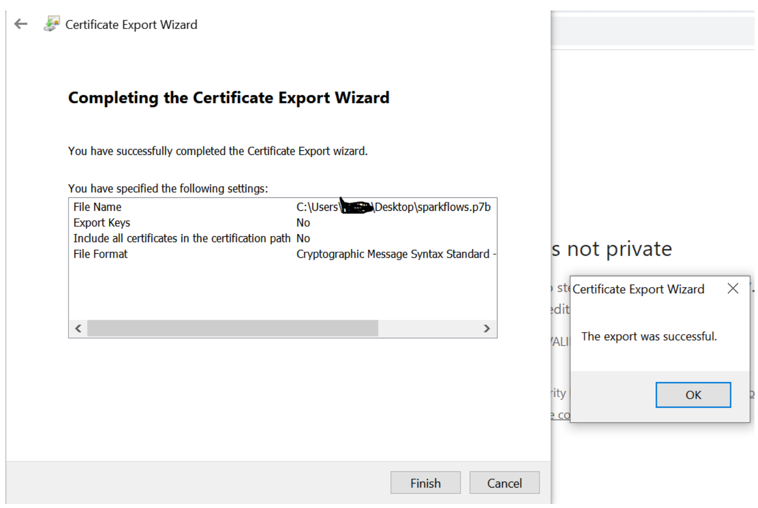

Certificate
General
Details
Certification Path
Show: <All>
Field
Value
Version
V3
Serial number
71576d23
Signature algorithm
sha256RSA
Signature hash alg...
sha256
Issuer
Sparkflows io, Spar...
Valid from
18 February 2020 1...
Valid to
17 February 2021 1...
Subject
Sparkflows io, Spar...
Edit Properties...
Copy to File...
OK


Certificate
General
Details
Certification Path
Show:
<All>
Field
Value
Version
V3
Serial number
71576d23
Signature algorithm
sha256RSA
Signature hash alg...
sha256
Issuer
Sparkflows io, Spar...
Valid from
18 February 2020 1...
Valid to
17 February 2021 1...
Subject
Sparkflows io, Spar...
Edit Properties...
Copy to File...
OK


Certificate Export Wizard
Export File Format
Certificates can be exported in a variety of file formats.
Select the format you want to use:
DER encoded binary X.509 (.CER)
Base-64 encoded X.509 (.CER)
Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
Include all certificates in the certification path if possible
Personal Information Exchange - PKCS #12 (.PFX)
Include all certificates in the certification path if possible
Delete the private key if the export is successful
Export all extended properties
Enable certificate privacy
Microsoft Serialized Certificate Store (.SST)
Next
Cancel



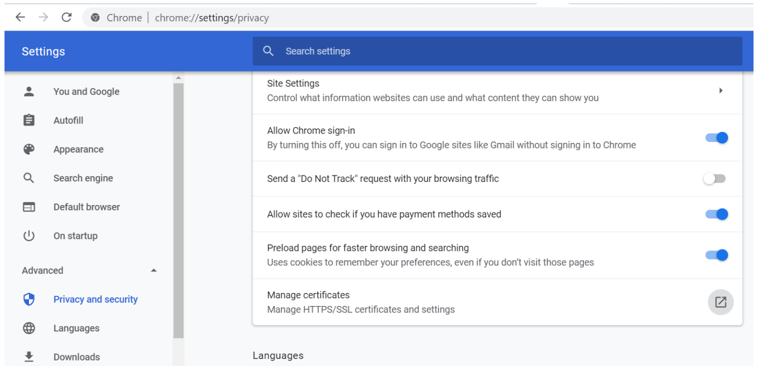

- After upadating the details `Success msg` will apear.

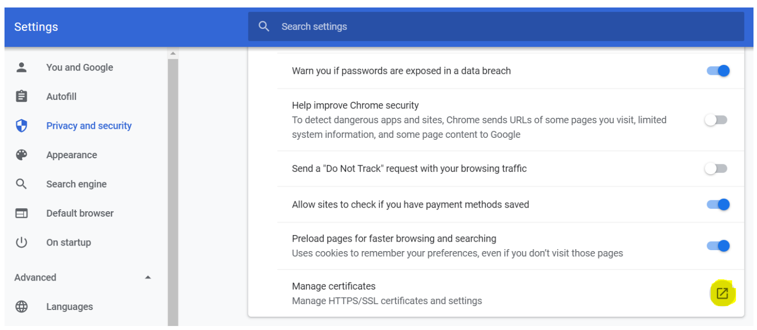


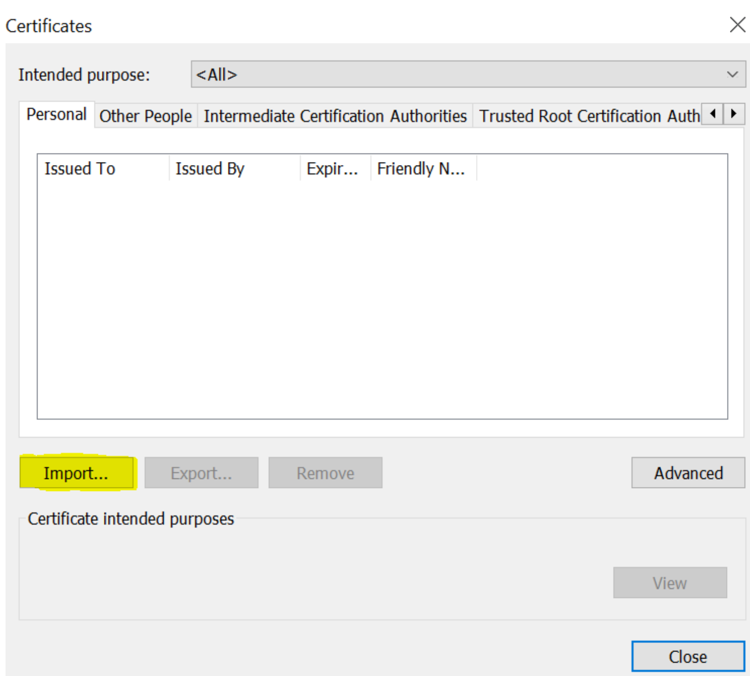

Next we need to add the exported certificate to the Browser.

## Add Certificate to Browser

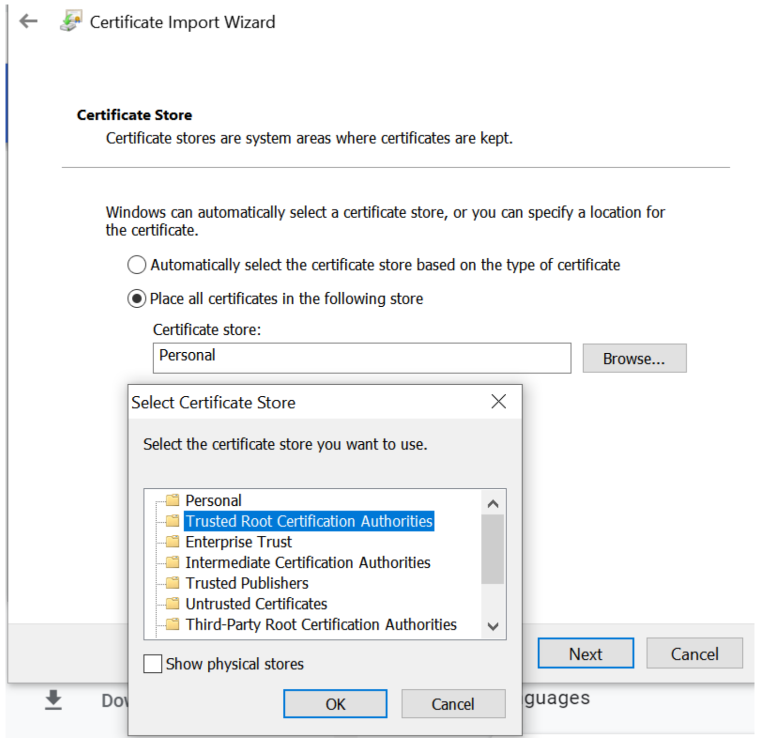
- Using Google chrome
- **Go to below location after opening `Google Chrome`.**
  - Settings -> Advanced -> Privacy and Security-> Manage Certificates
- Click on `Manage Certificate` icon.
- Click on `import`.


Chrome | chrome://settings/privacy
Settings
Search settings
You and Google
Autofill
Appearance
Search engine
Default browser
On startup
Advanced
Privacy and security
Languages
Downloads
Site Settings
Control what information websites can use and what content they can show you
Allow Chrome sign-in
By turning this off, you can sign in to Google sites like Gmail without signing in to Chrome
Send a "Do Not Track" request with your browsing traffic
Allow sites to check if you have payment methods saved
Preload pages for faster browsing and searching
Uses cookies to remember your preferences, even if you don't visit those pages
Manage certificates
Manage HTTPS/SSL certificates and settings
Languages


Settings
Search settings
You and Google
Autofill
Privacy and security
Appearance
Search engine
Default browser
On startup
Advanced
Languages
Warn you if passwords are exposed in a data breach
Help improve Chrome security
To detect dangerous apps and sites, Chrome sends URLs of some pages you visit, limited system information, and some page content to Google
Send a "Do Not Track" request with your browsing traffic
Allow sites to check if you have payment methods saved
Preload pages for faster browsing and searching
Uses cookies to remember your preferences, even if you don't visit those pages
Manage certificates
Manage HTTPS/SSL certificates and settings


Certificates
Intended purpose:
<All>
Personal
Other People
Intermediate Certification Authorities
Trusted Root Certification Auth
Issued To
Issued By
Expir...
Friendly N...
Import...
Export...
Remove
Advanced
Certificate intended purposes
View
Close

- Select `certificate from local system`, use `Trusted Root Certification Authorities` option and press `yes` to save it.




- Once the above process complete, close the `browser` and start again and try to login with above `URL`, It should work without any warnings.
- Help Url: https://peacocksoftware.com/blog/make-chrome-auto-accept-your-self-signed-certificate

### 3.1.6 Running on Another Port

There are 2 processes involved when running Fire.

- fire server
- fire

User's Browser talks with `fire server`, and in turn `fire server` talks with `fire`.

Both `fire server` and `fire` processes can be configured to listen on different ports.

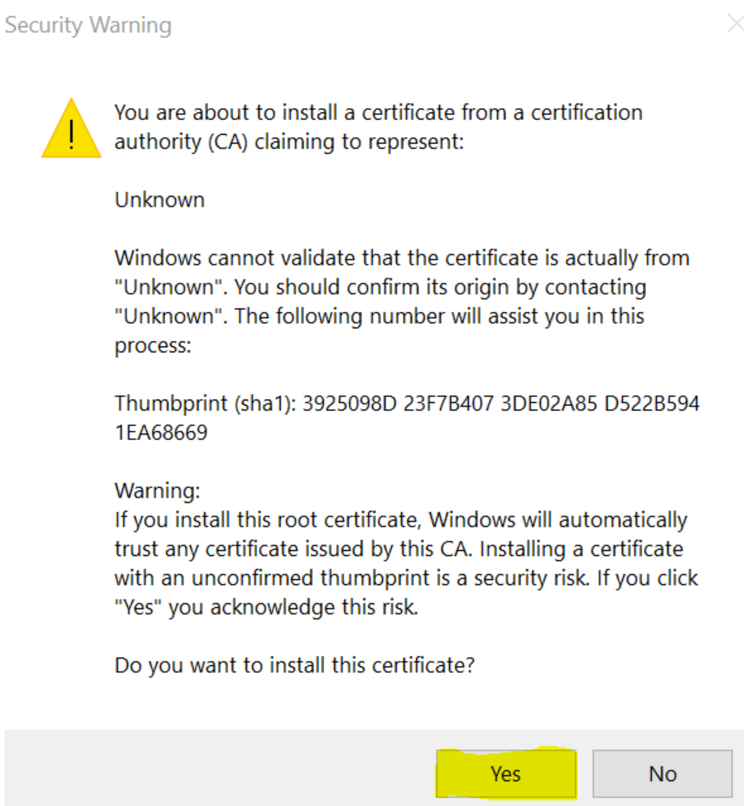
#### Running Fire Server on Another Port

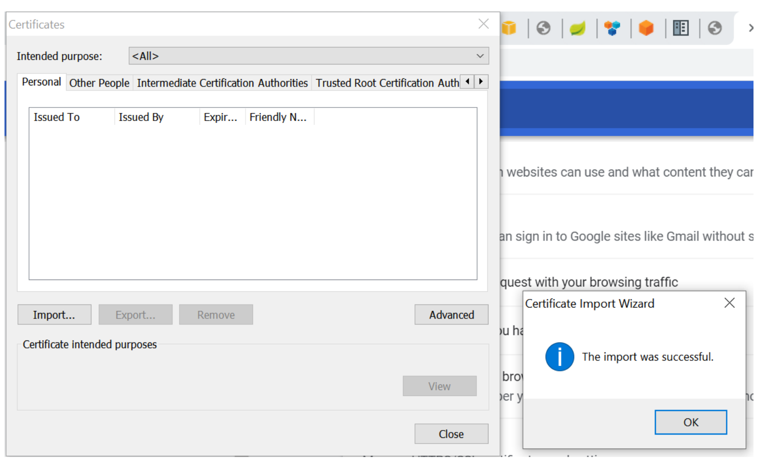
By default the fire server runs on the following ports:

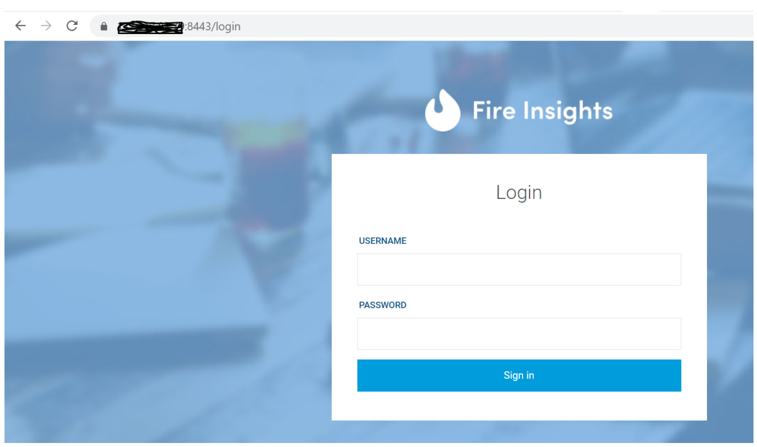
- 8080 (http)
- 8443 (https)

Below are the steps for running fire server on a different port.

- Navigate to the conf folder under Sparkflows install directory
- Open application.properties file:


Security Warning
You are about to install a certificate from a certification authority (CA) claiming to represent:
Unknown
Windows cannot validate that the certificate is actually from "Unknown". You should confirm its origin by contacting "Unknown". The following number will assist you in this process:
Thumbprint (sha1): 3925098D 23F7B407 3DE02A85 D522B594 1EA68669
Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.
Do you want to install this certificate?
Yes
No


Certificates
Intended purpose:
<All>
Personal
Other People
Intermediate Certification Authorities
Trusted Root Certification Auth
Issued To
Issued By
Expir...
Friendly N...
Import...
Export...
Remove
Advanced
Certificate intended purposes
View
Close
Certificate Import Wizard
The import was successful.
OK


[redacted]:8443/login
Fire Insights
Login
USERNAME
PASSWORD
Sign in

  - Configure http and https port numbers: Default 8080 for http and 8443 for https
  - http.port=8080
  - https.port=8443
- **In the Fire UI, under Administration/Configuration update the below property with the right port number.**
  - app.postMessageURL
- Restart Fire Server using one of the commands below depending on the environment (Unix/Linux or Windows) - run-fire-server.sh start - run-fire-server.bat

#### Running Fire on Another Port

Fire by default runs on port 8081.

In order to run Fire on a different port:

- **Navigate to the conf folder under Sparkflows install directory**
  - Open `application.properties` file:
  - Configure the http port
  - `fire.http.port=8081`
- Restart Fire using `./run-fire.sh start`

### 3.1.7 YARN Configurations

Fire can submit jobs to a YARN cluster. It can submit the spark jobs to run on YARN in either client or cluster mode.

#### Client Mode

For configuring to run in client mode, set the following parameter under **Administration/Configuration**:

```
spark.deploy-mode : client
```

In this mode, the spark driver runs on the same machine on which Fire is running. The workflow json file is written out to the directory **/tmp/fire/workflows** on the machine on which Fire is running.

#### Cluster Mode

For configuring to run in cluster mode, set the following parameter under **Administration/Configuration**:

```
spark.deploy-mode : cluster
```

In this mode, the spark driver runs on the spark cluster. The workflow json file is written out onto HDFS in the directory **.fireStaging** under the users HOME directory on HDFS.

The spark job reads the workflow json file from HDFS.

#### Impersonation

- Normally `app.impersonateUsers` is set to `true` so that the jobs are run as the logged in user.

---

**Note:** The logged in user into Fire should exist on HDFS

---

### 3.1.8 Configuring HTTPS for Fire

Fire server can listen on HTTPS. Fire Server comes with a pre-configured keystore.

Below are the steps for configuring Fire with your keystore and certificates.

#### Generate a Keystore

You can use the following command for generating the Keystore:

```
keytool -genkey -keyalg RSA -alias sparkflows -keystore keystore.jks -validity 365 -
↪keysize 2048 -ext san=ip:< host machine ip address>
```

You will be prompted with the following questions and enter something similar to the SAMPLE answers:

```
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  John Smith
What is the name of your organizational unit?
  [Unknown]:  BigData
What is the name of your organization?
  [Unknown]:  MyOrg
What is the name of your City or Locality?
  [Unknown]:  San Francisco
What is the name of your State or Province?
  [Unknown]:  California
What is the two-letter country code for this unit?
  [Unknown]:  CA
Is CN=John Smith, OU=BigData, O=MyOrg, L=San Francisco, ST=California, C=CA correct?
  [no]:  yes
Enter key password for <sparkflows>
        (RETURN if same as keystore password): Press the return key or Type and note␣
↪down the password
```

#### Copy the keystore into the Fire installation directory

Copy the generated `keystore.jks` file into the `conf` folder of your installation.

#### Update the keystore password

Update keystore.properties to udpdate the password of the new keystore.jks file:

```
keystore.password=123456
```

#### Port Number

Fire by default listens on port 8443 for https.

This is configured in the file `conf/application.properties`:

```
#Configure http and https port numbers : Default 8080 for http and 8443 for https
http.port=8080
https.port=8443
```

#### Finally restart the Fire Server

Restart the Fire server for the changes to take effect:

```
./run-fire-server.sh stop
./run-fire-server.sh start
```

### 3.1.9 Configuring Kerberos

Fire runs with a kerberized Spark cluster.

#### Steps for configuring Kerberos on Fire

- **Generate a keytab for Fire**
- **Place it in . . . /fire-x.y.z/conf directory**:

  ```
  While this is the recommended location, the keytab file can be placed in any␣
  ↪another directory too.
  ```

- **Make sure only the user running fire application has access to the keytab**. For example:

  ```
  -r-------- 1 fire staff 436 Jun 29 16:06 hive.keytab
  ```

- **Go to Administration/Configuration and update the following configurations to enable Kerberos for Fire**

| Configuration | Example Value | Details |
|---|---|---|
| kerberos.enabled | true | Set it to true to enable Kerberos for Fire |
| kerberos.keytab | /user/ec2-user/fire.keytab | Absolute path of the keytab generated for Fire |
| kerberos.principal | fire@EXAMPLE.COM | Kerberos Principal of the keytab of Fire |
| kerberos.KERBEROS_REALM | EXAMPLE.COM | Kerberos Realm |
| kerberos.KERBEROS_KDC | hostname.example.com | KDC Server |
| kerberos.hiveServer2Principal | hive/hive2_host@EXAMPLE.COM | HIVE Server2 Principal |

#### Steps for generating the keytab for Fire

Below are the steps for generating the keytab file. **We have chosen fire as the principal name. But you can have it as any user you are running Fire with**.

- **Start kadmin.local and add the new principal** `fire@EXAMPLE.COM`:

```
$ kadmin.local

kadmin.local: addprinc -randkey fire@EXAMPLE.COM

WARNING: no policy specified for fire@EXAMPLE.COM; defaulting to no policy
Principal "fire@EXAMPLE.COM" created.
```

- **Create fire keytab file**:

```
kadmin.local: xst -norandkey -k fire.keytab fire@EXAMPLE.COM

Entry for principal fire@EXAMPLE.COM with kvno 1, encryption type aes256-cts-hmac-
↪sha1-96 added to keytab

WRFILE:fire.keytab.

Entry for principal fire@EXAMPLE.COM with kvno 1, encryption type aes128-cts-hmac-
↪sha1-96 added to keytab

WRFILE:fire.keytab.

Entry for principal fire@EXAMPLE.COM with kvno 1, encryption type des3-cbc-sha1␣
↪added to keytab     WRFILE:fire.keytab.

Entry for principal fire@EXAMPLE.COM with kvno 1, encryption type arcfour-hmac␣
↪added to keytab WRFILE:fire.keytab.

Entry for principal fire@EXAMPLE.COM with kvno 1, encryption type des-hmac-sha1␣
↪added to keytab WRFILE:fire.keytab.

Entry for principal fire@EXAMPLE.COM with kvno 1, encryption type des-cbc-md5␣
↪added to keytab WRFILE:fire.keytab.
```

- **Exit kadmin.local**:

```
kadmin.local: exit
```

### Verifying that the keytab file was correctly created

Below are the steps for verifying the keytab file.

- **Ensure that the keytab file was created and it has the right permissions**:

```
$ ls -l fire.keytab

-rw------- 1 root root 382 Jul 24 17:55 fire.keytab
```

- **Further verify the contents of keytab file. A normal keytab file depending on your krb5.conf settings, looks like this**:

```
$ klist -e -k -t fire.keytab

Keytab name: FILE:fire.keytab

KVNO Timestamp Principal
.................................................................................
↪.................................................................................
```

(continues on next page)

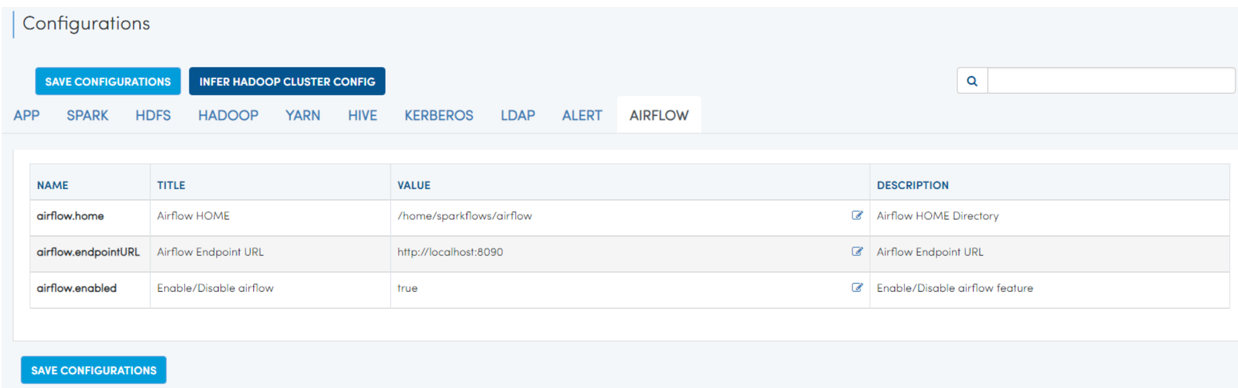
(continued from previous page)

```
1 07/24/16 17:55:07 fire@EXAMPLE.COM (aes256-cts-hmac-sha1-96)

1 07/24/16 17:55:08 fire@EXAMPLE.COM (aes128-cts-hmac-sha1-96)

1 07/24/16 17:55:08 fire@EXAMPLE.COM (des3-cbc-sha1)

1 07/24/16 17:55:08 fire@EXAMPLE.COM (arcfour-hmac)

1 07/24/16 17:55:08 fire@EXAMPLE.COM (des-hmac-sha1)

1 07/24/16 17:55:08 fire@EXAMPLE.COM (des-cbc-md5)
```

### 3.1.10 Configuring Pipelines

Fire uses Apache Airflow for executing Pipelines. Hence Airflow has to be installed on the same machine as Fire.

Below are the configurations needed in Fire for Airflow.



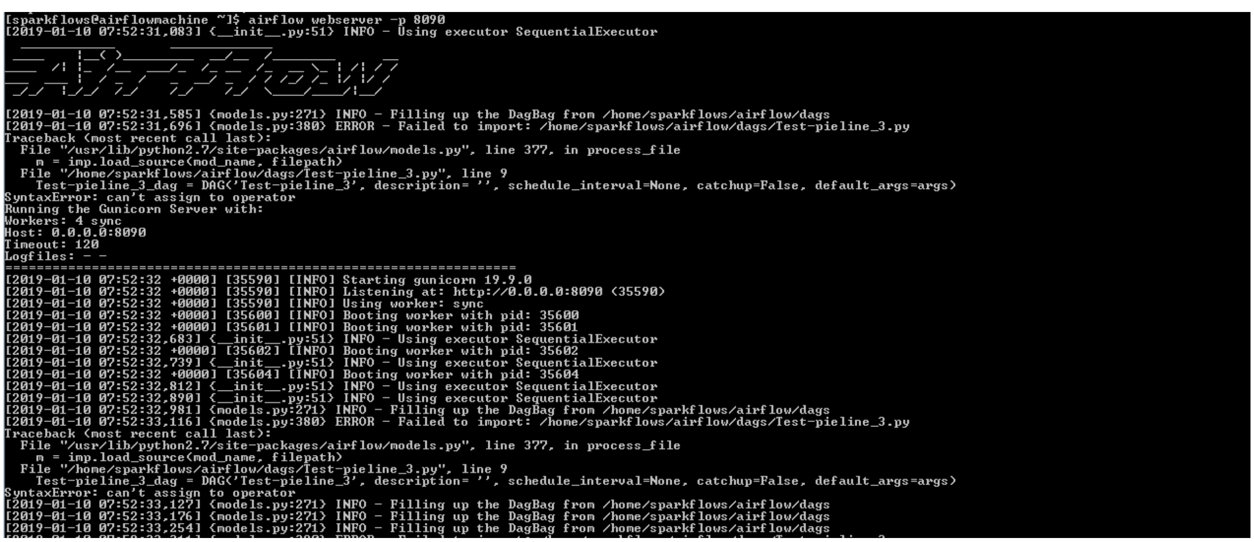

#### Airflow Installation

It explain the steps involved in installing Airflow on Centos and RHEL. Detailed Airflow Install Instructions is at: https://airflow.apache.org/installation.html

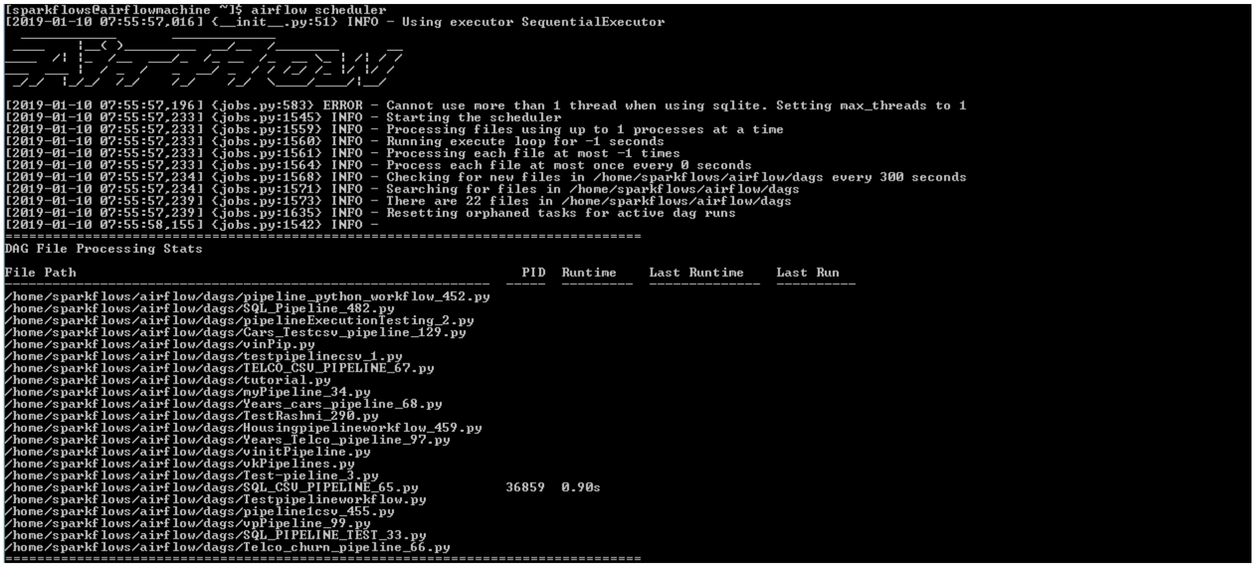
- Login to machine
- Before installing airflow update installed package:
- yum -y update
- Install python-pip and any required packages:
- sudo yum install epel-release
- sudo yum install python-pip
- Check the version of pip that is installed and if reqd upgrade:
- pip -V
- pip install –upgrade setuptools
- Note that for 1.10 you now need to preface install commands or export this env var:
- export SLUGIFY_USES_TEXT_UNIDECODE=yes

- Install gcc , gcc-c++ and dependencies for python 2.7
- sudo yum -y install gcc gcc-c++ kernel-devel
- sudo yum -y install python-devel libxslt-devel libffi-devel openssl-devel
- Airflow needs a home, ~/airflow is the default
- export AIRFLOW_HOME=~/airflow
- Install from pypi using pip
- pip install apache-airflow
- To check airflow version
- airflow version

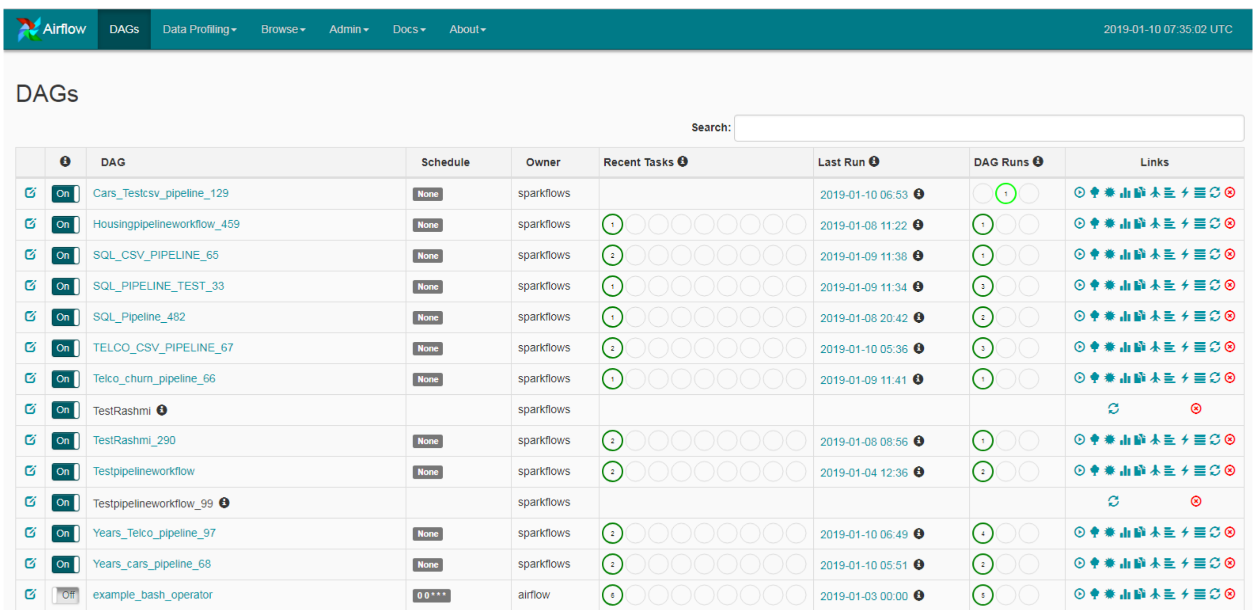

- Generate a Fernet key for Airflow(optional)
- python -c “from cryptography.fernet import Fernet; print(Fernet.generate_key().decode())”
- fgrc0MPUG1n3Q352Fp705A-bysNHX6EFRr7nYFTmXXA=
- update in airflow.cfa
- fernet key: fgrc0MPUG1n3Q352Fp705A-bysNHX6EFRr7nYFTmXXA=
- Initialize the Airflow database
- airflow initdb
- Start the web server, its default port is 8080, If any other application is running on 8080, we can update other port for airflow
- airflow webserver -p 8090




- Start the scheduler




- airflow scheduler
- Login in browser
- http://x.y.z.w:8090




## 3.1.11 Different Default Values on Startup

### Overview

Fire has a number of properties under Administration / Configuration. When initially installed they have certain default values. Administrators can log into Fire through their Browser and update the Properties.

However, there might be cases where you want Fire to come up with different default values for the Configurations when installed. This enables more automation and the Administrator does not have to go in and manually change the default values.

#### Steps

Below are the Steps to override the default Configuration values:

- Update the file **conf/configuration.properties** with the new key/value pairs

Now the default values are populated with the values provided in `configuration.properties`.

Fire comes with an empty `conf/configuration.properties` file. You can put in your values into it.

#### Remove properties from conf/configuration.properties

Fire will continue to take the final values from `conf/configuration.properties` for any property which is there in the file.

If you would like Fire not to use any of the properties from `conf/configuration.properties`, but take it from the database, then remove or comment out those properties in `conf/configuration.properties`.

#### Saving the new values into the DB

When the configuration values are saved, they get updated in the database.

Even if they are removed from `configuration.properties`, they would have been saved in the database.

### 3.1.12 Configuring LDAP/OAuth Authentication

Fire Insights supports various types of authencations:

- Database Authentication
- LDAP Authentication
- OAuth Authentication

#### Database Authentication

Fire Insights can authenticate the user against its own database.

User's password are stored encrypted.

This is the default authentication mechanism of Fire Insights. Users created in Fire are stored in the database.

#### LDAP Authentication

Fire Insights can be configured to authenticate the user against LDAP. Users have to be added to Fire, before they can log into Fire and start using it.

The following configurations have to be set appropriately.

configuration/authentication/../_assets/installation/ldap

### LDAP Parameters

Table 1: LDAP Parameters

| Name of Parameter | Description | Example |
|---|---|---|
| ldap.Order | Order in which to authenticate the user. Possible values are DB, LDAP_DB, DB_LDAP. | |
| ldap.URL | The URL of the LDAP server. The URL must be prefixed with ldap:// or ldaps://. The URL can optionally specify a custom port, for example: ldaps://ldap_server.example.com:1636. | ldap://localhost:10389 |
| ldap.Base | The distinguished name to use as a search base for finding users and groups. This should be similar to 'dc=sparkflows,dc=com'. | dc=sparkflows,dc=com |
| ldap.UserDn | Distinguished name of the user to bind as. This is used to connect to LDAP/AD for searching user and group information. This may be left blank if the LDAP server supports anonymous binds. | uid=john,ou=development,dc=sparkflows,dc=co |
| ldap.Password | The password of the bind user. | xyz |
| ldap.UserSearchBase | User Search Base | ou=development |
| ldap.UserSearchFilter | The base filter for searching for users. For Active Directory, this is typically '(objectClass=user)'. | For Active Directory : (objectClass=user) Other Example : (uid={0}) |
| ldap.GroupSearchBase | Group Search Base | ou=groups |
| ldap.GroupSearchFilter | Group Search Filter | For Active Directory : (objectClass=group) Other Example : (member={0}) |

### Note

For `ldap.UserSearchFilter` we can use strings like `(uid={USERNAME})` In this case {USERNAME} would be replaced by the real username of the user when searching in LDAP during `Add User`.

### LDAP Certificate

If `ldaps` is being used, the ldap certificate needs to be imported into cacerts.

For Reference : https://docs.oracle.com/cd/E19509-01/820-3399/ggfrj/index.html

### Importing a user from LDAP into Sparkflows

Once LDAP is enabled in Sparkflows, users can be imported into Sparkflows from LDAP.

- Go to Administration/User
- Click on Add/Sync User
- Enter the username and click on Search
- User details are fetched from LDAP
- Click on Add User to create the user in Sparkflows

### User Login

Once LDAP is enabled in Sparkflows, all the authentication for login in Sparkflows are done against LDAP.

### Search Order

Sparkflows would search in LDAP and then in its DB. Search order is determined by the parameter ldap.Order.

If it is set to `LDAP_DB`, it would first search for the User in LDAP and then in its own DB. This allows having the admin user in the Sparkflows DB if needed, so that all users are not locked out of the system in case LDAP goes down or ends up with invalid Configurations.

### Reference

Below are some great links for reference:

- Active Directory Search Filter Syntax : https://msdn.microsoft.com/en-us/library/aa746475(v=vs.85).aspx

### What if I get locked out

`ldap.Order` determines the order in which Sparkflows tries to log in the user. In case you are locked out of Sparkflows and are not able to log in, you can do the following:

- Add the below line to conf/configuration.properties:

```
ldap.Order=DB
```

- Then restart the fire server. Now you should be able to log in with your admin account.

Once things are back to normal, you can remove the line you added to `configuration.properties` and restart the fire server.

### Notes

- Search strings are not case sensitive

### OAuth Authentication

Fire Insights supports OAuth Authentication.

#### Create Users in Fire

First create the user in Fire under `Administration/Users`.

Log into Fire with the `admin` user in order to be able to create the New Users.

#### Configuring OAuth

In order the configure OAuth in Fire Insights, add the OAuth configuration parameters to `conf/application.properties`.

Below is an example of configuring OAuth in Fire with Okta.

```
# Okta settings
oauth.client.clientId: 0oadvfdsfsdA7Y68356
oauth.client.clientSecret: YSWFdZf9kfdsfsdfsdfsdnI0SVrswOJpHl
oauth.client.accessTokenUri: https://xyz.okta.com/oauth2/default/v1/token
oauth.client.userAuthorizationUri: https://xyz.okta.com/oauth2/default/v1/authorize
oauth.client.clientAuthenticationScheme: form
oauth.client.scope: openid profile email
oauth.resource.userInfoUri: https://xyz.okta.com/oauth2/default/v1/userinfo
```

#### Fire OAuth URL

In order to log in the user into Fire using OAuth, use the following URL:

- http://machine_name:port/login/oauth

This URL will take the user to the OAuth login page. After the user logs in there, the user is redirected back to Fire and is logged in.

If the user is already logged in, going to the above URL, automatically brings up the Fire page for the user.

### 3.1.13 HDInsight Integration

Fire Insights runs seamlessly on Azure HDInsight.

Fire can be installed on the master or edge nodes of the cluster.

#### HDInsights and Ports

https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-hadoop-port-settings-for-services

Linux-based HDInsight clusters only expose three ports publicly on the internet; 22, 23, and 443. These ports are used to securely access the cluster using SSH and services exposed over the secure HTTPS protocol.

Internally, HDInsight is implemented by several Azure Virtual Machines (the nodes within the cluster) running on an Azure Virtual Network. From within the virtual network, you can access ports not exposed over the internet. For example, if you connect to one of the head nodes using SSH, from the head node you can then directly access services running on the cluster nodes.

To join additional machines to the virtual network, you must create the virtual network first, and then specify it when creating your HDInsight cluster. For more information, see Extend HDInsight capabilities by using an Azure Virtual Network

https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-extend-hadoop-virtual-network

#### Port Configuration

Fire Insights by default listens on ports 8080 and 8443.

On HDInsight, port 8080 generally is already in use. So configure Fire Insights to listen on another port, say 8090.

Edit conf/application.properties:

```
#Configure http and https port numbers : Default 8080 for http and 8443 for https
http.port=8090
https.port=8443
```

#### Open the Port for access

Now the port 8090 needs to be opened to be accessed by the users using their Browser.

- https://stackoverflow.com/questions/45239566/accessing-http-on-custom-port-in-azure-hdinsight-cluster

#### Add proxy user

Fire needs to impersonate the logged in user.

In Ambari for the HDInsight cluster, add the Fire user in HDFS to be the proxy user.

Suppose Fire is installed as the user `fire`. Add the below to HDFS/Configuration in Ambari:

```
hadoop.proxyuser.fire.groups=*
hadoop.proxyuser.fire.hosts=*
```

#### Connecting Fire Insights to the HDInsight Cluster

In Fire Insights, under Administration/Configuration, configure the following for it to be able to connect to the HDInsight cluster.

- hdfs.namenodeURI=wasb://
- app.runOnCluster=true
- app.postMessageURL=
- app.sparkSubmitJar=

Clicking on `Infer Hadoop Configuration` would correctly infer these. Hit `Save` after that.

### 3.1.14 MapR Integration

This document describes details when installing Fire Insights on a MapR cluster.

#### Download Fire Insights

- Download MapR specific binary from : https://www.sparkflows.io/archives

#### Turn off Impersonation

- In Administration / Configuration of Sparkflows:

```
Turn off impersonation : Set app.impersonateUsers = false
Set maprfs : hdfs.namenodeURI = maprfs:///
Set spark-submit appropriately : spark.spark-submit = /opt/mapr/spark/xyz/bin/
↪spark-submit
```

#### Update http port

- Set `http port` to be different in `conf/application.properties` if there are other processes using the specified ports

#### Fire User

- Fire has to be installed as a user which can submit jobs to the MapR cluster. Say we installed Fire as user `mapr`:

```
Create a mapr user in sparkflows and log in as that user
Start using Sparkflows
```

### 3.1.15 Upgrading Fire

#### Stop Fire if it is running

Stop Fire with the below command from the directory in which it is installed:

```
run-fire-server.sh stop
```

#### Download the new fire tgz file

Download Fire tgz file from:

```
- https://www.sparkflows.io/download  OR

- https://www.sparkflows.io/archives
```

#### Unpack it

Unpack the tgz file with below on unix/linux:

```
tar xvf fire-x.y.z.tgz
```

#### Upgrade the H2 or MySQL database

- If you have updated the `conf/db.properties` file, copy it from your old location to the new directory
- Backup your existing H2 db files. By default they are in your home directory as `firedb.mv.db`
- If you are using MySQL, backup the fire database in MySQL.

- Execute the following commands on the Command Line to upgrade the Fire database:

```
cd <install_dir>/fire-x.y.z

./create-h2-db.sh       OR      ./create-mysql-db.sh
```

```
the above command creates or updates the existing db if one already exists
```

#### Restart Fire

Restart the Fire Server:

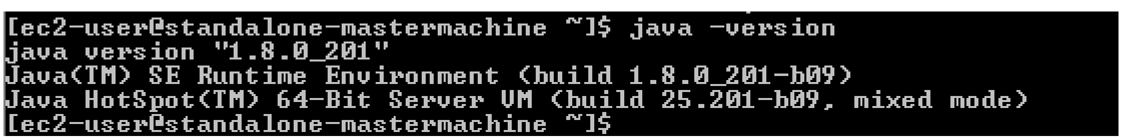
```
run-fire-server.sh start
```

### 3.1.16 Running Apache Spark Standalone

Fire can be run on Spark Standalone cluster. In this case, Hadoop does not need to be installed.

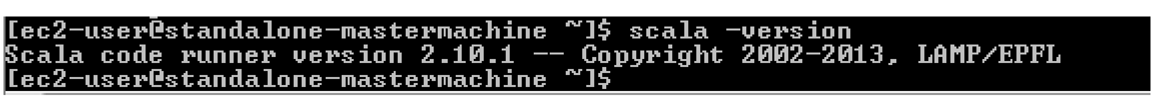
#### Installing Spark Standalone

- Install Java
  - wget –no-cookies –no-check-certificate –header "Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie" "https://download.oracle.com/otn-pub/java/jdk/8u201-b09/42970487e3af4f5aa5bca3f542482c60/jdk-8u201-linux-x64.rpm"
  - yum localinstall jdk-8u201-linux-x64.rpm
  - Java -version



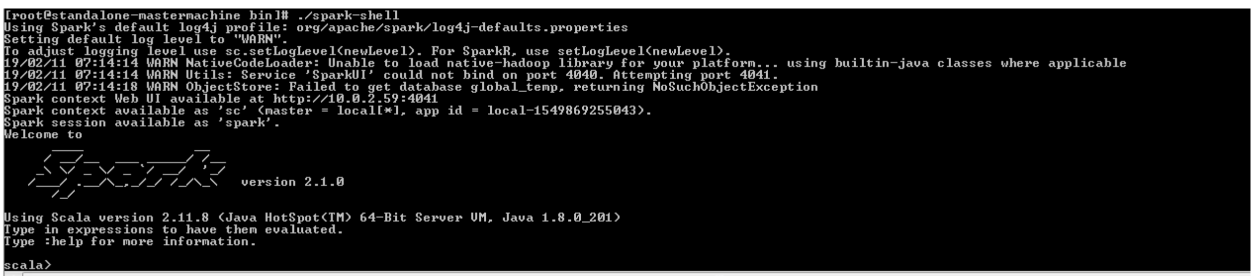

#### Install Scala

- Install Scala
  - wget http://www.scala-lang.org/files/archive/scala-2.10.1.tgz
  - tar xvf scala-2.10.1.tgz
  - sudo mv scala-2.10.1 /usr/lib
  - sudo ln -s /usr/lib/scala-2.10.1 /usr/lib/scala
  - export PATH=$PATH:/usr/lib/scala/bin ( we can add in .bash_profile)
  - scala -version



### Install Apache Spark

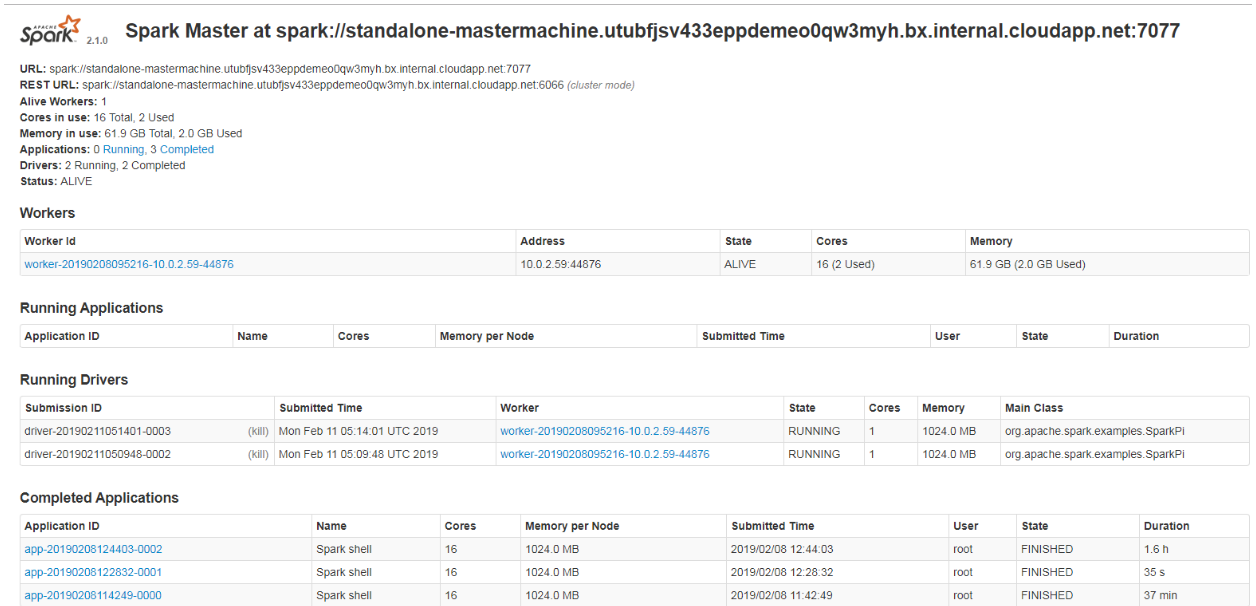
- Download Spark
  - wget http://d3kbcqa49mib13.cloudfront.net/spark-2.1.0-bin-hadoop2.7.tgz
- Extract, create a new directory under the /usr/local called spark and copy the extracted connect into it
  - tar xf spark-2.1.0-bin-hadoop2.7.tgz
  - mkdir /usr/local/spark
  - cp -r spark-2.1.0-bin-hadoop2.7/* /usr/local/spark
- Setup some Environment variables before you start spark-shell ( in .bash_profile)
  - export SPARK_EXAMPLES_JAR=/usr/local/spark/examples/jars/spark-examples_2.11-2.0.0.jar
  - PATH=$PATH:$HOME/bin:/usr/local/spark/bin
- Start you Scala Shell and run Spark
  - Go to sparkflows home directory
  - cd /usr/local/spark/bin
  - ./spark-shell




- Start a standalone master server by executing:
  - ./sbin/start-master.sh ( from spark home directory)
- Once started, the master will print out a spark://HOST:PORT URL
- You can also find this URL on the master's web UI,
  - http://Master_host_ip:8080/ by default

### Setup Spark Slave(Worker) Node

- Go to SPARK_HOME/conf/ directory.
- Edit the file spark-env.sh – Set SPARK_MASTER_HOST
  - If spark-env.sh is not present, spark-env.sh.template would be present. Make a copy of spark-env.sh.template with name spark-env.sh and add/edit the field SPARK_MASTER_HOST. Part of the file with SPARK_MASTER_HOST
  - cp ./conf/spark-env.sh.template ./conf/spark-env.sh
- Add a line in spark-env.sh :
  - SPARK_MASTER_HOST='MASTER_HOST_IP'



Spark 2.1.0 Spark Master at spark://standalone-mastermachine.utubfjsv433eppdemeo0qw3myh.bx.internal.cloudapp.net:7077

**URL:** spark://standalone-mastermachine.utubfjsv433eppdemeo0qw3myh.bx.internal.cloudapp.net:7077
**REST URL:** spark://standalone-mastermachine.utubfjsv433eppdemeo0qw3myh.bx.internal.cloudapp.net:6066 *(cluster mode)*
**Alive Workers:** 1
**Cores in use:** 16 Total, 2 Used
**Memory in use:** 61.9 GB Total, 2.0 GB Used
**Applications:** 0 Running, 3 Completed
**Drivers:** 2 Running, 2 Completed
**Status:** ALIVE

Workers

| Worker Id | Address | State | Cores | Memory |
|---|---|---|---|---|
| worker-20190208095216-10.0.2.59-44876 | 10.0.2.59:44876 | ALIVE | 16 (2 Used) | 61.9 GB (2.0 GB Used) |

Running Applications

| Application ID | Name | Cores | Memory per Node | Submitted Time | User | State | Duration |
|---|---|---|---|---|---|---|---|

Running Drivers

| Submission ID | Submitted Time | Worker | State | Cores | Memory | Main Class |
|---|---|---|---|---|---|---|
| driver-20190211051401-0003 (kill) | Mon Feb 11 05:14:01 UTC 2019 | worker-20190208095216-10.0.2.59-44876 | RUNNING | 1 | 1024.0 MB | org.apache.spark.examples.SparkPi |
| driver-20190211050948-0002 (kill) | Mon Feb 11 05:09:48 UTC 2019 | worker-20190208095216-10.0.2.59-44876 | RUNNING | 1 | 1024.0 MB | org.apache.spark.examples.SparkPi |

Completed Applications

| Application ID | Name | Cores | Memory per Node | Submitted Time | User | State | Duration |
|---|---|---|---|---|---|---|---|
| app-20190208124403-0002 | Spark shell | 16 | 1024.0 MB | 2019/02/08 12:44:03 | root | FINISHED | 1.6 h |
| app-20190208122832-0001 | Spark shell | 16 | 1024.0 MB | 2019/02/08 12:28:32 | root | FINISHED | 35 s |
| app-20190208114249-0000 | Spark shell | 16 | 1024.0 MB | 2019/02/08 11:42:49 | root | FINISHED | 37 min |

## Start spark as slave

- Goto SPARK_HOME/sbin and execute the following command.
  - ./start-slave.sh spark://MASTER_HOST_IP:7077

## Installing Fire

Install Fire on the master node.

- Download Fire Jar from website
  - wget https://s3.amazonaws.com/sparkflows-release/fire/rel-x.y.z/2/fire-x.y.z.tgz
  - tar xvf fire-x.y.z.tgz
- Go to below directory:
  - cd fire-x.y.z
  - Update the port of Fire-ui & Fire to 8090 & 8082 as default port 8080 & 8081 is used by standalone spark, we can chose any other also.
  - From fire-x.y.z directory, we need to go conf/application.properties and update the port No.

```
#Configure http and https port numbers : Default 8080 for http and 8443 for https
http.port=8090
https.port=8443

# Configure http ports for fire
fire.http.ports=8082

spring.jackson.deserialization.FAIL_ON_UNKNOWN_PROPERTIES=false
```

- Create database & run fire & fire-ui server
  - ./create-h2-db.sh
  - ./run-fire.sh start

- ./run-fire-server.sh start

### Configuring Fire

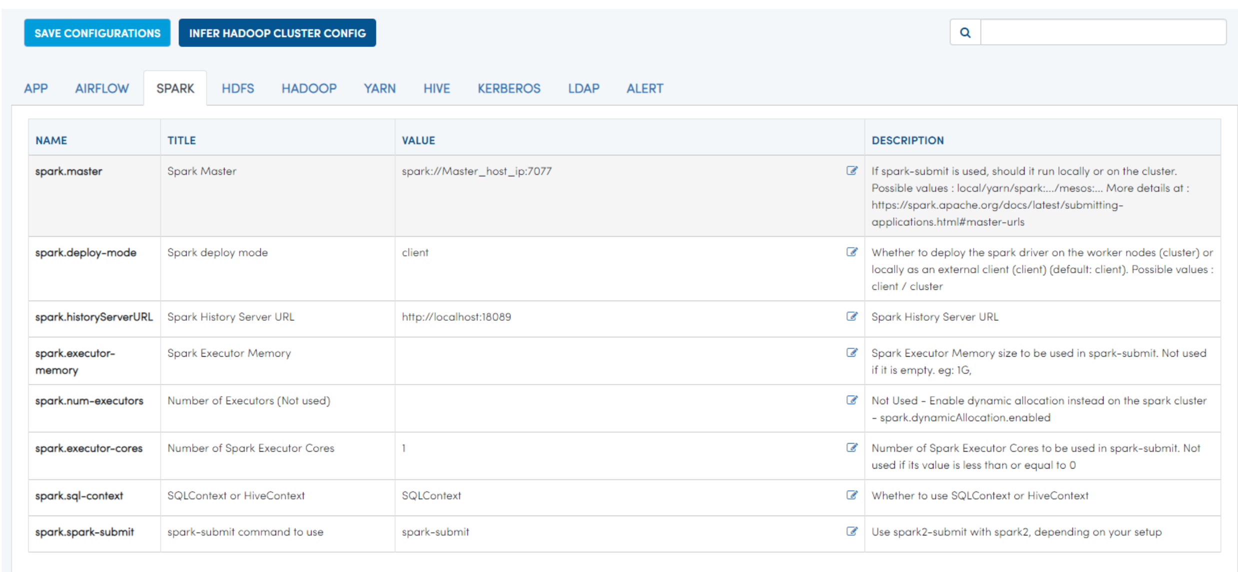
Below are the configuration for Fire to submit the jobs to the Spark Standalone Cluster.

- **Once The server fire & fire-ui start**
  - Login to http://Machine_ip:8090/#/dashboard
  - With password admin/admin.
  - Upload default applications.
  - Create a user ec2-user.
  - Login with ec2-user

### configurations in spark

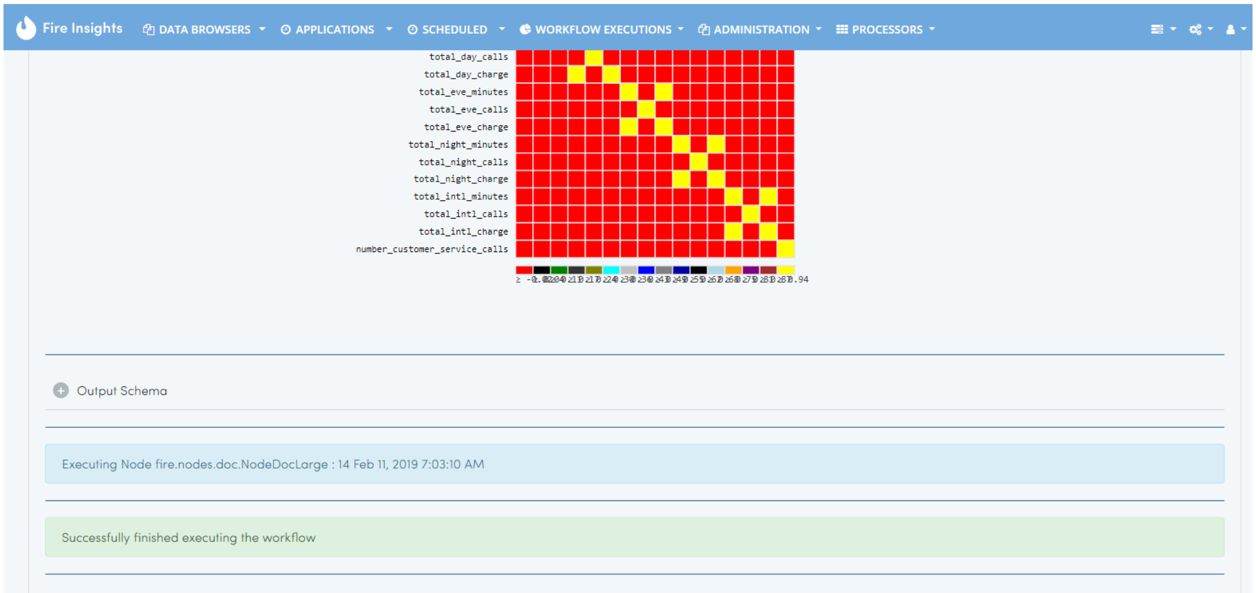
The following configurations have to be set appropriately

- **Go to administration section and open Spark configuration there we need to add Below details in specific setup like below:**
  - spark.master: spark://Master_host_ip:7077
  - spark.deploy-mode: client
  - spark.sql-context: SQLContext
  - After above updates save the configurations.




| NAME | TITLE | VALUE | DESCRIPTION |
|---|---|---|---|
| spark.master | Spark Master | spark://Master_host_ip:7077 | If spark-submit is used, should it run locally or on the cluster. Possible values : local/yarn/spark:.../mesos:... More details at : https://spark.apache.org/docs/latest/submitting-applications.html#master-urls |
| spark.deploy-mode | Spark deploy mode | client | Whether to deploy the spark driver on the worker nodes (cluster) or locally as an external client (client) (default: client). Possible values : client / cluster |
| spark.historyServerURL | Spark History Server URL | http://localhost:18089 | Spark History Server URL |
| spark.executor-memory | Spark Executor Memory | | Spark Executor Memory size to be used in spark-submit. Not used if it is empty. eg: 1G, |
| spark.num-executors | Number of Executors (Not used) | | Not Used - Enable dynamic allocation instead on the spark cluster - spark.dynamicAllocation.enabled |
| spark.executor-cores | Number of Spark Executor Cores | 1 | Number of Spark Executor Cores to be used in spark-submit. Not used if its value is less than or equal to 0 |
| spark.sql-context | SQLContext or HiveContext | SQLContext | Whether to use SQLContext or HiveContext |
| spark.spark-submit | spark-submit command to use | spark-submit | Use spark2-submit with spark2, depending on your setup |

### Now go to application and try to run any workflows




### 3.1.17 Running Fire as a Service

Fire Insights can be configured to run as a service. This way when the machine reboots, Fire Insights would be automatically restarted.

Below are the steps for configuring Fire Insights as a service.

CHAPTER 4

# Authentication

## 4.1 Authentication

Fire Insights supports various types of authencations:

- Database Authentication
- LDAP Authentication
- OAuth Authentication

### 4.1.1 Database Authentication

Fire Insights can authenticate the user against its own database.

User's password are stored encrypted.

This is the default authentication mechanism of Fire Insights. Users created in Fire are stored in the database.

### 4.1.2 LDAP Authentication

Fire Insights can be configured to authenticate the user against LDAP. Users have to be added to Fire, before they can log into Fire and start using it.

The following configurations have to be set appropriately.

| | | | | |
|---|---|---|---|---|
| 23 | ldap.Order | LDAP Order | DB | LDAP Order. Possible values : DB / DB_LDAP. For any change to be reflected required restart of the Sparkflows server. |
| 24 | ldap.URL | LDAP URL | ldap://localhost:10389 | LDAP URL |
| 25 | ldap.Base | LDAP Base | dc=example,dc=com | LDAP Base |
| 26 | ldap.UserDn | LDAP UserDn | uid=john,ou=bindusers,dc=example,dc=com | LDAP UserDn |
| 27 | ldap.Password | LDAP Password | johnspassword | LDAP Password |
| 28 | ldap.UserSearchBase | LDAP UserSearchBase | ou=sparkflow | LDAP UserSearchBase |
| 29 | ldap.UserSearchFilter | LDAP UserSearchFilter | (uid={0}) | LDAP UserSearchFilter |
| 30 | ldap.GroupSearchBase | LDAP GroupSearchBase | ou=groups | LDAP GroupSearchBase |
| 31 | ldap.GroupSearchFilter | LDAP GroupSearchFilter | member={0} | LDAP GroupSearchFilter |

### LDAP Parameters

Table 1: LDAP Parameters

| Name of Parameter | Description | Example |
|---|---|---|
| ldap.Order | Order in which to authenticate the user. Possible values are DB, LDAP_DB, DB_LDAP. | |
| ldap.URL | The URL of the LDAP server. The URL must be prefixed with ldap:// or ldaps://. The URL can optionally specify a custom port, for example: ldaps://ldap_server.example.com:1636. | ldap://localhost:10389 |
| ldap.Base | The distinguished name to use as a search base for finding users and groups. This should be similar to 'dc=sparkflows,dc=com'. | dc=sparkflows,dc=com |
| ldap.UserDn | Distinguished name of the user to bind as. This is used to connect to LDAP/AD for searching user and group information. This may be left blank if the LDAP server supports anonymous binds. | uid=john,ou=development,dc=sparkflows,dc=com |
| ldap.Password | The password of the bind user. | xyz |
| ldap.UserSearchBase | User Search Base | ou=development |
| ldap.UserSearchFilter | The base filter for searching for users. For Active Directory, this is typically '(objectClass=user)'. | For Active Directory : (objectClass=user) Other Example : (uid={0}) |
| ldap.GroupSearchBase | Group Search Base | ou=groups |
| ldap.GroupSearchFilter | Group Search Filter | For Active Directory : (objectClass=group) Other Example : (member={0}) |

### Note

For `ldap.UserSearchFilter` we can use strings like `(uid={USERNAME})` In this case {USERNAME} would be replaced by the real username of the user when searching in LDAP during `Add User`.

### LDAP Certificate

If `ldaps` is being used, the ldap certificate needs to be imported into cacerts.

For Reference : https://docs.oracle.com/cd/E19509-01/820-3399/ggfrj/index.html

#### Importing a user from LDAP into Sparkflows

Once LDAP is enabled in Sparkflows, users can be imported into Sparkflows from LDAP.

- Go to Administration/User
- Click on Add/Sync User
- Enter the username and click on Search
- User details are fetched from LDAP
- Click on Add User to create the user in Sparkflows

#### User Login

Once LDAP is enabled in Sparkflows, all the authentication for login in Sparkflows are done against LDAP.

#### Search Order

Sparkflows would search in LDAP and then in its DB. Search order is determined by the parameter ldap.Order.

If it is set to `LDAP_DB`, it would first search for the User in LDAP and then in its own DB. This allows having the admin user in the Sparkflows DB if needed, so that all users are not locked out of the system in case LDAP goes down or ends up with invalid Configurations.

#### Reference

Below are some great links for reference:

- Active Directory Search Filter Syntax : https://msdn.microsoft.com/en-us/library/aa746475(v=vs.85).aspx

#### What if I get locked out

`ldap.Order` determines the order in which Sparkflows tries to log in the user. In case you are locked out of Sparkflows and are not able to log in, you can do the following:

- Add the below line to conf/configuration.properties:

```
ldap.Order=DB
```

- Then restart the fire server. Now you should be able to log in with your admin account.

Once things are back to normal, you can remove the line you added to `configuration.properties` and restart the fire server.

#### Notes

- Search strings are not case sensitive

### 4.1.3 OAuth Authentication

Fire Insights supports OAuth Authentication.

### Create Users in Fire

First create the user in Fire under `Administration/Users`.

Log into Fire with the `admin` user in order to be able to create the New Users.

### Configuring OAuth

In order the configure OAuth in Fire Insights, add the OAuth configuration parameters to `conf/application.properties`.

Below is an example of configuring OAuth in Fire with Okta.

```
# Okta settings
oauth.client.clientId: 0oadvfdsfsdA7Y68356
oauth.client.clientSecret: YSWFdZf9kfdsfsdfsdfsdnI0SVrswOJpHl
oauth.client.accessTokenUri: https://xyz.okta.com/oauth2/default/v1/token
oauth.client.userAuthorizationUri: https://xyz.okta.com/oauth2/default/v1/authorize
oauth.client.clientAuthenticationScheme: form
oauth.client.scope: openid profile email
oauth.resource.userInfoUri: https://xyz.okta.com/oauth2/default/v1/userinfo
```

### Fire OAuth URL

In order to log in the user into Fire using OAuth, use the following URL:

- http://machine_name:port/login/oauth

This URL will take the user to the OAuth login page. After the user logs in there, the user is redirected back to Fire and is logged in.

If the user is already logged in, going to the above URL, automatically brings up the Fire page for the user.

## 4.1.4 SSO

Single sign-on (SSO) enables you to authenticate your users using your organization's identity provider. If your identity provider supports the SAML 2.0 protocol, you can use Fire Insights SSO to integrate with your identity provider.

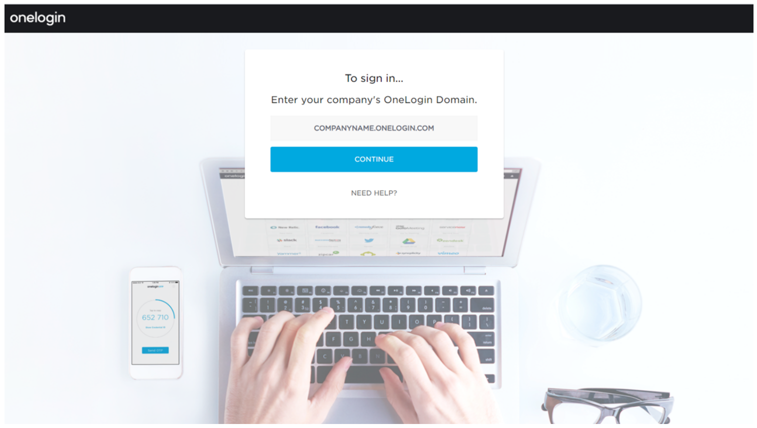
Below are the steps for setting up & configuring OneLogin with Fire Insights.

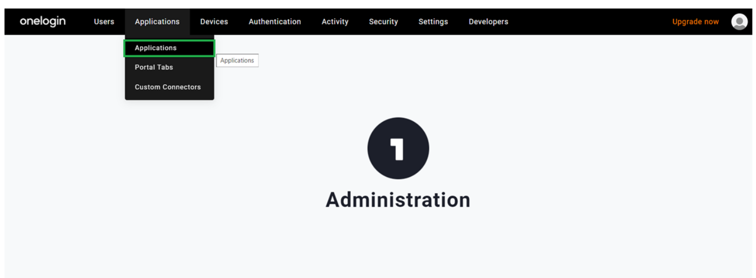
### SAML OneLogin setup

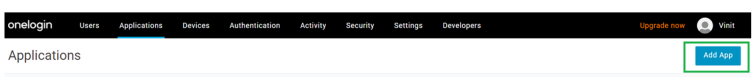
Below are steps to setup SAML 2.0 OneLogin

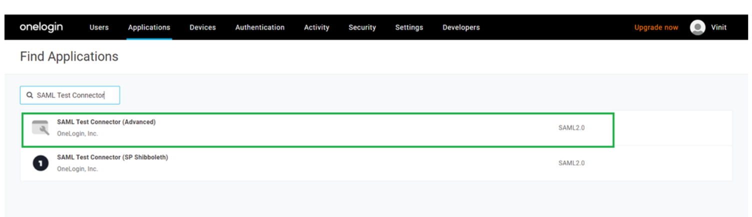
1. Create an account at one Login
2. SignIn into oneLogin
3. Go to administrator >> Click on applications menu >>
4. Add an app
5. Select an application:

```
Search application 'SAML Test Connector'
Select the application SAML Test Connector (Advanced).
```

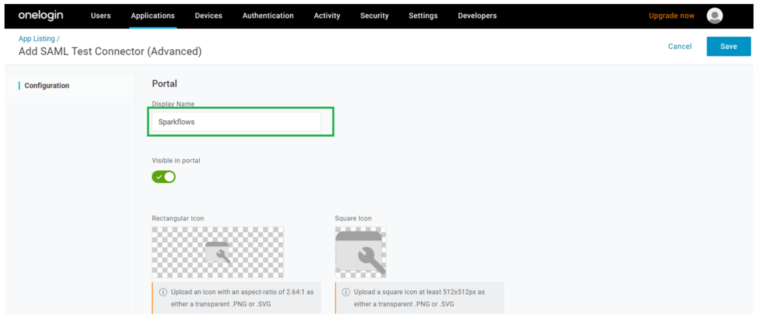

onelogin
To sign in...
Enter your company's OneLogin Domain.
COMPANYNAME.ONELOGIN.COM
CONTINUE
NEED HELP?


onelogin
Users
Applications
Devices
Authentication
Activity
Security
Settings
Developers
Upgrade now
Applications
Portal Tabs
Custom Connectors
Applications
Administration


onelogin
Users
Applications
Devices
Authentication
Activity
Security
Settings
Developers
Upgrade now
Vinit
Applications
Add App

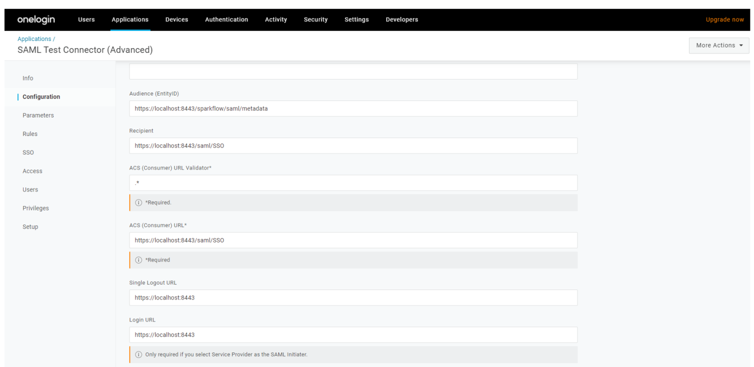

onelogin
Users
Applications
Devices
Authentication
Activity
Security
Settings
Developers
Upgrade now
Vinit
Find Applications
SAML Test Connector
SAML Test Connector (Advanced)
OneLogin, Inc.
SAML2.0
SAML Test Connector (SP Shibboleth)
OneLogin, Inc.
SAML2.0

6. Input an application name and save it.




7. Configure the newly created app and add below information:

```
Audience (EntityID)
Recipient
ACS (Consumer) URL Validator*
ACS (Consumer) URL*
Single Logout URL
Login URL
```




## Fire Insights SAML oneLogin Configuration

Fire Insights can be Configured with SAML 2.0 OneLogin as below.

Go to folder `conf/sso.saml.properties` file:

Add below information from newly created application in oneLogin:

1. Enable sso in sparkflows:

```
sparkflows.sp.sso.enable=true
```

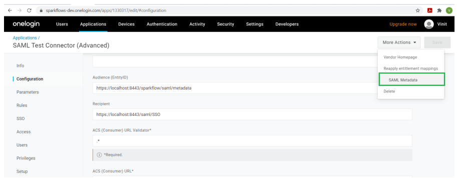
2. Create user locally in application if user doesn't exist in Fire Insights, otherwise app will show page 'User not found':

```
sparkflows.sp.auto.user.create=true
```

3. Metadata url of identity provider.

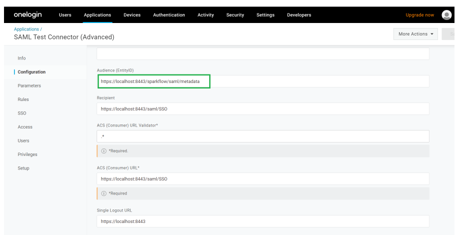
```
saml2.idp.metadata-url=https://sparkflows-dev.onelogin.com/saml/metadata/5f5d16a1-
↪07d1-4167-a305-489d2ee0b18b
```

4. Identifier of the SP entity (must be a URI) Audience URI



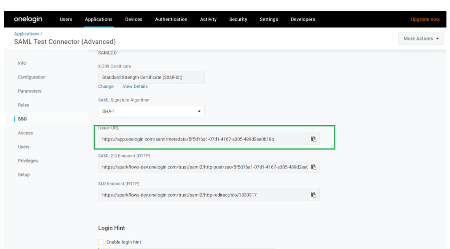
```
saml2.sp.entityid=https://localhost:8443/sparkflow/saml/metadata
```

5. Identifier of the IdP entity (must be a URI)



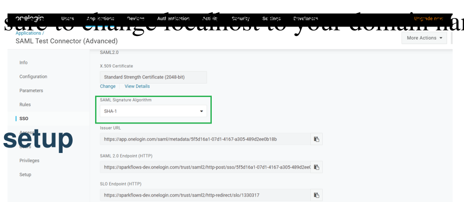
```
saml2.idp.entityid=https://app.onelogin.com/saml/metadata/5f5d16a1-07d1-4167-a305-
↪489d2ee0b18b
```

6. Algorithm that the toolkit will use on signing process.



```
saml2.security.signature_algorithm=http://www.w3.org/2001/04/xmldsig-more#rsa-sha1
```

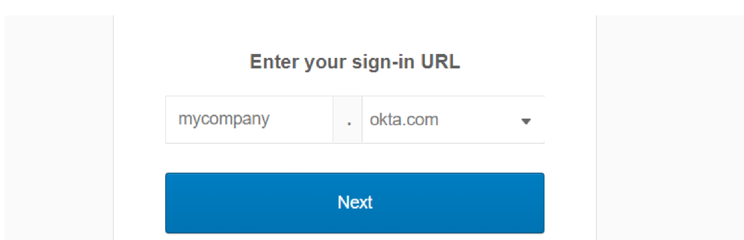
**Note:** Make sure to change localhost to your domain name or your ip

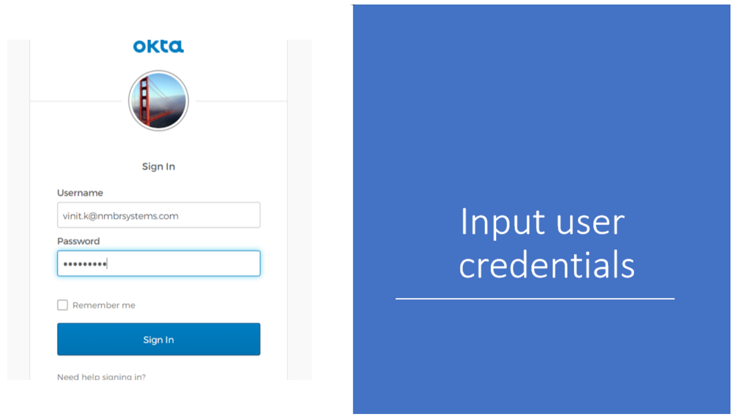


## SAML okta setup

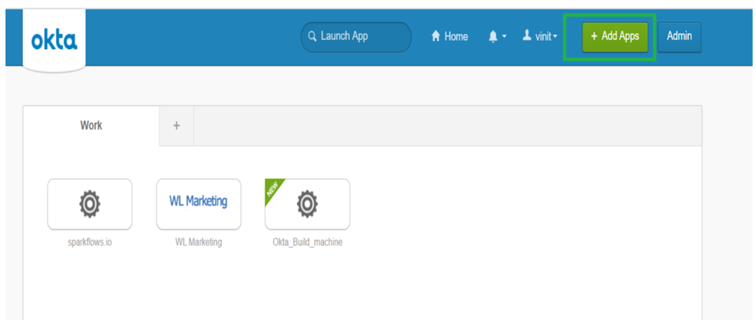
Below are steps to setup SAML 2.0 okta

1. Create an account at okta

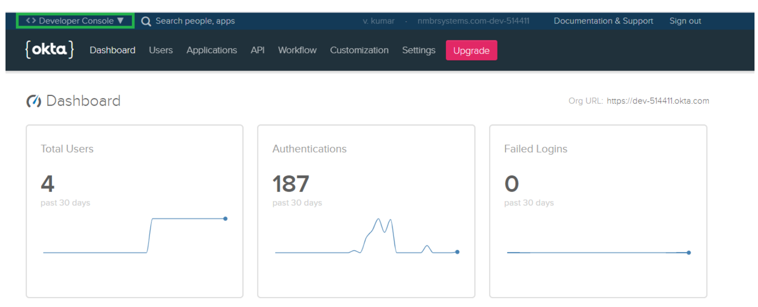





2. SignIn into okta
3. After login go to home and Click on Admin

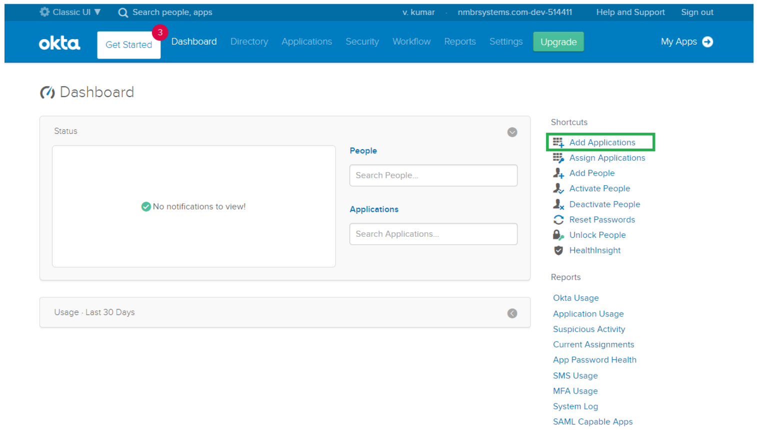


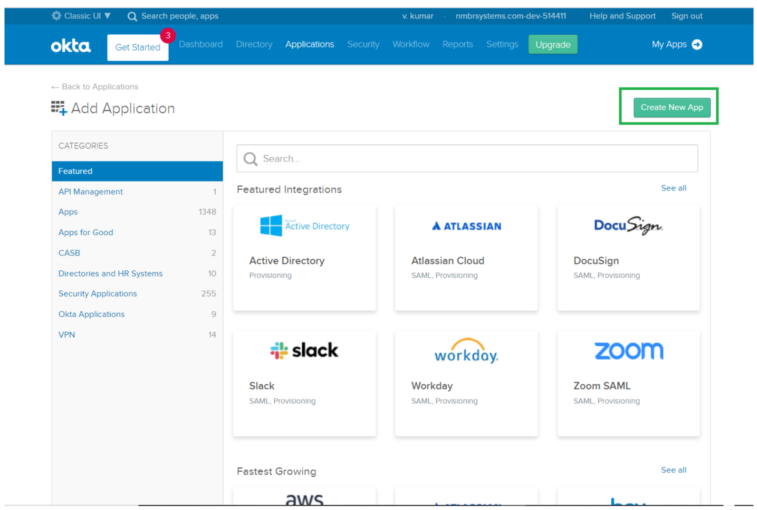

4. Click on Developer Console

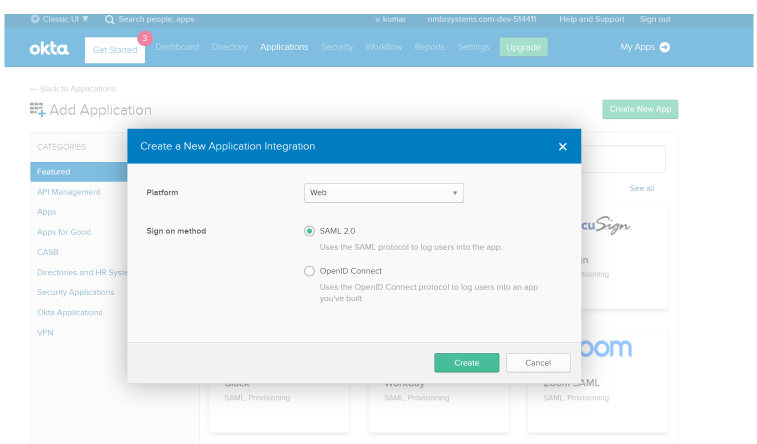


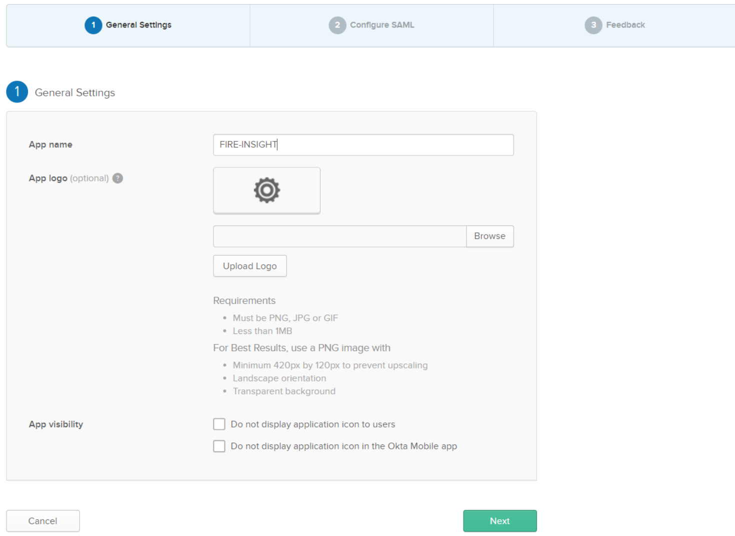

5. Add an app:
6. Create New App:
7. Select SAML 2.0
8. Input app name and click next:
9. Configure the newly created app and add below information
10. Click finish

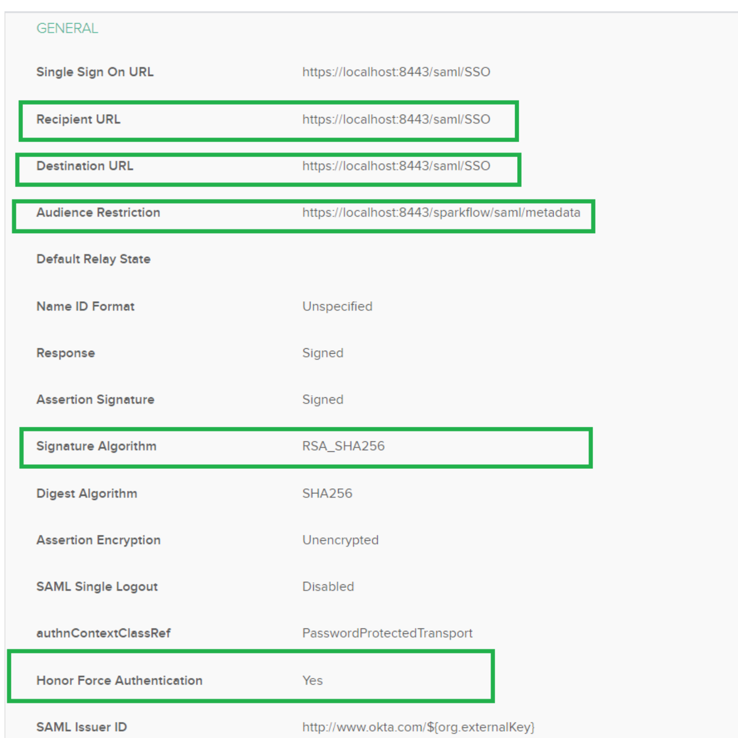
**Note:** Make sure to change localhost to your domain name or your ip

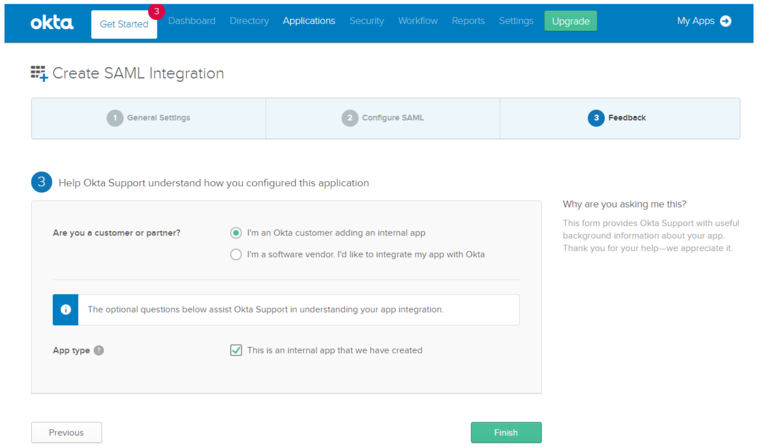

Classic UI
Search people, apps
v. kumar
nmbrsystems.com-dev-514411
Help and Support
Sign out
okta
Get Started
Dashboard
Directory
Applications
Security
Workflow
Reports
Settings
Upgrade
My Apps
Dashboard
Status
People
Search People...
No notifications to view!
Applications
Search Applications...
Usage - Last 30 Days
Shortcuts
Add Applications
Assign Applications
Add People
Activate People
Deactivate People
Reset Passwords
Unlock People
HealthInsight
Reports
Okta Usage
Application Usage
Suspicious Activity
Current Assignments
App Password Health
SMS Usage
MFA Usage
System Log
SAML Capable Apps


Classic UI
Search people, apps
v. kumar
nmbrsystems.com-dev-514411
Help and Support
Sign out
okta
Get Started
Dashboard
Directory
Applications
Security
Workflow
Reports
Settings
Upgrade
My Apps
← Back to Applications
Add Application
Create New App
CATEGORIES
Featured
API Management 1
Apps 1348
Apps for Good 13
CASB 2
Directories and HR Systems 10
Security Applications 255
Okta Applications 9
VPN 14
Search...
Featured Integrations
See all
Active Directory
Active Directory
Provisioning
ATLASSIAN
Atlassian Cloud
SAML, Provisioning
DocuSign
DocuSign
SAML, Provisioning
slack
Slack
SAML, Provisioning
workday.
Workday
SAML, Provisioning
zoom
Zoom SAML
SAML, Provisioning
Fastest Growing
See all
aws


okta
Add Application
Create New App
Create a New Application Integration
Platform
Web
Sign on method
SAML 2.0
Uses the SAML protocol to log users into the app.
OpenID Connect
Uses the OpenID Connect protocol to log users into an app you've built.
Create
Cancel


1 General Settings
2 Configure SAML
3 Feedback
1 General Settings
App name
FIRE-INSIGHT
App logo (optional)
Browse
Upload Logo
Requirements
Must be PNG, JPG or GIF
Less than 1MB
For Best Results, use a PNG image with
Minimum 420px by 120px to prevent upscaling
Landscape orientation
Transparent background
App visibility
Do not display application icon to users
Do not display application icon in the Okta Mobile app
Cancel
Next


GENERAL
Single Sign On URL
https://localhost:8443/saml/SSO
Recipient URL
https://localhost:8443/saml/SSO
Destination URL
https://localhost:8443/saml/SSO
Audience Restriction
https://localhost:8443/sparkflow/saml/metadata
Default Relay State
Name ID Format
Unspecified
Response
Signed
Assertion Signature
Signed
Signature Algorithm
RSA_SHA256
Digest Algorithm
SHA256
Assertion Encryption
Unencrypted
SAML Single Logout
Disabled
authnContextClassRef
PasswordProtectedTransport
Honor Force Authentication
Yes
SAML Issuer ID
http://www.okta.com/${org.externalKey}




### Fire Insights SAML Okta Configuration

Fire Insights can be configured with SAML 2.0 Okta as below.

Go to folder `conf/okta.saml.properties` file:

Add below information from newly created application in oneLogin:

1. Enable sso in sparkflows:

```
sparkflows.sp.sso.enable=true
```

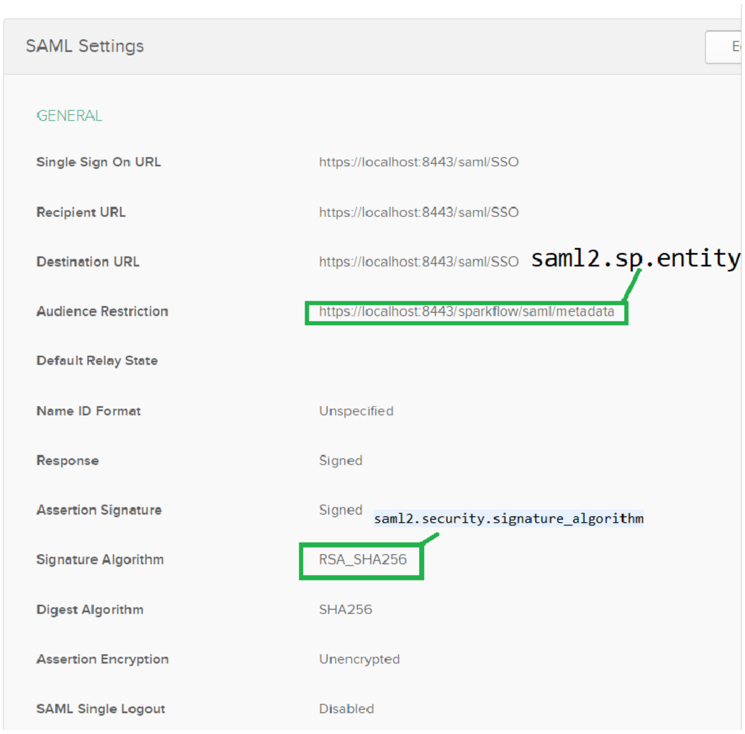
2. Create user locally in application if user doesn't exist in Fire Insights, otherwise app will show page 'User not found':

```
sparkflows.sp.auto.user.create=true
```

3. Copy Okta config info

```
# Identifier of the SP entity (must be a URI) Audience URI
saml2.sp.entityid=https://localhost:8443/sparkflow/saml/metadata
# Algorithm that the toolkit will use on signing process
saml2.security.signature_algorithm=http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
```

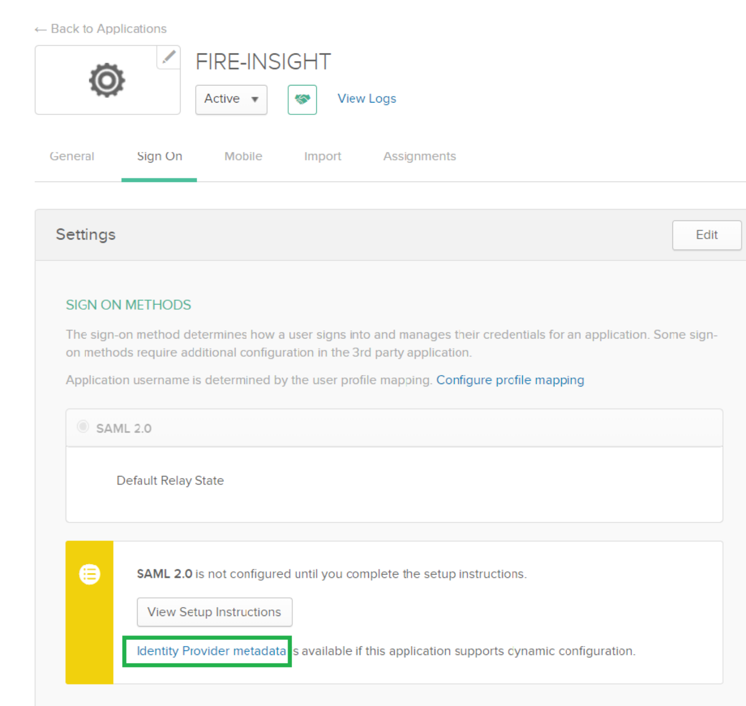
4. Right click on identity provider metadata and select Copy link address

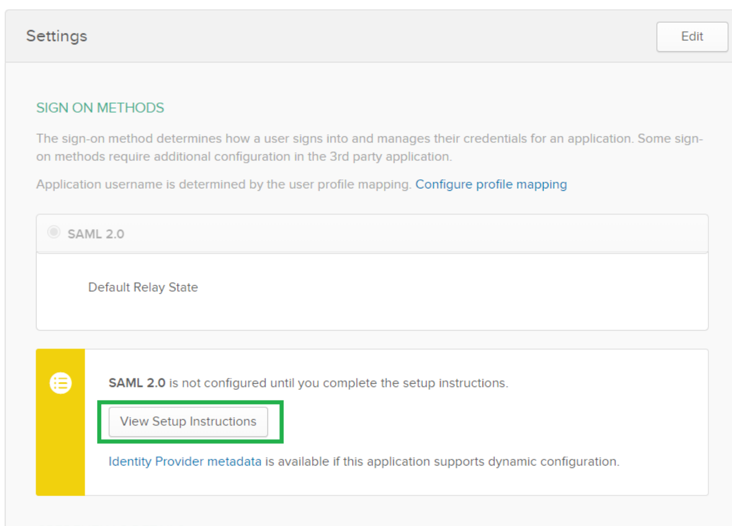


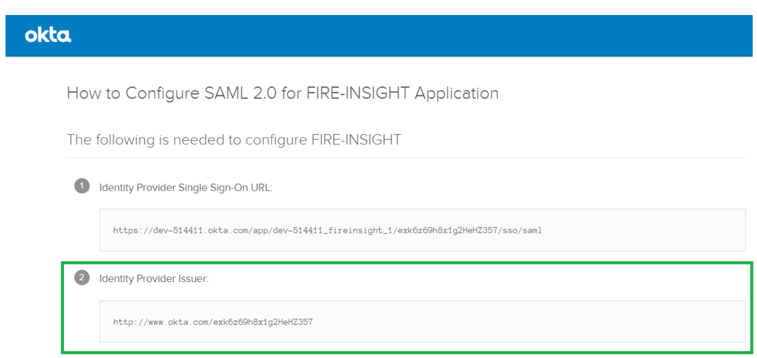

```
#Metadata url of identity provider
saml2.idp.metadata-url=https://dev-514411.okta.com/app/exk6sc27dyq4istq0357/sso/saml/metadata
```

5. Capture Issuer url

**Note:** Make sure to change localhost to your domain name or your ip


← Back to Applications
FIRE-INSIGHT
Active
View Logs
General
Sign On
Mobile
Import
Assignments
Settings
Edit
SIGN ON METHODS
The sign-on method determines how a user signs into and manages their credentials for an application. Some sign-on methods require additional configuration in the 3rd party application.
Application username is determined by the user profile mapping. Configure profile mapping
SAML 2.0
Default Relay State
SAML 2.0 is not configured until you complete the setup instructions.
View Setup Instructions
Identity Provider metadata is available if this application supports dynamic configuration.


Settings
Edit
SIGN ON METHODS
The sign-on method determines how a user signs into and manages their credentials for an application. Some sign-on methods require additional configuration in the 3rd party application.
Application username is determined by the user profile mapping. Configure profile mapping
SAML 2.0
Default Relay State
SAML 2.0 is not configured until you complete the setup instructions.
View Setup Instructions
Identity Provider metadata is available if this application supports dynamic configuration.
CREDENTIALS DETAILS


okta
How to Configure SAML 2.0 for FIRE-INSIGHT Application
The following is needed to configure FIRE-INSIGHT
1 Identity Provider Single Sign-On URL:
https://dev-514411.okta.com/app/dev-514411_fireinsight_1/exk6z69h8x1g2HeHZ357/sso/saml
2 Identity Provider Issuer:
http://www.okta.com/exk6z69h8x1g2HeHZ357

CHAPTER 5

# Security

## 5.1 Security

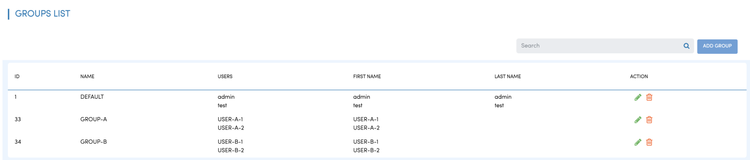
### 5.1.1 User Group Role Permission

Fire Insights supports Users, Groups, Roles, Permissions. A User can belong to multiple groups and have multiple roles.

Each role can have multiple permissions.

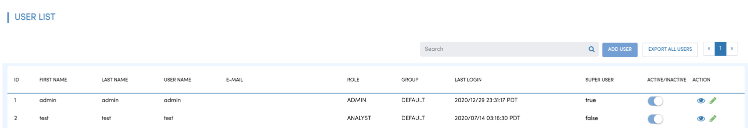
#### Groups

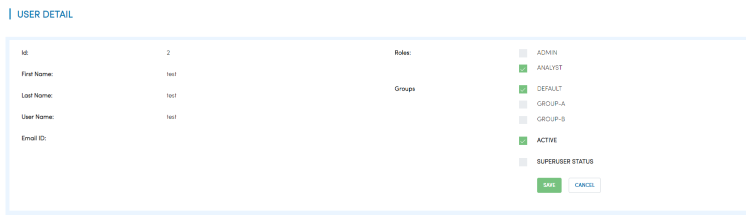
There can be multiple groups in Fire Insights.




#### Users

Fire Insights supports multiple users. Each user can belong to multiple groups, and also have multiple Roles.

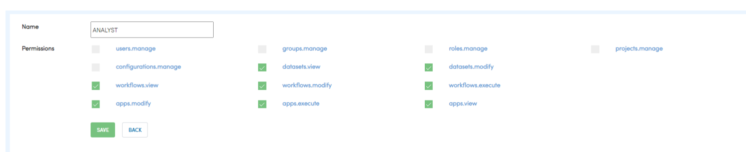





### Permissions

Fire Insights supports the following Permissions. Permissions are associated with Roles.

| Title | Description |
|---|---|
| users.manage | create, modify & disable user |
| groups.manage | Create, modify & delete the group |
| roles.manage | Create, modify & delete the roles |
| projects.manage | Create, modify & delete the projects |
| configurations.manage | modify diifferent configurations |
| datasets.view | view dataset in specified project |
| datasets.modify | modify datasets in specified project |
| workflows.view | view workflows in specified project |
| workflows.modify | modify workflows in specified project |
| workflows.execute | execute workflow in specified project |
| apps.modify | modify analytics application |
| apps.execute | execute analytics application |
| apps.view | view analytics application |




### Roles

A user can have multiple Roles. The actions which a user can do depends on the Roles they belong to.

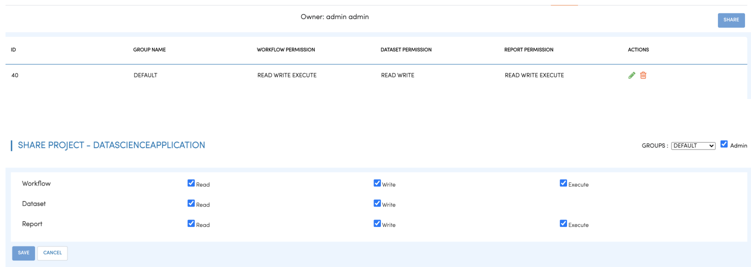

## 5.1.2 Sharing Projects

A project can be shared with multiple Groups. A Project is visible only to those users who belong to the groups with whom it has been shared with.

Below, the Project is shared with the `DEFAULT` group.

The following permissions can be given to a group during sharing of the project.

All users belonging to the group get the associated permissions on the Project.

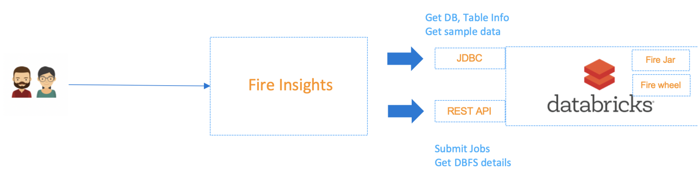


### 5.1.3 Databricks Security

Users in Fire Insights access Databricks via Databricks Tokens.

Whenever users interact with Databricks in Fire Insights, they have the access which is assigned to the token in Databricks.

Below diagrams show the integration of Fire Insights with Databricks.




#### Viewing DB/Tables

In Fire Insights users can view the databases and tables. They are accessed via JDBC from Databricks cluster using the token.

The same applies if users chose to execute a query to view a few records from the table.

#### Executing Workflows

When users execute workflows in Fire Insights, they are submitted to the Databricks cluster view the REST API using the Databricks token. These jobs post back messages to Fire Insights. They use a token generated specifically for the job to post back the messages.

#### Databricks Connections

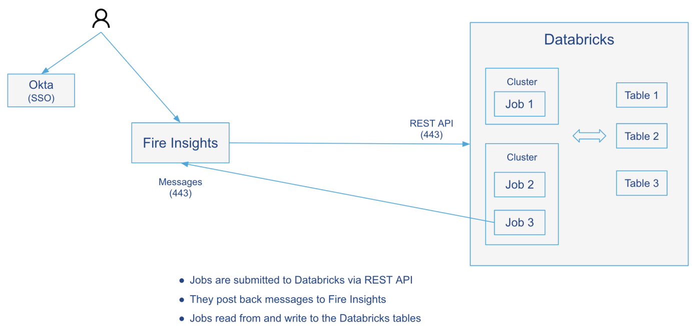
The Databricks cluster details and token are specified in a Connection. The user uses the connections when talking to Databricks.

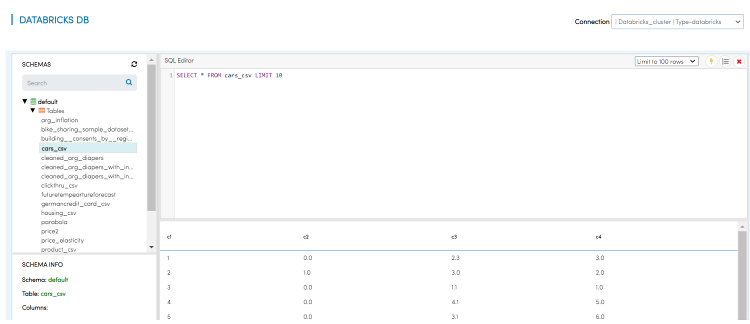
Connections can be at the global level or at the Project level. Global level connections are created by the admin user. Project level connections are created by the Project users.

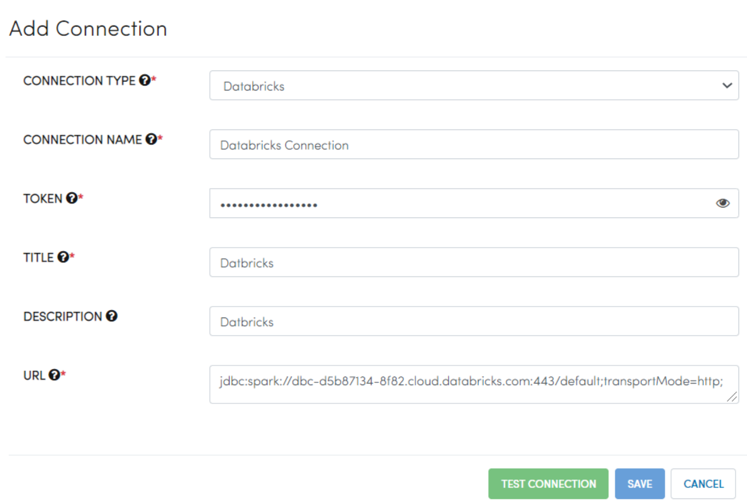
Fire Insights would also support defining Group level connections.

### 5.1.4 Admin user

Fire Insights support variety of permissions for Roles. Each user can be assigned one more more Roles.

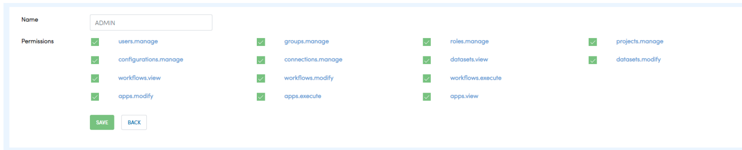

Okta
(SSO)
Fire Insights
REST API
(443)
Messages
(443)
Databricks
Cluster
Job 1
Cluster
Job 2
Job 3
Table 1
Table 2
Table 3
• Jobs are submitted to Databricks via REST API
• They post back messages to Fire Insights
• Jobs read from and write to the Databricks tables


DATABRICKS DB
Connection
SCHEMAS
SQL Editor
SCHEMA INFO


Add Connection
CONNECTION TYPE *
Databricks
CONNECTION NAME *
Databricks Connection
TOKEN *
TITLE *
Datbricks
DESCRIPTION
Datbricks
URL *
jdbc:spark://dbc-d5b87134-8f82.cloud.databricks.com:443/default;transportMode=http;
TEST CONNECTION
SAVE
CANCEL

## Permissions supported by Fire Insights

Below are the permissions supported by Fire Insights.



| Title | Description |
|---|---|
| users.manage | create, modify & disable user |
| groups.manage | Create, modify & delete the group |
| roles.manage | Create, modify & delete the roles |
| projects.manage | Create, modify & delete the projects |
| configurations.manage | modify diiferent configurations |
| connections.manage | add & modify diifferent connections |
| datasets.view | view dataset in specified project |
| datasets.modify | modify datasets in specified project |
| workflows.view | view workflows in specified project |
| workflows.modify | modify workflows in specified project |
| workflows.execute | execute workflow in specified project |
| apps.modify | modify analytics application |
| apps.execute | execute analytics application |
| apps.view | view analytics application |

## Permissions for Admin User

In Fire Insights generally the below permissions are associated with Admin features

- users.manage
- groups.manage
- roles.manage
- configurations.manage

An admin user in Fire Insights is one who has users.manage permission.

## Admin User Rights

The Admin user gets the following rights.

## Operating Fire Insights

In Fire Insights an admin user can do the following administration tasks:

- Configure Fire Insights
- Run Diagnostics
- Manage Users, Groups, Permissions
- Load Sample Projects

- View Server Logs
- Cleanup Data

### Projects/Data etc

As regards to Projects, the Admin user can do the following:

- View all the Projects
- View the executions of all the workflows
- View the executions of all the Analytical Apps
- **Onboarding Analytics Apps for a Customer**
  - Who creates that Project which will hold the Analytics App? Admin user. Now the admin user becomes the owner of that Project and be able to see everything.
  - Who shares that project with the Group of the Customer? Admin user.

### Deleting Users/Groups

In Fire Insights, users and groups cannot be deleted. Users can be made inactive.

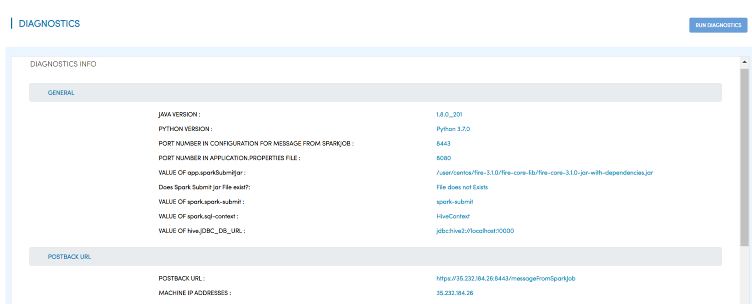
### Superuser

A user in Fire Insights can be marked to be a super-user. A super-user has all the same rights as the admin user.

### Details on the Admin user rights

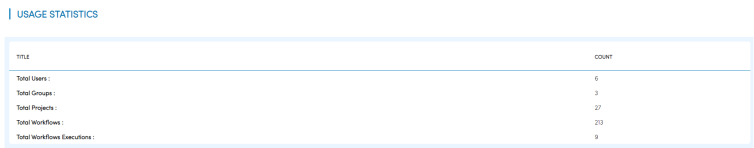
### Diagnostics

The admin user can view detailed informations about Machine environments.



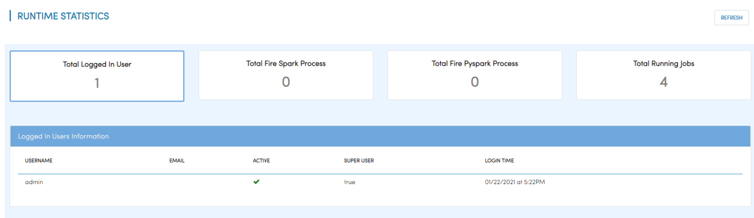

### Usage Statistics

The admin user can view Total Users, Groups, Roles, Projects, Workflows & Workflows Executions



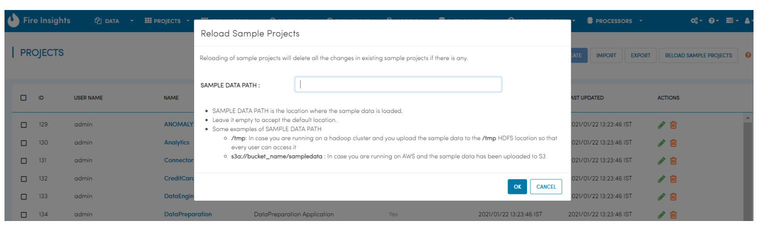

### Runtime Statistics

The admin user can view Total Logged In Users, Total Fire Spark Processes, Total Fire Pyspark Processes & Total Running Jobs




### Sample Projects

The admin user can RELOAD SAMPLE PROJECTS, as by default Fire Insights comes with sample projects containing different types of workflows & datasets




### Global Connections

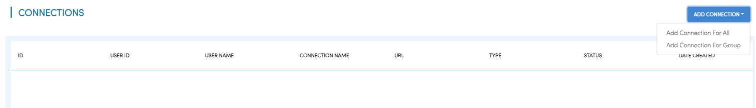
The admin user can Add Connections which everyone can use and also connections at the Group Level.

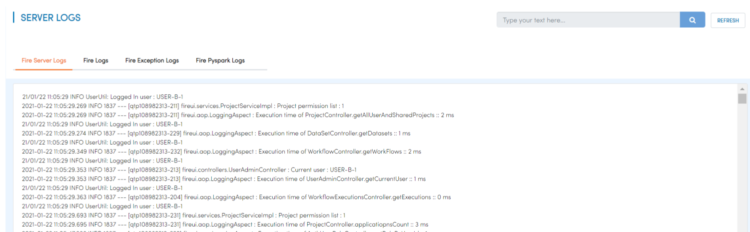
### Server Logs

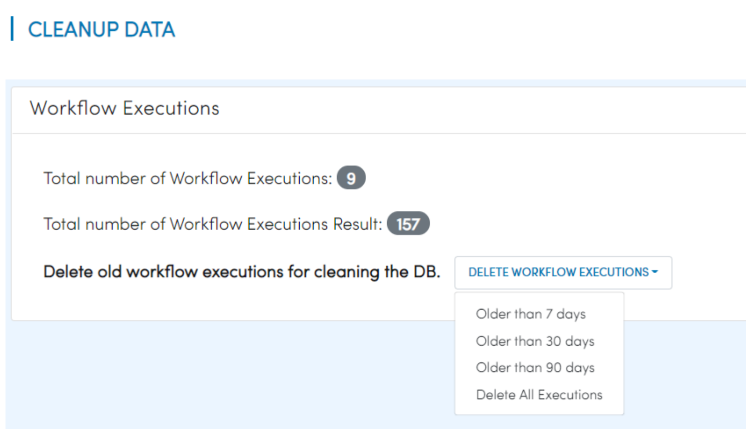
The admin user can view Fire Server Logs, Fire Logs, Fire Exception Logs & Fire Pyspark Logs

### Cleanup Data

The admin user can Delete old workflow executions for cleaning the DB which is Older than Last 7 days, Older than Last 30 days, Older than Last 90 days & Delete All Executions


CONNECTIONS
ADD CONNECTION
Add Connection For All
Add Connection For Group
ID
USER ID
USER NAME
CONNECTION NAME
URL
TYPE
STATUS


SERVER LOGS
REFRESH
Fire Server Logs
Fire Logs
Fire Exception Logs
Fire Pyspark Logs


CLEANUP DATA
Workflow Executions
Total number of Workflow Executions: 9
Total number of Workflow Executions Result: 157
Delete old workflow executions for cleaning the DB.
DELETE WORKFLOW EXECUTIONS
Older than 7 days
Older than 30 days
Older than 90 days
Delete All Executions

# CHAPTER 6

# Operating Fire Insights

## 6.1 Operating Guide

### 6.1.1 Logs in Fire Insights

In Fire Insights there are 2 processes which run:

- fire server
- fire engine

#### Logs for Fire Web Server

The logs for Fire Web Server go into fireserver.log. The logging level is determined by the properties file conf/log4j.properties.

#### Example log4j.properties

#### How to change the various logging levels

#### Logs for Fire Engine

The logs for Fire Engine go into fire.log.

### 6.1.2 Installing JDBC Drivers for Workflows

Fire has JDBC Processors for reading from JDBC sources or writing to JDBC sinks.

In order to connect to a JDBC source like Oracle/DB2 etc. the JDBC driver needs to be installed in Fire.

Below are the steps for installing the JDBC driver into Fire:

- *Download the JDBC jar file*
- *Copy it into 'fire-user-lib' directory under the Fire installation*
- *Restart fire*

### Download the JDBC jar file

Download the JDBC jar file for the Database you are looking to connect to.

### Copy it into fire-user-lib

Under the Fire installation directory, there is `fire-user-lib` directory.

Copy the downloaded JDBC jar file into it.

### Stop Fire Processes

Stop the running Fire processes with `./run-fire.sh stop`

They will be restarted automatically.

### Running Workflows depending on the jars added

When running workflows which depend on the jar file, select the checkbox for that jar file in the Workflow Execution Page.

### Downloading the JDBC jar files

### MySQL

- MySQL connector can be downloaded from : https://dev.mysql.com/downloads/connector/j/
- After downloading untar it with : `tar xvf mysql-connector-java-5.1.46.tar.gz`
- After untaring the jdbc jar file is available in the directory
- Use the jar file (mysql-connector-java-5.1.46.jar) for installation in Fire

### PostgreSQL

- PostgresSQL JDBC drivers can be downloaded from : https://jdbc.postgresql.org/download.html

### Oracle

- Oracle JDBC drivers can be downloaded from : https://www.oracle.com/technetwork/database/features/jdbc/jdbc-drivers-12c-download-1958347.html

#### JDBC Drivers

When using the JDBC processors, the following can be used for the JDBC Driver. Below are the JDBC URL's for some databases:

- MySQL : com.mysql.jdbc.Driver
- PostgreSQL : org.postgresql.Driver
- Oracle : oracle.jdbc.driver.OracleDriver

#### Example JDBC URL

Below are some example JDBC URL for reading from Relational sources when using the JDBC Processors:

- MySQL : jdbc:mysql://localhost:3306/mydb
- PostgreSQL : jdbc:postgresql://localhost:5432/mydb

### 6.1.3 Installing JDBC Drivers for Interactive Dashboard

Interactive Dashboard work with JDBC sources. The appropriate JDBC jars have to be installed.

Below are the steps for installing the JDBC driver for Interactive Dashboards:

- *Download the JDBC jar file*
- *Copy it into 'fire-server-lib' directory under the Fire installation*
- *Restart fire-server*

#### Download the JDBC jar file

Download the JDBC jar file for the Database you are looking to connect to.

#### Copy it into fire-server-lib

Under the Fire installation directory, there is `fire-server-lib` directory.

Copy the downloaded JDBC jar file into it.

#### Restart Fire Server

Restart Fire with `./run-fire-server.sh restart`

Fire does not need to be restarted.

#### Downloading MySQL Connector

- MySQL connector can be downloaded from : https://dev.mysql.com/downloads/connector/j/
- After downloading untar it with : `tar xvf mysql-connector-java-5.1.46.tar.gz`
- After untaring the jdbc jar file is available in the directory
- Use the jar file (mysql-connector-java-5.1.46.jar) for installation in Fire

### 6.1.4 Running Tesseract in Fire

In order to run Tesseract, perform the below installation steps:

#### Download & Install the Tesseract Language Data files

- Download and Install the tesseract language data files on each of the worker nodes of the cluster
- Install them in the same directory on each of the worker nodes
  - `git clone https://github.com/tesseract-ocr/tessdata.git`
- Make sure that the tessdata directory is accessible to all the users.

#### Set TESSDATA_PREFIX as an Environment Variable and restart the Sparkflows server

- Point the environment variable TESSDATA_PREFIX to the tessdata directory
  - `export TESSDATA_PREFIX=/home/centos/tessdata`
- Restart the sparkflows server
- If the above is not done correctly, then the Sparkflows server would exit when any OCR node is run

#### Include TESSDATA_PREFIX in spark configs when submitting the job

Include the following in spark configs when running workflows containing the OCR node:

- `--conf spark.executorEnv.TESSDATA_PREFIX=/home/centos/tessdata`
- where the tesseract language data files are in `/home/centos/tessdata` directory on each of the worker nodes

#### Error if TESSDATA_PREFIX is not set correctly

If `TESSDATA_PREFIX` is not set, the spark program would run into the error below.

- Error opening data file /Users/saudet/projects/bytedeco/javacpp-presets/tesseract/cppbuild/macosx-x86_64/share/tessdata/eng.traineddata
- Please make sure the TESSDATA_PREFIX environment variable is set to the parent directory of your "tessdata" directory.
- Failed loading language 'eng'
- Tesseract couldn't load any languages!

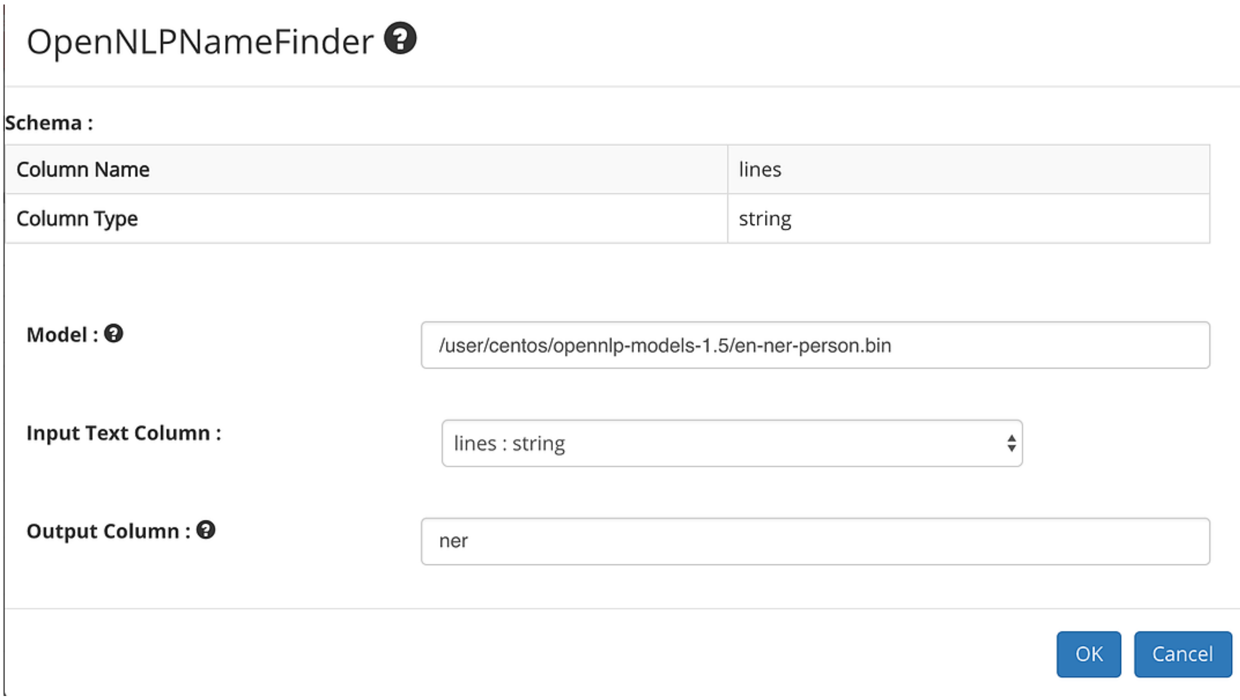
### 6.1.5 Running Apache OpenNLP Model Jars in Fire Insights

#### When running locally

- Create a directory called opennlp-models-1.5 on the local file system
- Download the Apache OpenNLP model jar from : http://opennlp.sourceforge.net/models-1.5/
  - eg: wget http://opennlp.sourceforge.net/models-1.5/en-ner-person.bin
- Copy the Apache OpenNLP model jar into the opennlp-models-1.5 directory created

### When running on a Spark cluster

- Copy the model file onto HDFS into a directory called opennlp-models-1.5
- For example /user/centos/opennlp-models-1.5/en-ner-person.bin
- The model file should be accessible by all the users who would use it




## 6.1.6 Installing/Using OpenNLP model jars

### When running locally

- Create a directory called opennlp-models-1.5 on the local file system
- Download the OpenNLP model jar from : http://opennlp.sourceforge.net/models-1.5/
  - eg: wget http://opennlp.sourceforge.net/models-1.5/en-ner-person.bin
- Copy the OpenNLP model jar into the opennlp-models-1.5 directory created

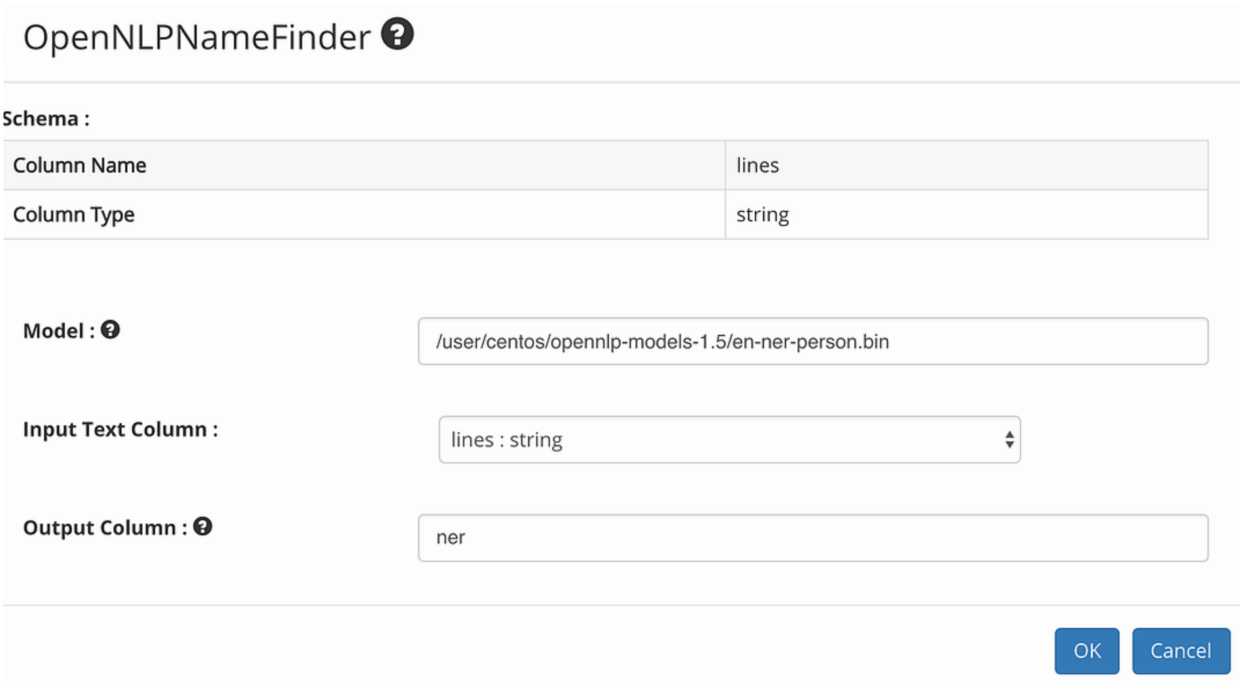
### When running on a Spark cluster

- Copy the model file onto HDFS into a directory called opennlp-models-1.5
- For example /user/centos/opennlp-models-1.5/en-ner-person.bin
- The model file should be accessible by all the users who would use it

### Using OpenNLP model jars

- Specify the path of the jar file in the dialog box of the Open NLP nodes in the workflow

- For example for the OpenNLPNameFinder node the path can be : /user/centos/opennlp-models-1.5/en-ner-person.bin




### 6.1.7 Using Juypter

Jupyter is extensively used by Data Scientists.

#### Overview

Fire can be used to easily create a downsampled dataset. Fire provides a `sample` processor for it.

Once the dataset size has been reduced, Data Scientists can model with it in Jupyter.

Once the modeling process is complete, the algorithm can be run on the full data in Fire.

### 6.1.8 Maintenance Tasks

#### Cleaning H2 DB

Fire Insights by default uses the H2 embedded database.

It is important to keep the size of the database in control. All the Fire Insights tables are relatively small except those which store the result of workflow execution.

#### Cleaning Old Workflow Executions

It is important to regularly delete the old workflow executions in order to keep the size of the H2 DB in control.

- Go to the Administration/Cleanup Data

- Click on Delete old Workflow Executions in order to delete the old workflow executions.

#### Compact H2 DB File

If the H2 DB file size grows too large (> 3GB), then follow the steps below for compacting it.

By default H2 DB file is in the home folder of the user running Fire Insights. It is named as firedb.mv.db

- Store Fire Insights
- Make a copy of firedb.mv.db file to be safe
- Use the commands below for compacting it

java -cp ~/fire-3.1.0/db/h2/h2-1.4.199.jar org.h2.tools.Shell URL: jdbc:h2:./firedb Driver : org.h2.Driver User : fire Password : fire

SHUTDOWN COMPACT

#### Deleting old files

Regularly delete the following folders:

- /tmp/fire/workflowlogs
- /tmp/fire/workflows

### 6.1.9 Installing MySQL

This document captures the details for installing MySQL on Centos7

#### Steps for installing MySQL on Centos7

- Check your hostname

To check your hostname run:

```
hostname
hostname -f
```

- Update your system

Run below command to update your system:

```
sudo yum update
```

- Install wget if its not on your system

You will need wget to complete this guide. It can be installed as follows:

```
sudo yum install wget
```

### Install MySQL

MySQL must be installed from the community repository.

- Download and add the repository

Download and add the repository, then update:

```
wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm
sudo yum update
```

- Install MySQL as usual and start the service

Install MySQL as usual and start the service. During installation, you will be asked if you want to accept the results from the .rpm file's GPG verification. If no error or mismatch occurs, enter y:

```
sudo yum install mysql-server
sudo systemctl start mysqld
```

### Harden MySQL Server

- Harden security Concern

Run the mysql_secure_installation script to address several security concerns in a default MySQL installation:

```
sudo mysql_secure_installation
```

- To check Existing password generated

To check Existing password generated:

```
sudo grep 'temporary password' /var/log/mysqld.log
```

- You can also create new password while installing too.

### Using MySQL

The standard tool for interacting with MySQL is the mysql client which installs with the mysql-server package. The MySQL client is used through a terminal

- Root Login

To log in to MySQL as the root user:

```
mysql -u root -p
```

- When prompted, enter the root password you assigned when the mysql_secure_installation script was run

You'll then be presented with a welcome header and the MySQL prompt as shown below:

```
mysql>
```

### To Provide access from remote pcs

Inorder to Access MySQL from Remote PC, run below command:

```
CREATE USER 'root'@'%' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

NOTES * The Port on which MySQL Running ie 3306, should be accessible from target machine.

### Create a New MySQL User and Database

In the example below, testdb is the name of the database, testuser is the user, and password is the user's password:

```
create database testdb;
create user 'testuser'@'localhost' identified by 'password';
grant all on testdb.* to 'testuser' identified by 'password';
```

### Create a Sample Table

- Log back in as testuser

Login with testuser:

```
mysql -u testuser -p
```

- Create a sample table

Create a sample table called customers. This creates a table with a customer ID field of the type INT for integer (auto-incremented for new records, used as the primary key), as well as two fields for storing the customer's name:

```
use testdb;
create table customers (customer_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, first_
↪name TEXT, last_name TEXT);
```

### Reset the MySQL Root Password

If you forget your root MySQL password, it can be reset.

- Stop the current MySQL server instance

Stop the current MySQL server instance, then restart it with an option to not ask for a password:

```
sudo systemctl stop mysqld
sudo mysqld_safe --skip-grant-tables &
```

- Reconnect to the MySQL server

Reconnect to the MySQL server with the MySQL root account:

```
mysql -u root
```

- Use the following commands to reset root's password

Use the following commands to reset root's password. Replace password with a strong password:

```
use mysql;
update user SET PASSWORD=PASSWORD("password") WHERE USER='root';
flush privileges;
exit
```

- Restart MySQL

Then restart MySQL:

```
sudo systemctl start mysqld
```

### MySQL JDBC Driver

Download the MySQL JDBC driver from http://www.mysql.com/downloads/connector/j/5.1.html

Extract the JDBC driver JAR file from the downloaded file. For example:

tar zxvf mysql-connector-java-8.0.11.tar.gz

mysql-connector-java.jar

CHAPTER 7

Quick Start Guide

## 7.1 Quickstart Guide

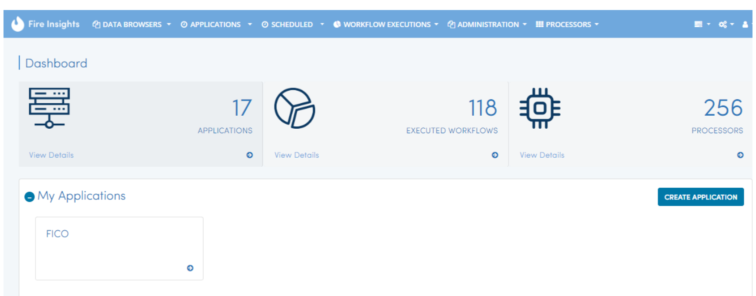
The quickstart gets you started with Fire Insights.

Let's get started!

### 7.1.1 Step 1: Create Project

Before you can start creating a workflow, you will need to create a 'Project'. Project is a bucket where all your artifacts such as datasets, workflows, dashboards etc. related to a project would reside. Projects are equivalent to workspaces in IDEs.

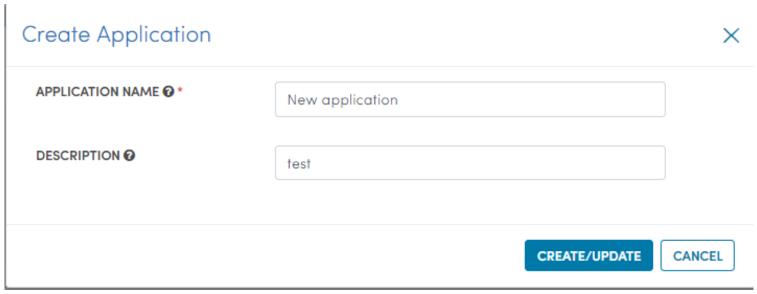
From the landing page of Fire Insights, click on "Create Application" to create a new application.

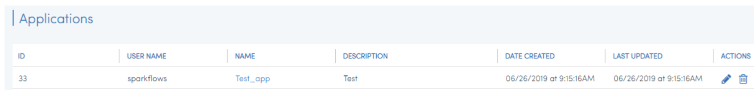



Specify name and description, and click on "Create/Update" button. The new application is created and it is now ready to use.

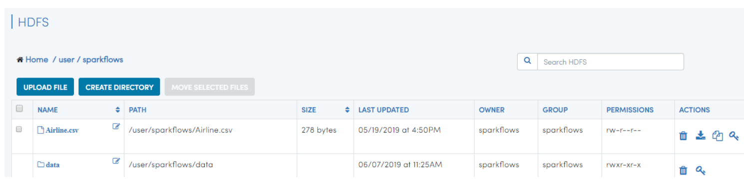
### 7.1.2 Step 2 : Upload Data Files

Every workflow needs data to work on. As a next step, you will upload a CSV file that you want to process in your workflow.




If you have your data in CSV file, click on "Data Browsers" and select "HDFS". Your home directory will be displayed. Initially, it will be empty as you have not uploaded any file.




Click on "Upload File" button. Choose one or more CSV files that you want to upload.

After selecting the files, click "Upload All".

In order to use CSV files in workflow, Fire Insights requires that you wrap them in datasets. In the next step, you will create datasets from the files you have just uploaded.

### 7.1.3 Step 3 : Create Dataset

Before any data can be used in a workflow, it needs to be wrapped in a dataset. If you uploaded CSV files in the previous step, in this step you will wrap them in a dataset.

The steps involved in creating a dataset are:

- *Open the Application where you want to create dataset*
- *Click on "Datasets" tab*
- *Click on "Create" and choose "Datasets"*
- *Select your dataset type and enter the fields in the dialog*
- *Update the schema of the dataset*
- *Click "Save"*

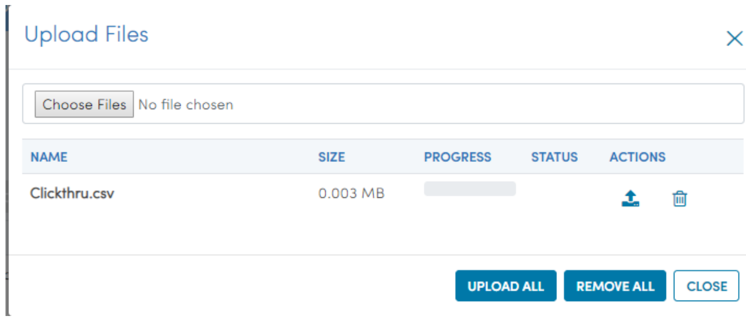
When you open your application, all existing datasets specific to the application are displayed in the Datsets tab.

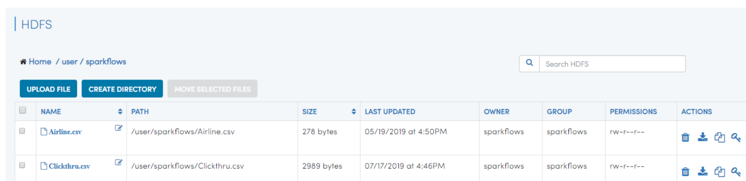
Click on "Create" and choose "Dataset" from the dropdown.

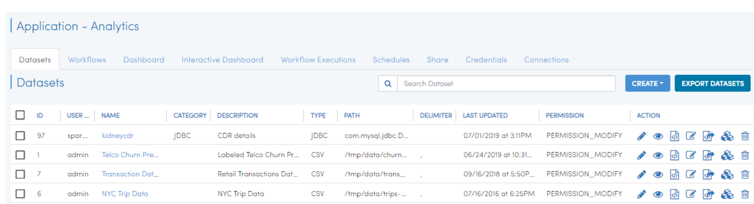
In the pop-up choose "CSV" and then click "OK".

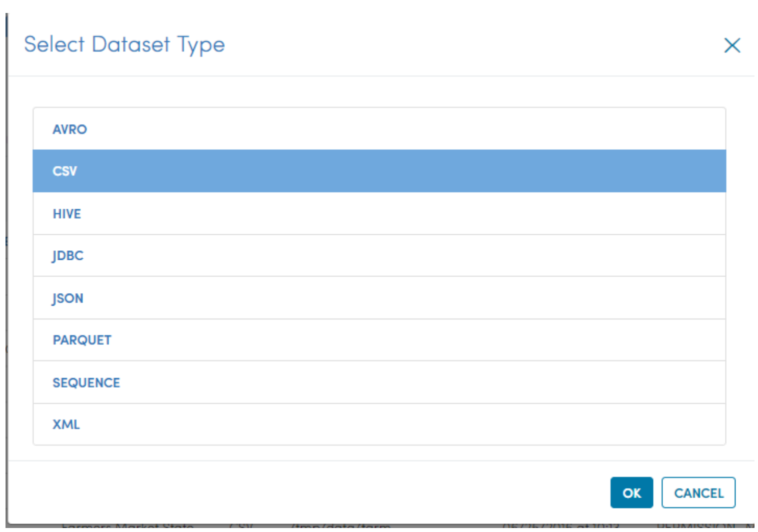
Fill in the required fields as below.

- *Name* : Name of the new dataset
- *Description* : Description of the new dataset

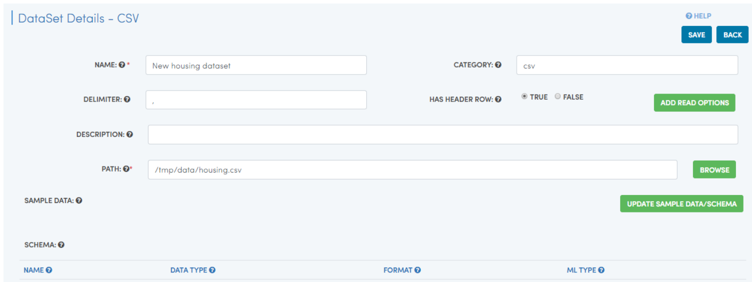

Upload Files
Choose Files
No file chosen
NAME
SIZE
PROGRESS
STATUS
ACTIONS
Clickthru.csv
0.003 MB
UPLOAD ALL
REMOVE ALL
CLOSE


HDFS
Home / user / sparkflows
UPLOAD FILE
CREATE DIRECTORY
NAME
PATH
SIZE
LAST UPDATED
OWNER
GROUP
PERMISSIONS
ACTIONS
/user/sparkflows/Airline.csv
278 bytes
05/19/2019 at 4:50PM
sparkflows
sparkflows
/user/sparkflows/Clickthru.csv
2989 bytes
07/17/2019 at 4:46PM
sparkflows
sparkflows


Application - Analytics
Datasets
Workflows
Dashboard
Interactive Dashboard
Workflow Executions
Schedules
Share
Credentials
Connections
Datasets
CREATE
EXPORT DATASETS

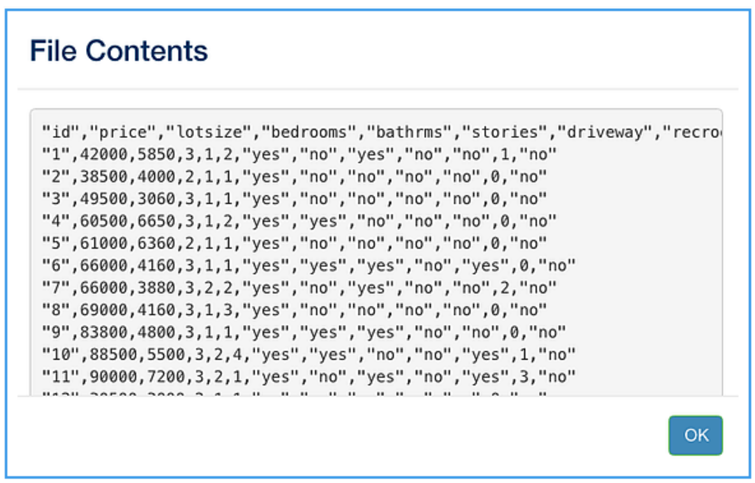

Select Dataset Type
AVRO
CSV
HIVE
JDBC
JSON
PARQUET
SEQUENCE
XML
OK
CANCEL

- *Has Header Row* : Indicate whether the dataset has a header row specifying the name of the columns or not
- *Delimiter* : Indicates the delimiter to be used between the fields in the data
- *Path* : Path for the location of the file or directory containing the data files for the dataset




Now click on "Update dataset/schema" to update the schema of the dataset. Sample data for the dataset will be displayed followed by the schema.

In the example below, a dataset is created from a housing.csv file. It is a comma separated file with a header row specifying the names of the various columns.




If the data file did not have a header row, Fire Insights will give standard column names of "C0, C1" etc.

You can update the column names in the schema based on your data.


Now click "Save' to save the new dataset and you are ready to use it in your workflows.

### 7.1.4 Step 4 : Create Workflow

After you have created the datasets, you can start building workflows to process them.

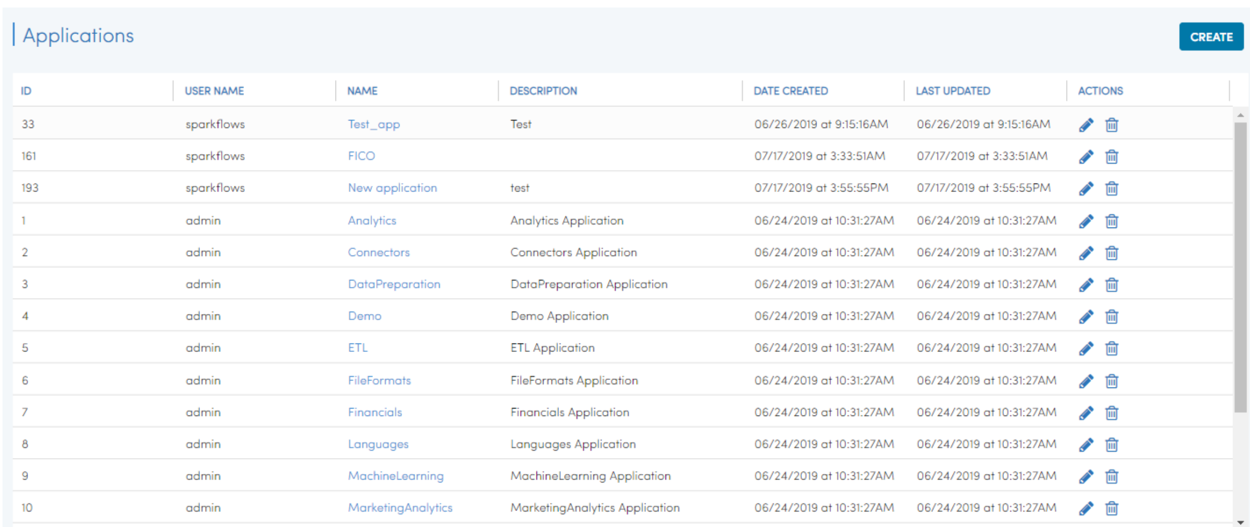
A typical workflow takes one or more dataset, cleans them and joins them, and creates an enriched dataset. After the enriched dataset is created, you can add additional processors to build machine learning models.

At a high level,creating a workflow involves the following steps:

- *Open the Application where you want to create your workflow*
- *Click "Workflows" tab*
- *Create empty workflow*
- *Add processors*
- *Save workflow*

#### Application

Open the application where you want to create your new workflow.



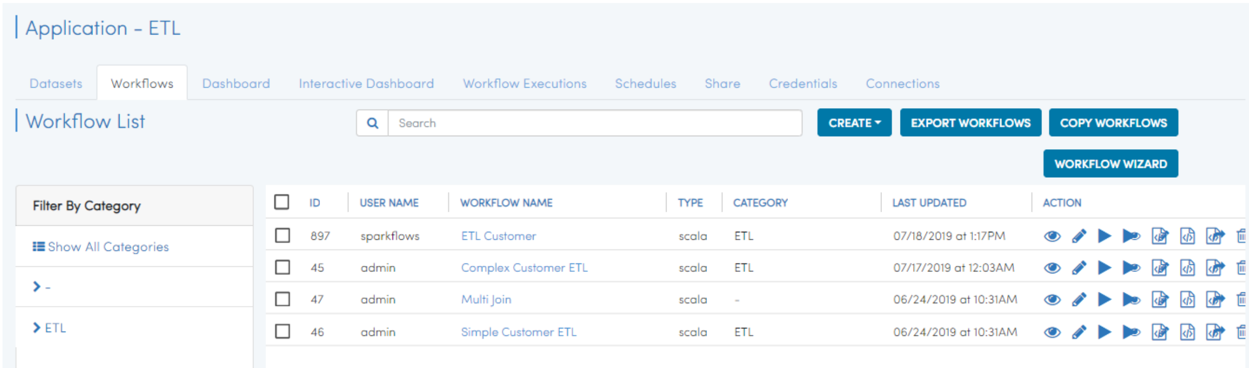

#### Workflows Tab

Click "Workflows" tab to view the list of workflows already in the application. The workflow list will be empty if no workflows have been created earlier.

#### Create Empty Workflow

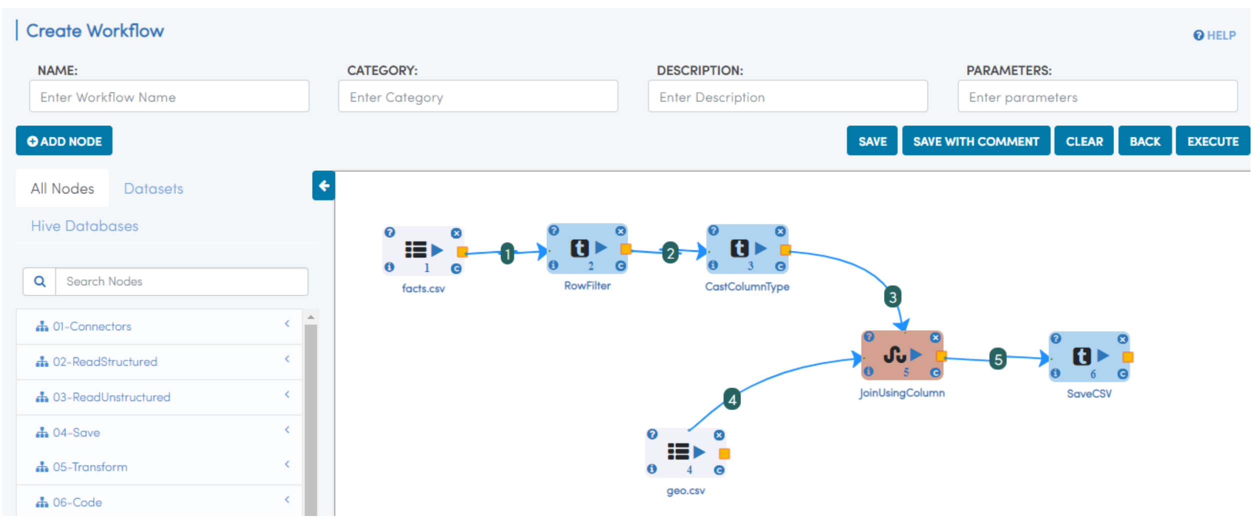
Click "Create" button and choose the type of workflow you want to create. In the "Create Workflow" page, enter a name, category and description of the workflow. Category is used to group various workflows. For instance, if you have several workflows for customer reports, you can group them by specifying "Customer Reports" category.

Click "Save" to save the empty workflow.




### Add Processors

After you have saved the empty workflow, you can start adding processors to process the datasets that you had defined earlier. Click on the processors on the left hand side pane. This will make the processor appear on the workflow canvas. Add other procesors,configure and connect them as needed. Two processors can be connected by clicking on the yellow box in the first processor and dragging it to the second processor.




### Save Workflow

Once you are satisfied with your workflow, save the workflow by clicking on 'Save' button.

Each time the workflow is saved, a new version of workflow is created.

## 7.1.5 Step 5 : Execute Workflow

After you have created a workflow, it is time to execute it and view the results.

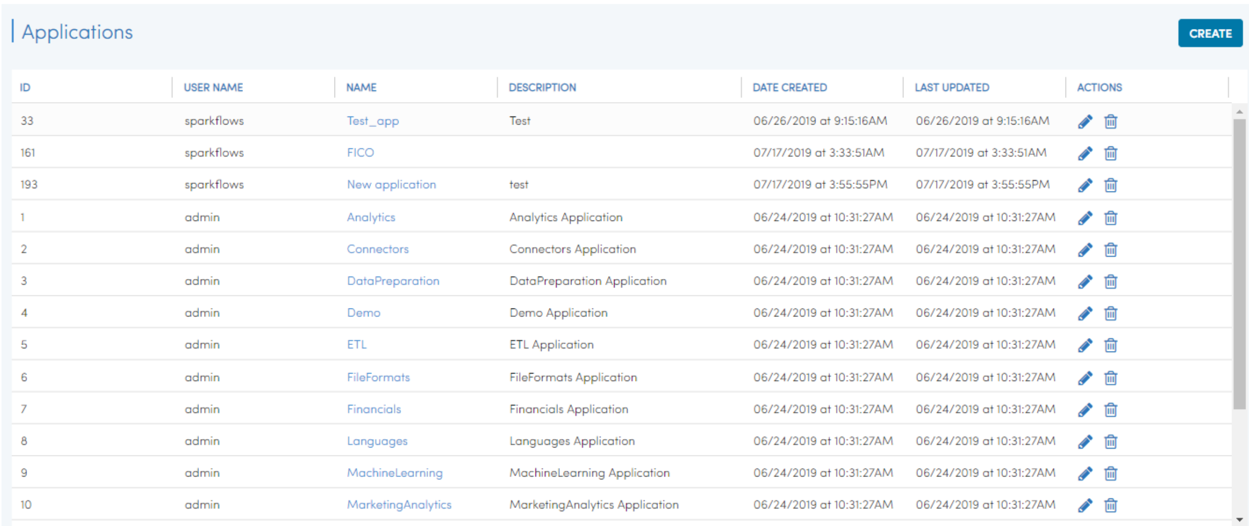
Executing a workflow involves the following steps:

- *Go to Application page where you want to execute the workflow*
- *Click "Workflows" tab*
- *Click on the play button*

- *Specify parameter(if any)*
- *Click on Execute*

### Application page

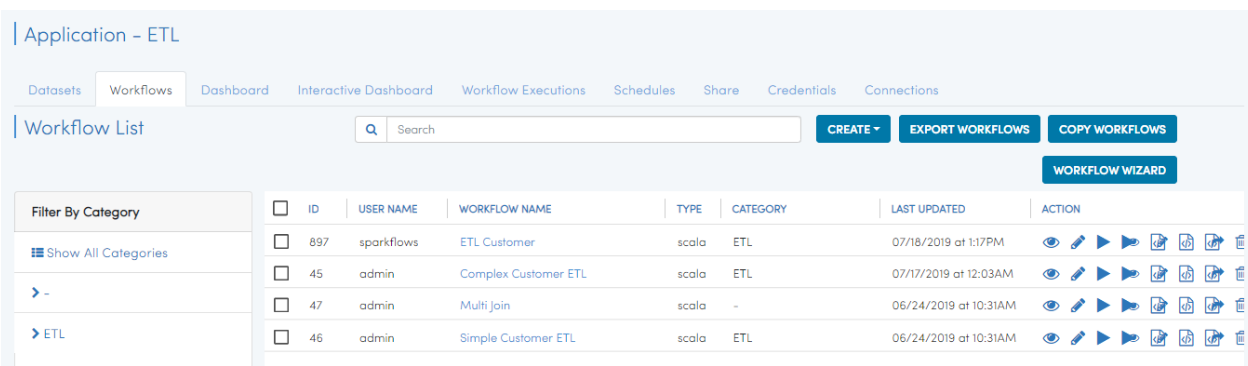
Open the application where you have created the workflow to be executed.




| ID | USER NAME | NAME | DESCRIPTION | DATE CREATED | LAST UPDATED | ACTIONS |
|---|---|---|---|---|---|---|
| 33 | sparkflows | Test_app | Test | 06/26/2019 at 9:15:16AM | 06/26/2019 at 9:15:16AM | |
| 161 | sparkflows | FICO | | 07/17/2019 at 3:33:51AM | 07/17/2019 at 3:33:51AM | |
| 193 | sparkflows | New application | test | 07/17/2019 at 3:55:55PM | 07/17/2019 at 3:55:55PM | |
| 1 | admin | Analytics | Analytics Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |
| 2 | admin | Connectors | Connectors Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |
| 3 | admin | DataPreparation | DataPreparation Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |
| 4 | admin | Demo | Demo Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |
| 5 | admin | ETL | ETL Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |
| 6 | admin | FileFormats | FileFormats Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |
| 7 | admin | Financials | Financials Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |
| 8 | admin | Languages | Languages Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |
| 9 | admin | MachineLearning | MachineLearning Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |
| 10 | admin | MarketingAnalytics | MarketingAnalytics Application | 06/24/2019 at 10:31:27AM | 06/24/2019 at 10:31:27AM | |

### Workflows

Click "Workflows" tab to view the list of workflows in the application.




| ID | USER NAME | WORKFLOW NAME | TYPE | CATEGORY | LAST UPDATED | ACTION |
|---|---|---|---|---|---|---|
| 897 | sparkflows | ETL Customer | scala | ETL | 07/18/2019 at 1:17PM | |
| 45 | admin | Complex Customer ETL | scala | ETL | 07/17/2019 at 12:03AM | |
| 47 | admin | Multi Join | scala | - | 06/24/2019 at 10:31AM | |
| 46 | admin | Simple Customer ETL | scala | ETL | 06/24/2019 at 10:31AM | |

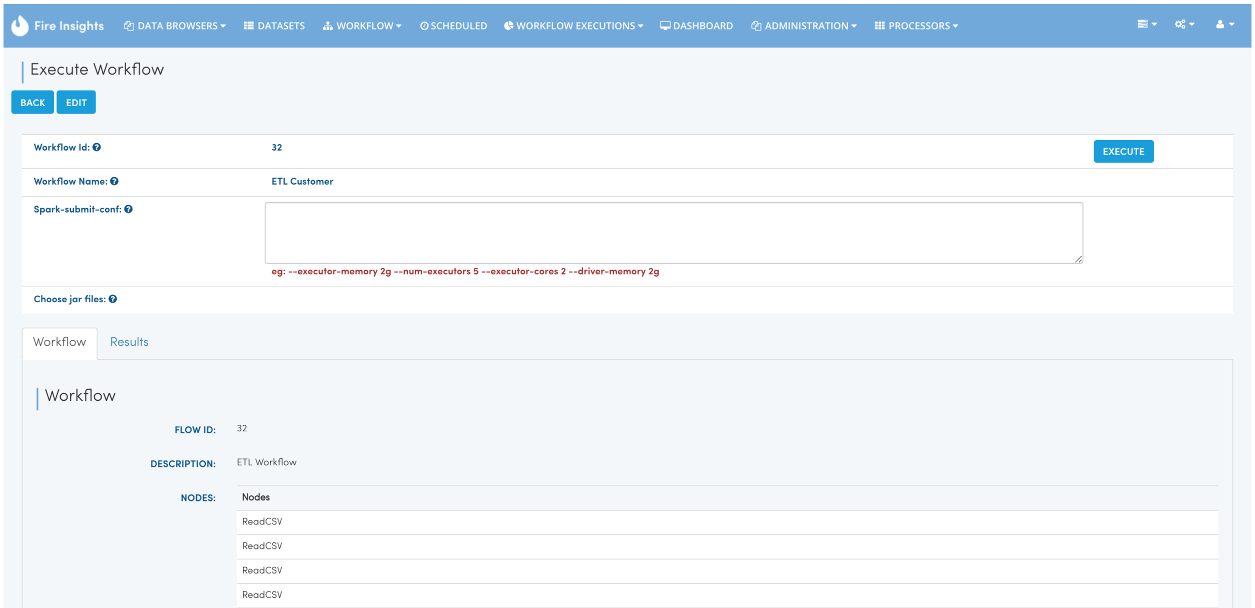
### Click on the Play Button

Against each workflow there are a list of icons under "Actions" column for performing various actions on a specific workflow.

Click "Play" icon under "actions" column to execute the workflow.

### Execute workflow page

Specify any paramters for your workflow.




### Execute Workflow

Once you have specified the parameters, click on "Execute" button. The results of execution are streamed back into your browser.

## 7.1.6 Step 6 : Create Dashboard

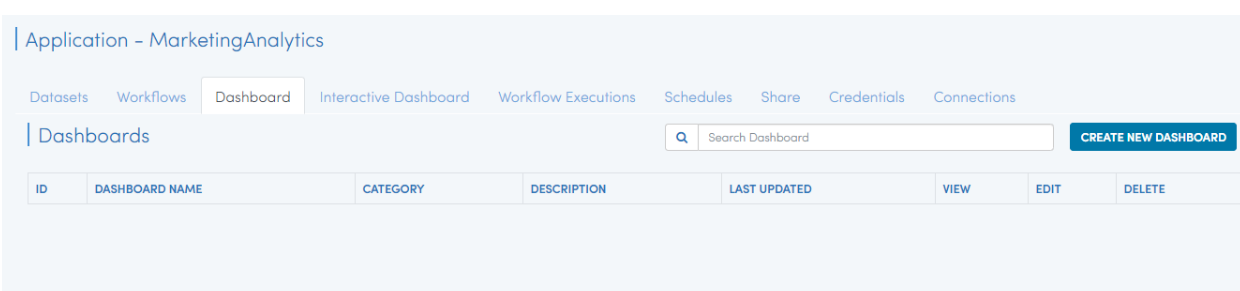
Dashboards allow you to display the output of multiple workflows in one place.

The steps involved in creating a dashboard are:

- *Go to Dashboard tab*
- *Click on Create New Dashboard*
- *Drag and drop selected Nodes from the workflows into the Dashboard canvas*
- *Save the Dashboard*

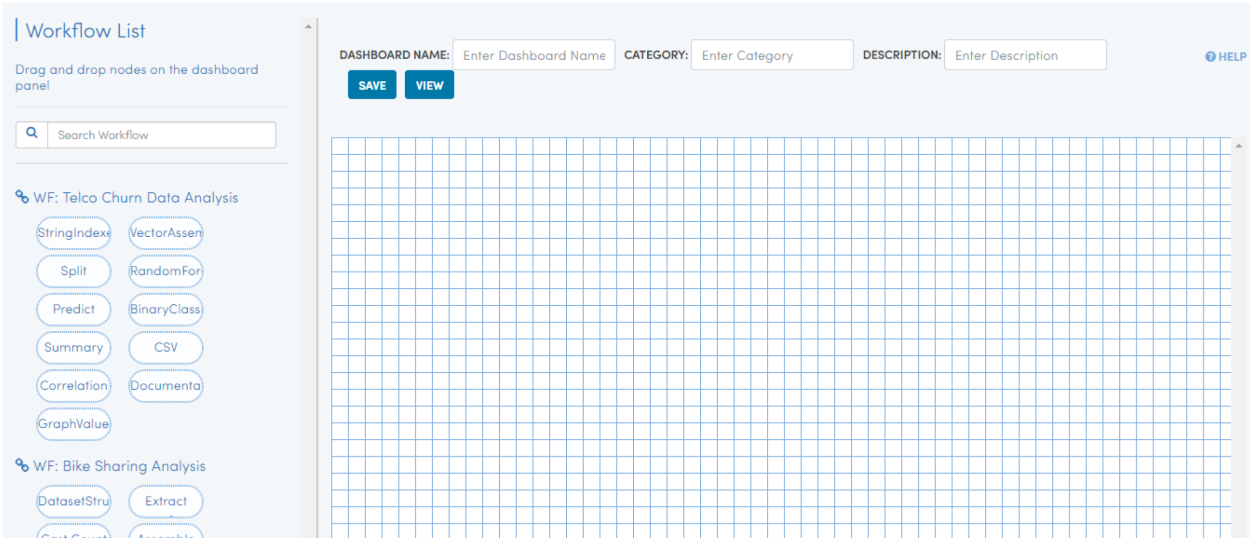
### Dashboards

Selecting Dashboard tab will take to Dashboard page.



### Create Dashboard

This would open up the Dashboard Designer Page.




### Name Dashboard

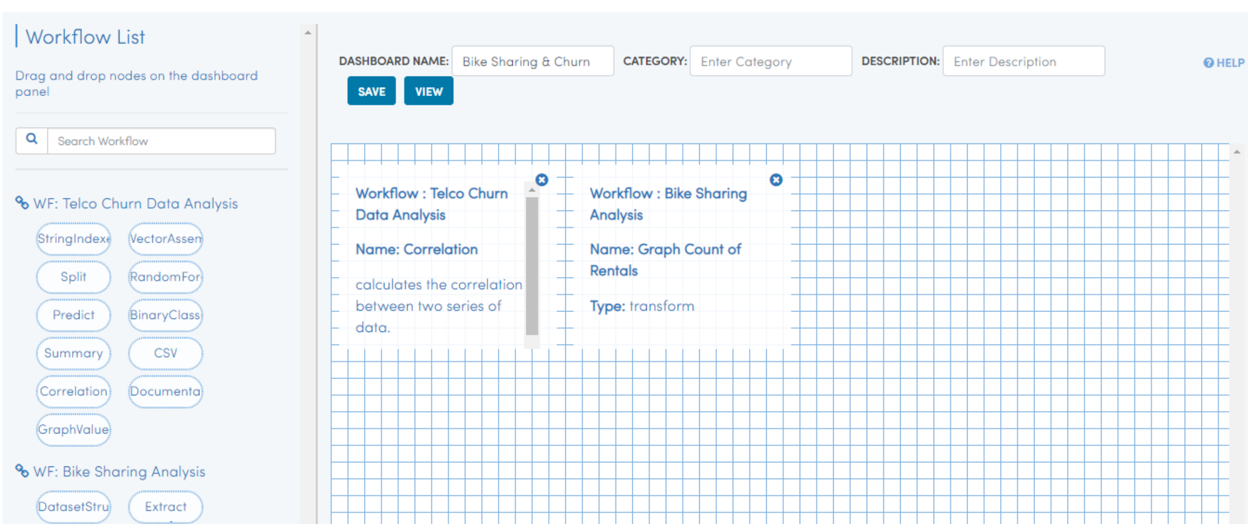
Give a name to your dashboard. You can also add a description for the new dashboard.

### Build Dashboard

On the left hand side of the Dashboard Designer, the list of workflows would show up. With each workflow, the nodes inside the workflow would be displayed.

Nodes inside the workflow can be dragged and dropped onto the dashboard to make them part of the dashboard.

In the dashboard below we have added two nodes to the dashboard.



### Save Dashboard

Finally save the dashboard.

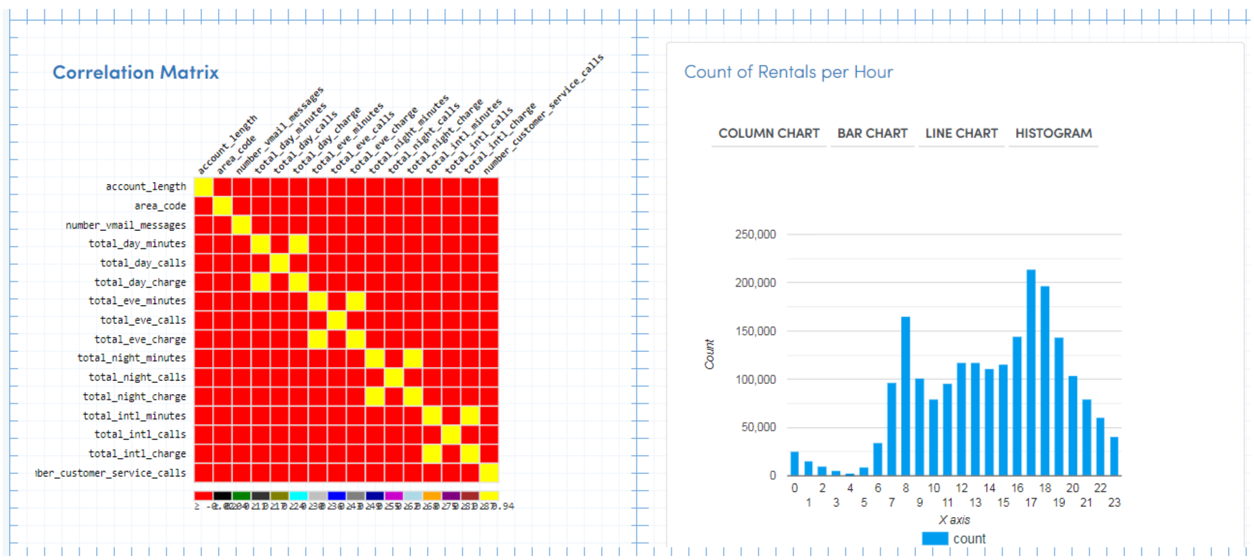
In order to view the dashboard, click on the 'View' button.

### View Dashboard

Click on the 'View' button to view the dashboard.

The dashboard shows the content from the latest execution of the workflow.

If the workflow has never been executed, the dashboard would not show anything.



CHAPTER 8

---

# User Guide

---

## 8.1 User Guide

### 8.1.1 Datasets

Fire Insights allows you to define your DataSets. These DataSets are then used in Workflows as data sources. DataSet sources can be local file system when running in local mode, or HDFS & HIVE when running on a Spark cluster.

**Schema**

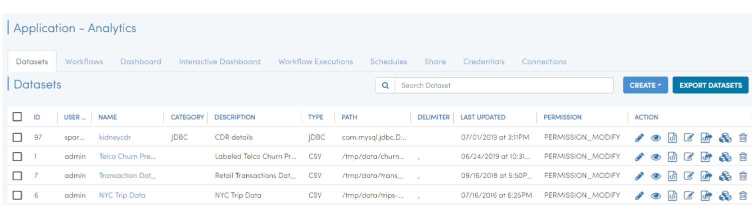
- DataSets have Schema defined for them. This allows Fire Insights to read and create a DataFrame out of it. The DataFrame is then used for transforms, machine learning etc.

**File formats**

- Sparkflows supports various File formats and is able to infer the schema. These include `CSV/TSV`, `Parquet`, `Avro`, `JSON`, `XML` files.
- Sparkflows also supports creating datasets from `HIVE` tables. This is not necessary as in the Workflows HIVE Processors can be directly connected to specific HIVE tables (instead of creating a Dataset in Fire for them).

**Dataset Listing Page**

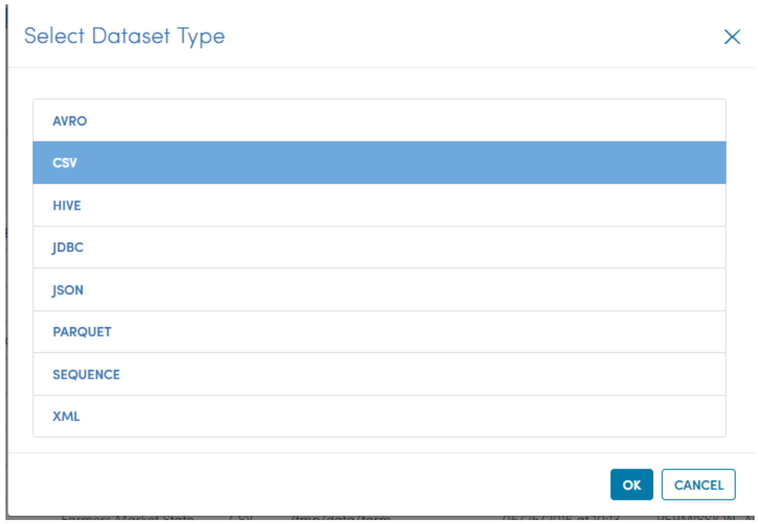
When you open any application, all existing Datasets specific to the application are displayed in the Datasets tab.



### Creating New Datasets

You can define a New Dataset by clicking on the `Create Dataset` button in the Dataset page.

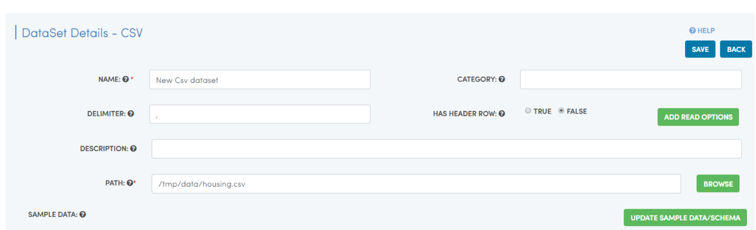
It will bring up the dialog box below. Select the format of the file for which the new Dataset is being created.




### Entering Field Details

Below are the details of the fields in the `Create Dataset` page:

- **NAME** : Name of the New Dataset we are creating.
- **DESCRIPTION** : Description of the New Dataset.
- **HAS HEADER ROW** : This is used for CSV/TSV files. It indicates whether the dataset has a header row specifying the name of the columns or not.
- **DELIMITER** : Delimiter field is also used for CSV/TSV files. It indicates the delimiter to be used between the fields in the data.
- **PATH** : It defines the location of the file or directory containing the data files for the Dataset. It can either point to a single file, or to a directory containing a set of files. All the files have to have the same schema.




### Updating the Schema of the Dataset

You can update the Schema of the Dataset by clicking on `Update Sample Data/Schema`. It would display sample data for the dataset followed by the Schema inferred by Fire Insights.

In this example, the data file did not have a header row. So Fire gave it standard column names of `C0, C1` etc.

You can update the column names in the schema based on your data.

### Saving the New Dataset

Click on the `Save` button to save the New Dataset created.

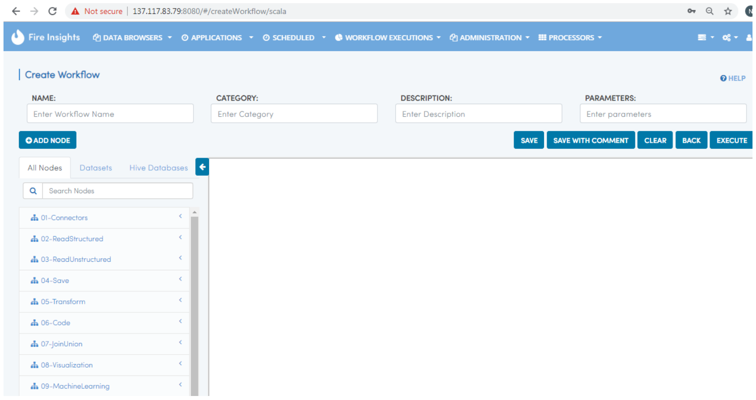
## 8.1.2 Workflows

### Creating Workflows

Fire Insights enables users to define end-to-end workflows for data pipelining leveraging pre-packaged nodes for common ETL and Machine Learning models. Workflows are then saved and executed to produce results. Sparkflows provides a a very intuitive and user friendly editor to achieve the same.

### Define New Workflow

Click on 'Create New Workflow' for creating a New Workflow, It supports two engines - spark & pyspark. It will open the Workflow Editor where the workflow can be created.



### Adding New Nodes to the Workflow

- Workflows editor has a list of Nodes menu on the LHS. Clicking on any of the Nodes creates it in the workspace.

### Creating Edges

- Nodes can be connected by edges.
- Click on the orange box and drag to the next node to create an edge between them.

### Deleting Edges

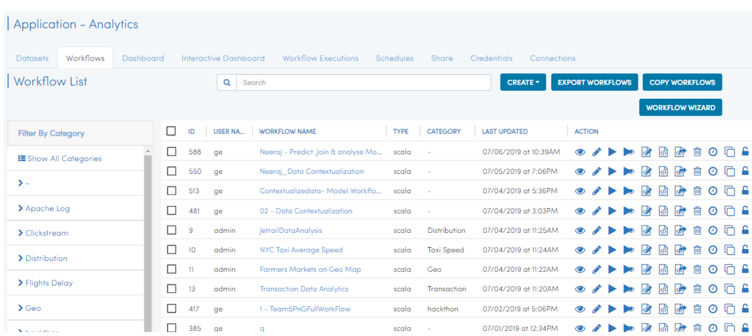
- Edges can be deleted by double clicking on them.

### Saving Workflow

- Give the workflow a name.
- Click on the Save button to create the new workflow.

### View Workflows

You can view the workflows by going to the Workflows Page inside specific applications.



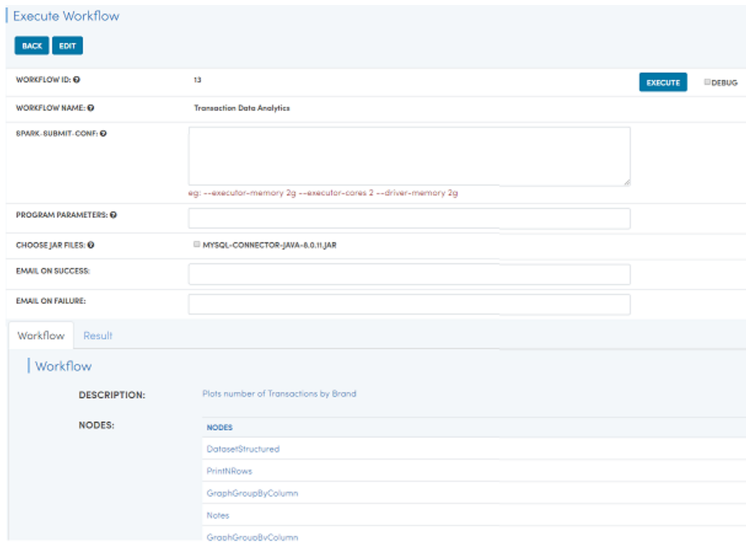

### Executing Workflows

Fire Workflows can be executed in the following ways:

- **Interactively within the User Interface**
- **Submitting the workflows using spark-submit through the command line**
- **Scheduling for execution with your scheduler of choice**

### Interactively within the User Interface

Workflows can be executed from the browser by going into the Execute page of the workflow.




### Executing Workflows with spark-submit

Workflows are saved as text files in JSON format. Workflows can be submitted to be run on the cluster with spark-submit:

```
spark-submit    --class    fire.execute.WorkflowExecuteFromFile    --master yarn    --deploy-mode client    --executor-memory 1G    --num-executors 1    --executor-cores 1       fire-core-1.4.2-jar-with-dependencies.jar       --postback-url http://<machine>:8080/messageFromSparkJob        --job-id 1         --workflow-file     kmeans.wf
```

In the above:

For providing extra variables to the workflow, the following parameters can be added to spark-submit:

```
--var name1=value1   --var name2=value2    --var name3=value3
```

In the workflow, these variables can be used with $name1 $name2 Specific nodes make use of the variables by substituting $name with the value provided for the name.

For running the workflow in debug mode, add the following parameters:

```
--debug true
```

### Workflow JSON

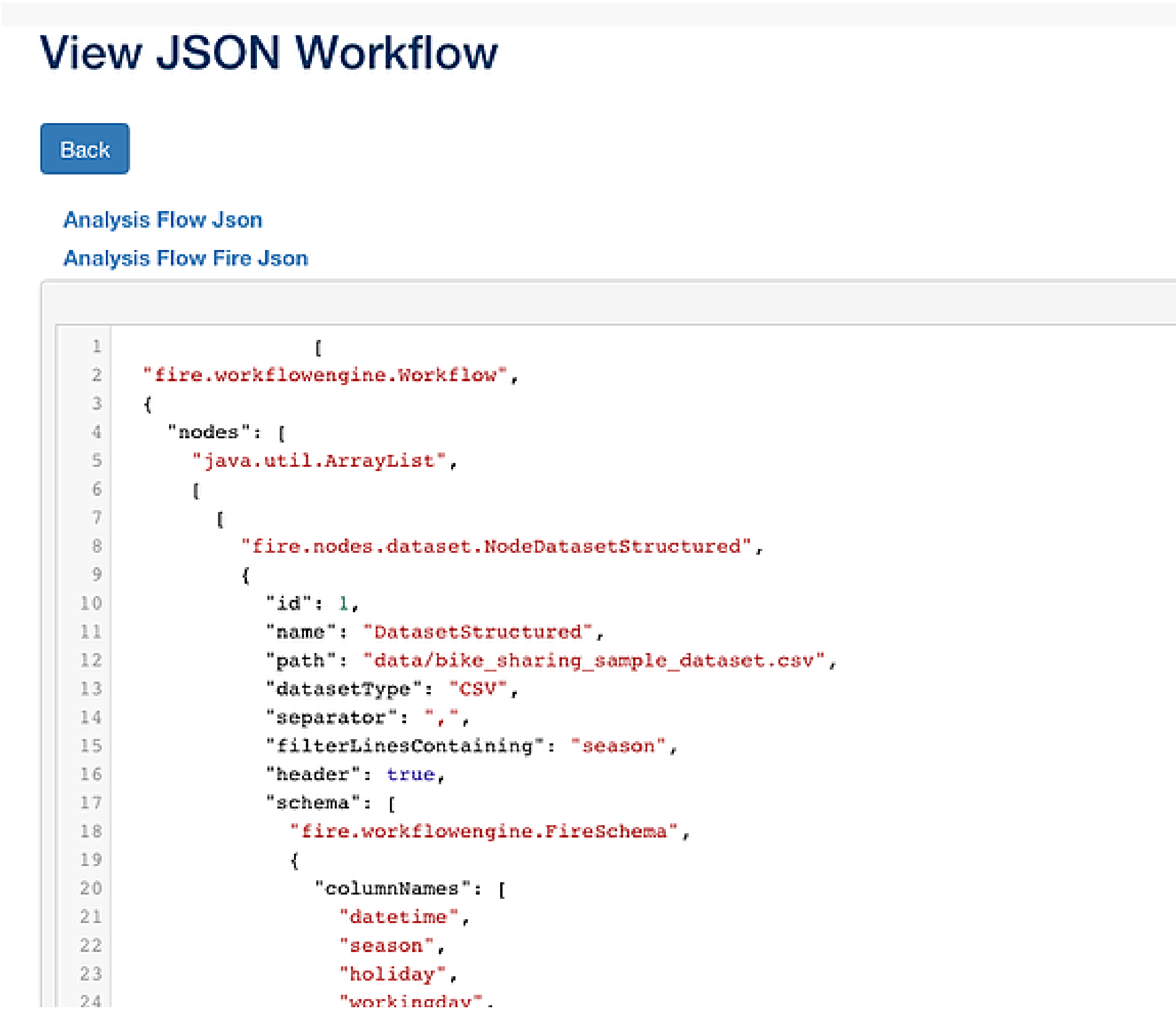
In Sparkflows, workflows are saved as JSON Strings.

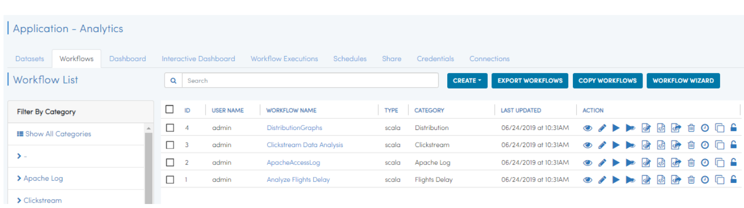
The View JSON Workflow page of the Workflow displays the JSON representations of the workflow.

### Scheduling Workflow execution with Scheduler of choice

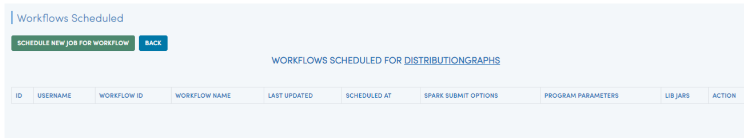
Since Fire workflows can be submitted with spark-submit, you can use your scheduler of choice for scheduling the execution of the workflows.

- Click on Schedule Button of Workflow we want to schedule

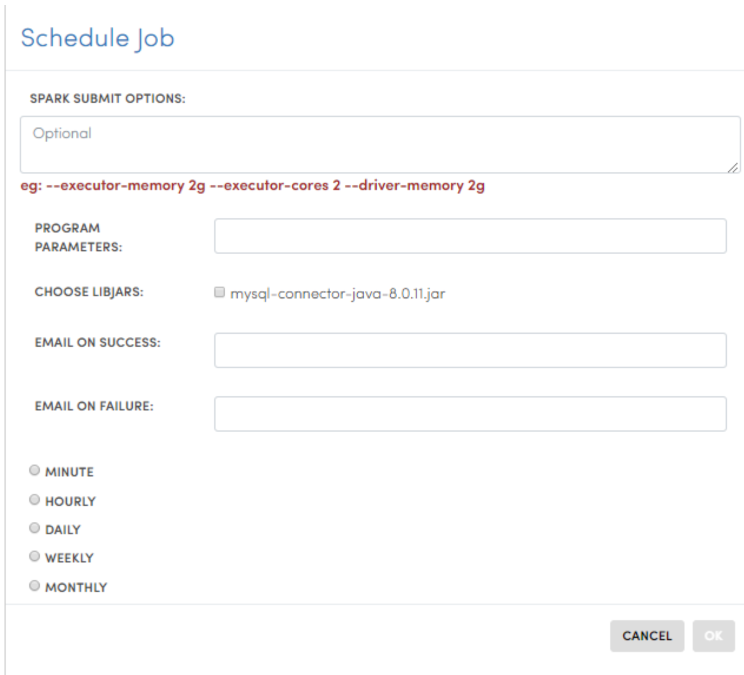

View JSON Workflow
Back
Analysis Flow Json
Analysis Flow Fire Json
[
"fire.workflowengine.Workflow",
{
"nodes": [
"java.util.ArrayList",
[
[
"fire.nodes.dataset.NodeDatasetStructured",
{
"id": 1,
"name": "DatasetStructured",
"path": "data/bike_sharing_sample_dataset.csv",
"datasetType": "CSV",
"separator": ",",
"filterLinesContaining": "season",
"header": true,
"schema": [
"fire.workflowengine.FireSchema",
{
"columnNames": [
"datetime",
"season",
"holiday",


Application - Analytics
Workflow List
CREATE
EXPORT WORKFLOWS
COPY WORKFLOWS
WORKFLOW WIZARD
Filter By Category

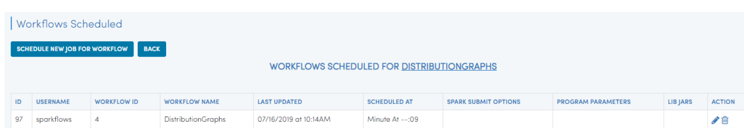
- Click on Tab Schedule New Job for Workflow




  - Update the scheduled timing & email notifications after success & failure of workflow as per our requirments.




- Click on OK to save the changes.




### Debugging Workflows

Many times it is helpful to be able to debug the workflows. One easy way is to check the debug checkbox in the UI when executing the workflow.

Running in debug mode does a few things:

- Performs a count() after executing each Processor. This makes it easier to track errors. It takes out Sparkflows lazy execution from the picture.
- Displays the number of records processed at each stage.
- Display more information, for each SQL etc. which are being executed.

### Passing Parameters to Workflows

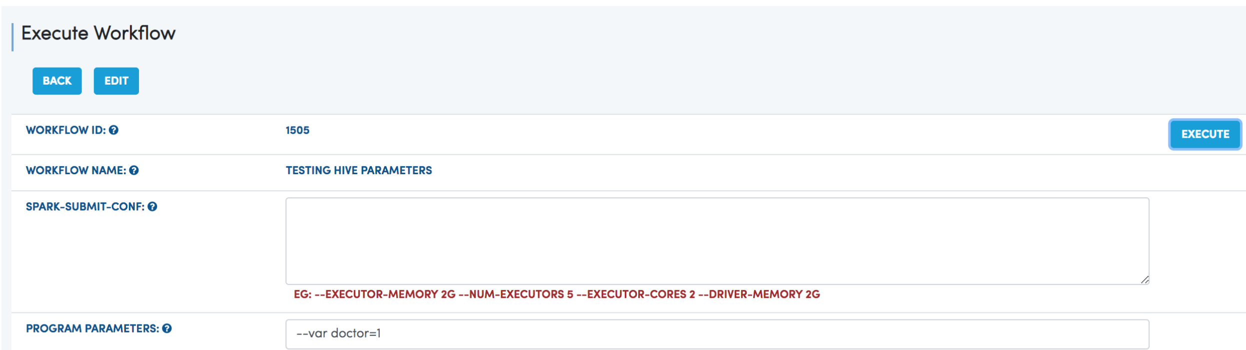
Fire Insights runs the spark jobs with `spark-submit`. It takes in the workflow JSON as a parameter. There are multiple ways to pass extra parameters to the workflow. If the same parameter is specified multiple times, the order of precendence in which they are applied is as shown below:

- Through Program Parameters passed during Workflow Execution
- By specifying the parameters in the Workflow Editor
- Through a Parameter Processor in the workflow
- A Node creating a variable during execution time

### Through Program Parameters in Fire during Workflow Execution

Key/Value pairs can be passed to Fire during Workflow Execution. An example of it is `--var doctor=1` These Key/Value pairs would override any Key/Value pair passed through the Parameter Processor in the workflow.

Below is a screenshot:




### By specifying the parameters in the Workflow Editor

Parameters can be specified in the Workflow Editor. They can be specified in the following format:

They can be passed with `--var name1=value1 --var name2=value2`

### Through a Parameter Processor in the Workflow

A Parameter Processor can be added to the workflow. It allows passing key/value pairs to the workflow.

### A Processor creating a variable during execution time

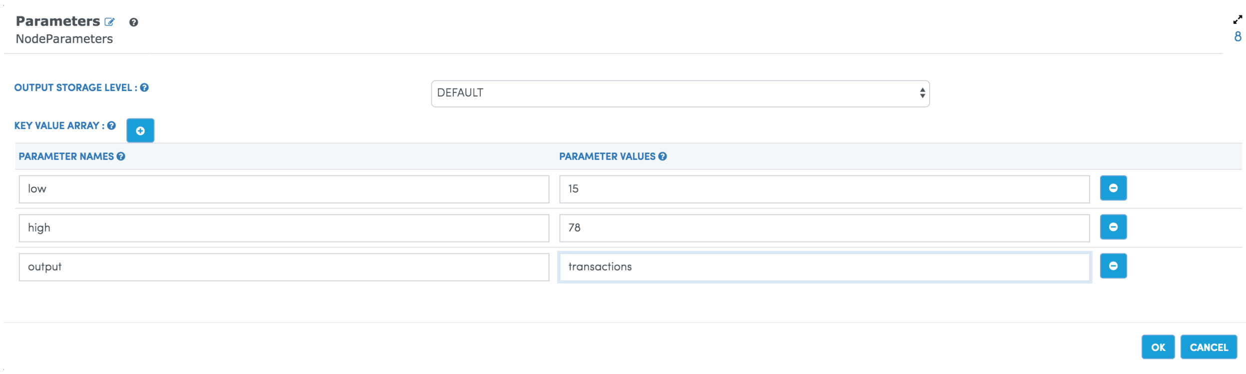
A Processor can also create a parameter during run time. A Processor creates a new variable and puts it into the JobContext.

jobContext.nodeGeneratedParameters.put(variable, “”+count);

This parameter can then be later used by another Processor.

For example `NodeCount` puts the count of records into a variable in the Job Context.

`NodeAssert` uses this variable when evaluating expressions.




### Through `--var` parameters with spark-submit

Fire Insights workflow can also be directly executed on the cluster with spark-submit.

In this case, extra parameters can be passed with `--var`:

```
spark-submit    --class fire.execute.WorkflowExecuteFromFile    --master yarn    --
↪deploy-mode client   fire-core-3.1.0-jar-with-dependencies.jar    --postback-url␣
↪http://<machine>:8080 --job-id 1       --workflow-file kmeans.wf    --var␣
↪name1=value1  --var  name2=value2
```

In the workflow, these parameters can be used with `$name1 $name2`

Specific nodes make use of the parameters by **substituting $name with the value** provided for the name.

An **example** would be : `--var id=3`

When specifying the expression in the RowFilter Node we can use : `id > $id`

In the above **$id** would be replaced with **3**.

### Specifying `--var` parameters for all in Sparkflows User Interface

Sparkflows also allows specifying the **–var** parameters to be passed to all the jobs submitted through Sparkflows. Below is the screen under Administration/Configuration.

In the above, **app.vars** parameter allows specifying a space separated list of name=value pairs.

Each of these are passed to the jobs submitted by Sparkflows with `--var name=value`

### Workflow Execution Results

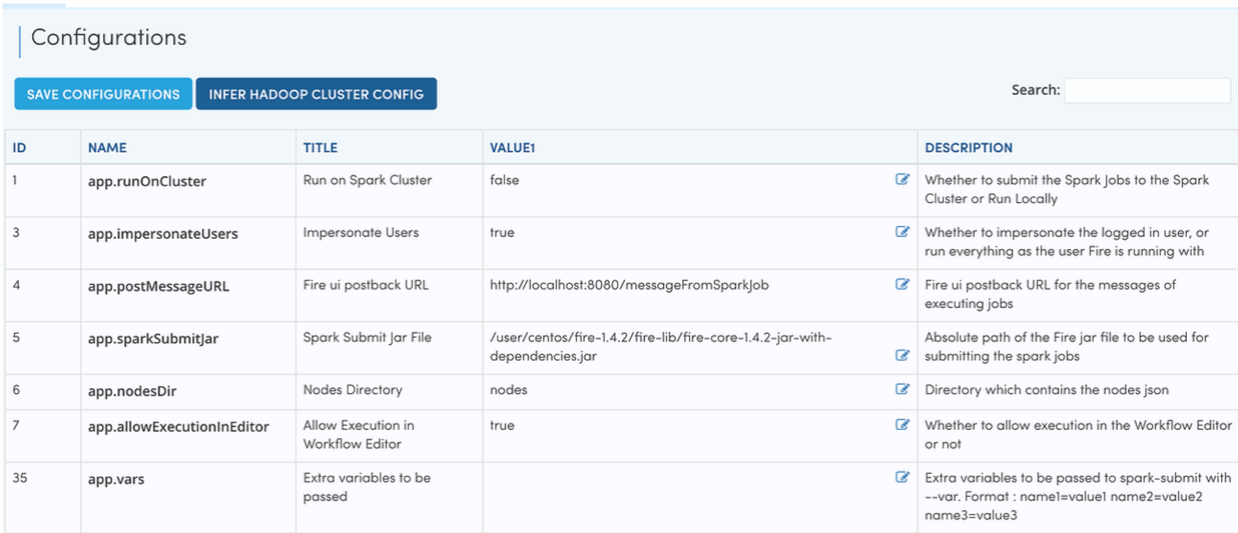
The results of Workflow Execution are streamed into the Browser as they are executed and displayed in rich Format. A workflow may run for a very long time.

The results of past executions can also be viewed in the Workflow Executions page.

## 8.1.3 Visualizations

### Visualization Processors

There are a number of Nodes/Processors in Fire which produce rich visualizations.



Configurations

SAVE CONFIGURATIONS | INFER HADOOP CLUSTER CONFIG | Search:

| ID | NAME | TITLE | VALUE1 | DESCRIPTION |
|---|---|---|---|---|
| 1 | app.runOnCluster | Run on Spark Cluster | false | Whether to submit the Spark Jobs to the Spark Cluster or Run Locally |
| 3 | app.impersonateUsers | Impersonate Users | true | Whether to impersonate the logged in user, or run everything as the user Fire is running with |
| 4 | app.postMessageURL | Fire ui postback URL | http://localhost:8080/messageFromSparkJob | Fire ui postback URL for the messages of executing jobs |
| 5 | app.sparkSubmitJar | Spark Submit Jar File | /user/centos/fire-1.4.2/fire-lib/fire-core-1.4.2-jar-with-dependencies.jar | Absolute path of the Fire jar file to be used for submitting the spark jobs |
| 6 | app.nodesDir | Nodes Directory | nodes | Directory which contains the nodes json |
| 7 | app.allowExecutionInEditor | Allow Execution in Workflow Editor | true | Whether to allow execution in the Workflow Editor or not |
| 35 | app.vars | Extra variables to be passed | | Extra variables to be passed to spark-submit with --var. Format : name1=value1 name2=value2 name3=value3 |

Workflow Executions

Search:

| Id | Analysisflow_ID | USER ID | Name | Category | Description | Start Time | Last Updated | View Models | View Result |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 31 | 1 | Housing | Machine Learning | Housing Price Prediction | 07/21/2016 at 11:51:11AM | 07/21/2016 at 11:51:14AM | View Model | View Execution Result |
| 2 | 52 | 1 | Ratings | Recommendations | Movie Lens Recommendations | 07/21/2016 at 11:51:26AM | 07/21/2016 at 11:51:34AM | View Model | View Execution Result |
| 3 | 5 | 1 | BikeSharing_Analysis | Machine Learning | BikeSharing Analysis | 07/21/2016 at 11:51:42AM | 07/21/2016 at 11:51:50AM | View Model | View Execution Result |
| 4 | 31 | 1 | Housing | Machine Learning | Housing Price Prediction | 07/21/2016 at 12:09:41PM | 07/21/2016 at 12:09:42PM | View Model | View Execution Result |
| 5 | 40 | 1 | kmeans | Machine Learning | KMeans | 07/21/2016 at 4:17:07PM | 07/21/2016 at 4:17:08PM | View Model | View Execution Result |
| 6 | 5 | 1 | BikeSharing_Analysis | Machine Learning | BikeSharing Analysis | 07/21/2016 at 5:10:09PM | 07/21/2016 at 5:10:15PM | View Model | View Execution Result |

These Processors can be added to any workflow and are applied to the data.

Visualization Processors include:

- Graph Values
- Geo
- Group by Column
- Weekday Distribution
- Monthly Distribution
- Yearly Distribution
- Heatmaps
- Tables

### Batch Dashboards

Fire allows you to create Dashboards.

Processors in Fire can output data in Tables, Charts, Maps and Simple Strings. Dashboards allow combining the output of various processors into one User Interface.

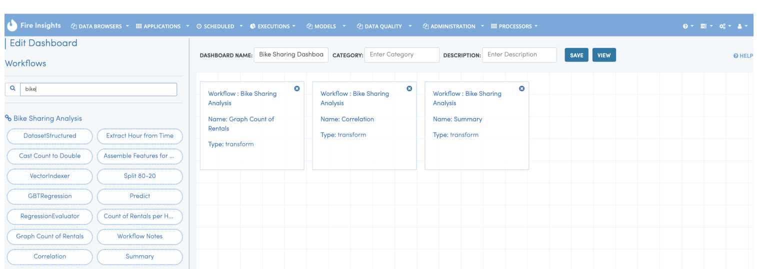
For example we might want to output a chart of number of bike rentals per hour, another by per day and another map displaying the total number of bike rentals per city for the day. Dashboards can combine all these into one view.

### Creating Dashboards

- For creating Dashboards, drag and drop the required processors from the workflows into the Dashboard Canvas.
- When the corresponding workflows are run, the output is stored by Fire into the relational store. These get displayed into the dashboard.

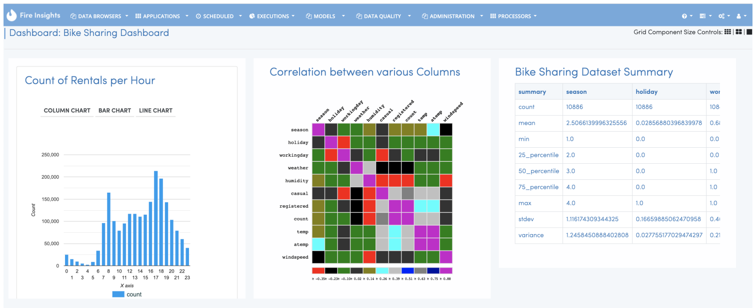
### Editing Dashboards

Editing Dashboards is like creating dashboards, except that you click the edit button to edit the corresponding Dashboard.




### Viewing Dashboards

Once a Dashboard has been created you can view it, by clicking on the View button.




## Streaming Dashboards

- Fire allows you to create Streaming Workflows.
- Streaming workflows have a mini-batch duration - say 30 seconds.
- In this case, the output in the Dashboards get updated every 30 seconds as new data come in.

## Interactive Dashboard

Fire allows you to create interactive Dashboard.

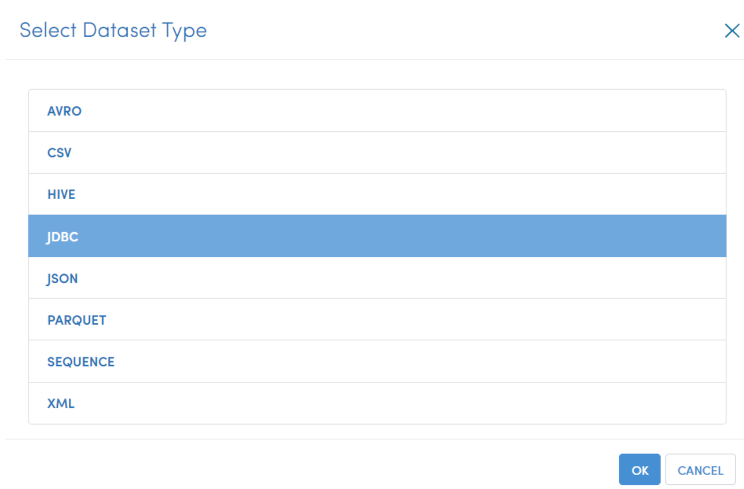
Fire allows us to create New Dataset using JDBC data type from MYSQL DB & use datasets in creating charts & dashboard.

## Creating I-Dashboard

- For creating I-Dashboard, Create JDBC datasets if not available.

You can define a New Dataset by clicking on the `Create Dataset` button in the Dataset page.

It will bring up the dialog box below. Select the format of the file for which the new Dataset is being created.



### Entering Field Details

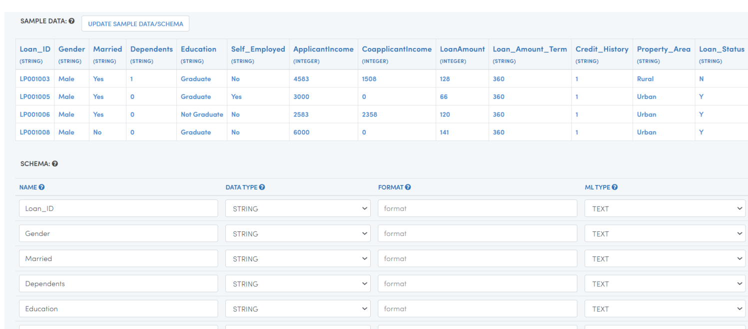
Below are the details of the fields in the `Create Dataset` page:

- **NAME** : Name of the New Dataset we are creating.
- **DESCRIPTION** : Description of the New Dataset.
- **CATEGORY** : category of the New Dataset.
- **JDBC DRIVER** : Enter JDBC DRIVER.
- **JDBC URL** : Enter JDBC URL for MYSQL DB.
- **USER** : username for MYSQL DB.
- **PASSWORD** : password for MYSQL DB.
- **DB** : Database for MYSQL DB.
- **TABLE** : Table for MYSQL.


### Updating the Schema of the Dataset

You can update the Schema of the Dataset by clicking on `Update Sample Data/Schema`. It would display sample data for the dataset followed by the Schema inferred by Fire Insights.

You can update the column names in the schema based on your data.

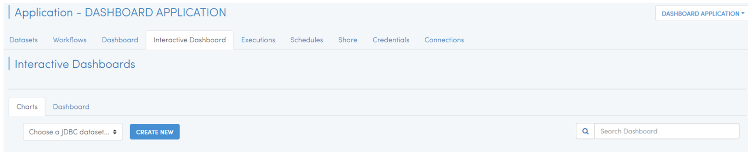



### Saving the New Dataset

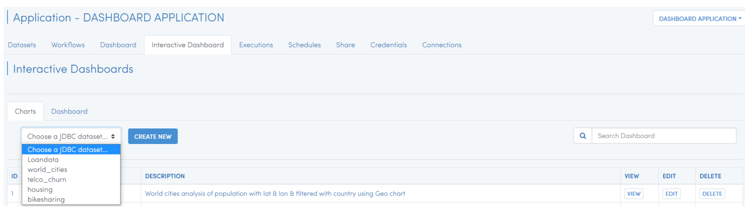
Click on the `Save` button to save the New Dataset created.

## Interactive Dashboard

Click on `Interactive Dashboard` tab in the same application where you have created JDBC Dataset.

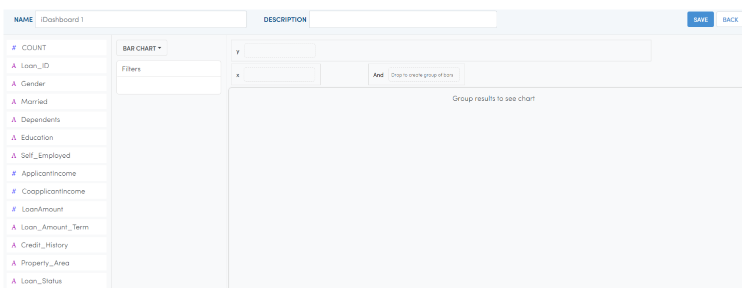



Click on `chart` tab & select Choose a JDBC dataset, there you will find all JDBC datasets created under your application.




Select any JDBC dataset for which you want to create `chart` & select `CREATE NEW`

It will take you to new page, as below




Select the `chart type`, you want to see chart

Selected `Bar chart` & updated column for x & y axis and add some filter

Add NAME, DESCRIPTION & save it

Once you save it, the chart will appear in chart list page

Similarly you can create different chart using specified chart type

Now using existing chart, you can create new dashboard

Select `Dashboard` tab & Click on CREATE DASHBOARD

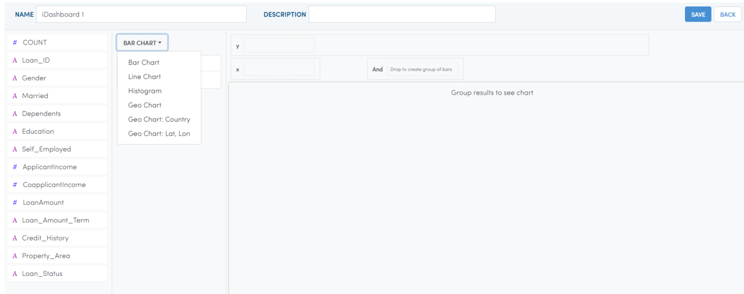
it will take us to New Dashboard page

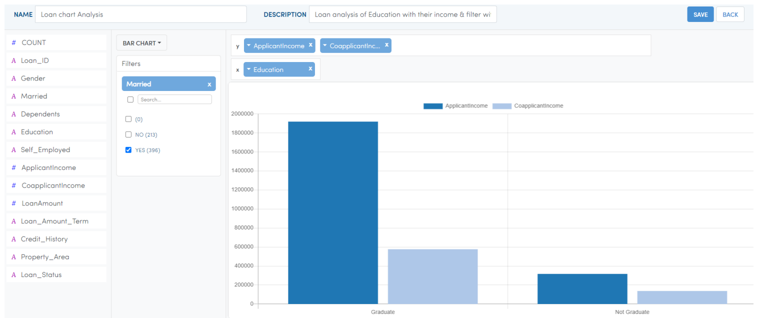
Using drag & drop you need to add chart in canvas, Add NAME, DESCRIPTION & SAVE it.

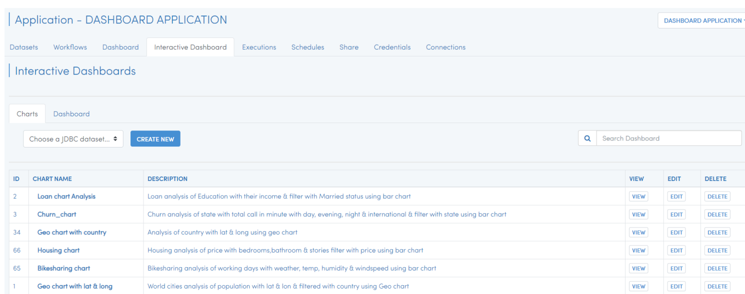
Once the Dashboard got saved successfully, it will show in dashboard list page from where you can view, edit & delete it.

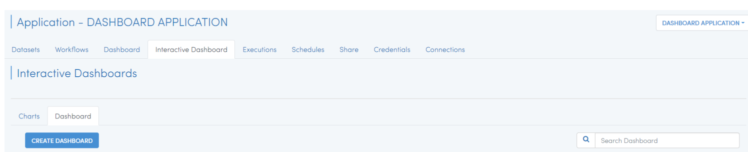
## Exporting Visuals

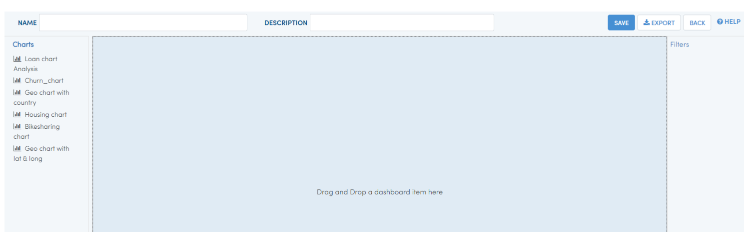
Fire Insights enables you to export the output, dashboards and visuals in various ways.

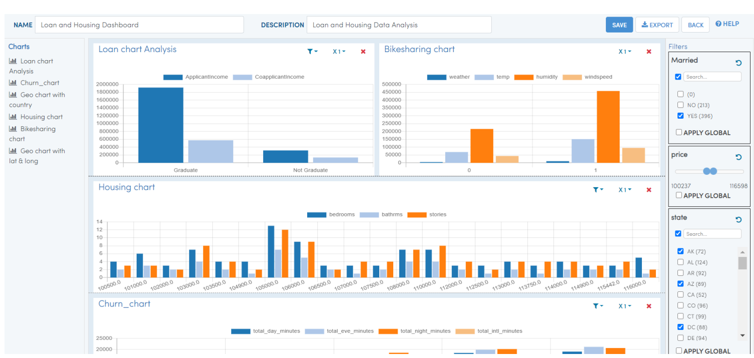

NAME
iDashboard 1
DESCRIPTION
SAVE
BACK
COUNT
Loan_ID
Gender
Married
Dependents
Education
Self_Employed
ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Credit_History
Property_Area
Loan_Status
BAR CHART
Bar Chart
Line Chart
Histogram
Geo Chart
Geo Chart: Country
Geo Chart: Lat, Lon
Group results to see chart

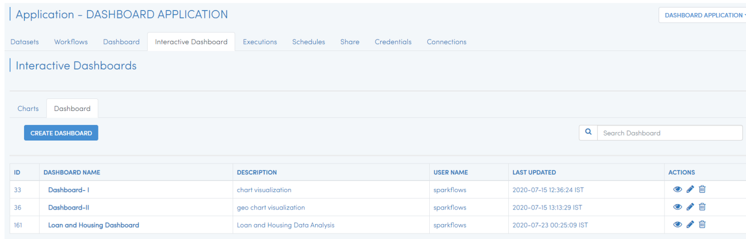

NAME
Loan chart Analysis
DESCRIPTION
Loan analysis of Education with their income & filter wi
SAVE
BACK
BAR CHART
Filters
Married
YES (398)
ApplicantIncome
CoapplicantIncome
Education
Graduate
Not Graduate


Application - DASHBOARD APPLICATION
Datasets
Workflows
Dashboard
Interactive Dashboard
Executions
Schedules
Share
Credentials
Connections
Interactive Dashboards
Charts
Dashboard
Choose a JDBC dataset...
CREATE NEW
Search Dashboard
ID
CHART NAME
DESCRIPTION
VIEW
EDIT
DELETE
Loan chart Analysis
Churn_chart
Geo chart with country
Housing chart
Bikesharing chart
Geo chart with lat & long


Application - DASHBOARD APPLICATION
Datasets
Workflows
Dashboard
Interactive Dashboard
Executions
Schedules
Share
Credentials
Connections
Interactive Dashboards
Charts
Dashboard
CREATE DASHBOARD
Search Dashboard


NAME
DESCRIPTION
SAVE
EXPORT
BACK
HELP
Charts
Filters
Drag and Drop a dashboard item here

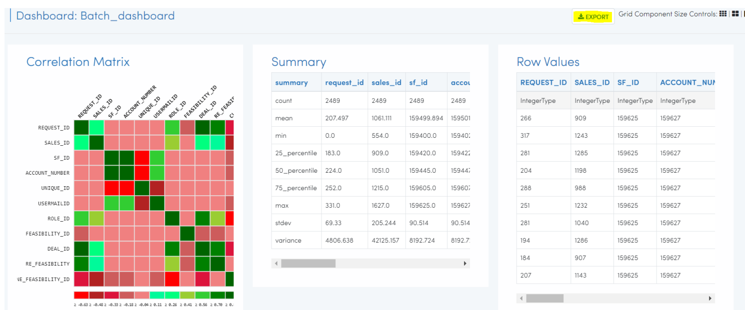

NAME
Loan and Housing Dashboard
DESCRIPTION
Loan and Housing Data Analysis
SAVE
EXPORT
BACK
HELP
Charts
Loan chart Analysis
Bikesharing chart
Housing chart
Churn_chart
Filters
Married
price
state
APPLY GLOBAL



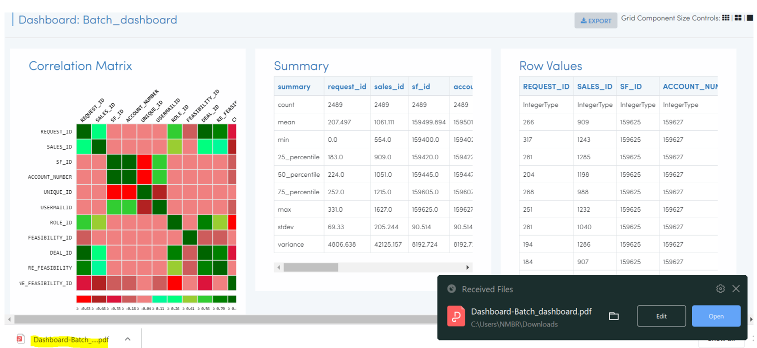

### Exporting dashboard

Since Fire Insights is Browser based end to end, its easy to export the pages as PDF files.

- Go to dashboard under your application where you have created batch dashboard
- On clicking on view option, able to visualize etc. added in that dashboard, there you will have `Export` option, Click on that.




It will Export the whole batch dashboard in pdf format on local machine.




### Exporting output

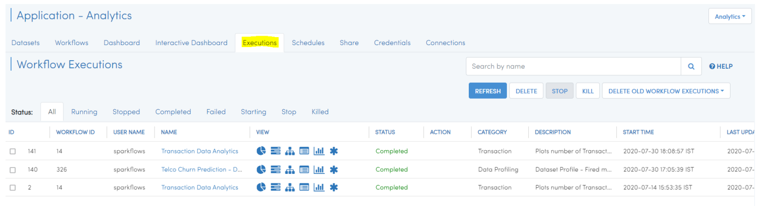
Once the workflow successfully completed, the output result can be exported.

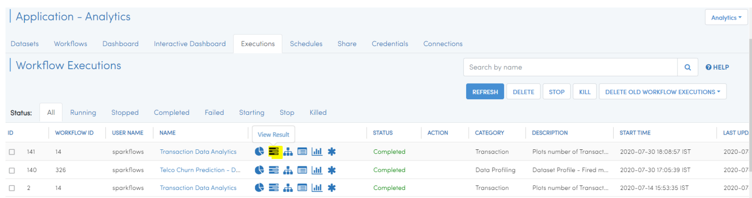
- Go to application page where you created workflow & successfully executed.

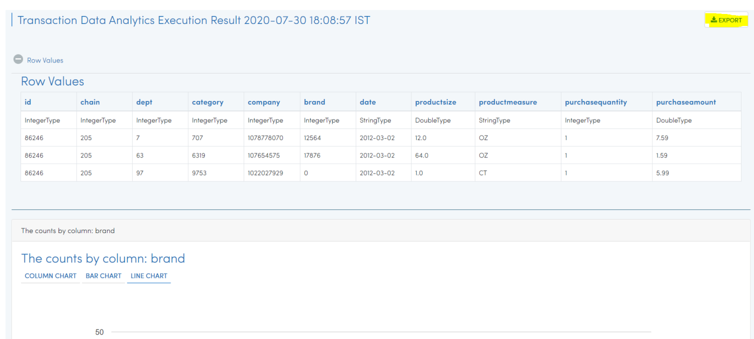
Clicking on `Executions` tab the latest workflow execution will show in list page.

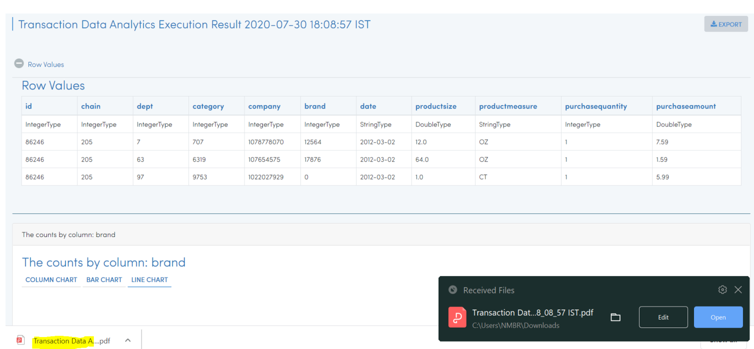
On action icon you can see `view result`, it will take to next page.

On opening above link, able to view result of specific workflow submitted & have Export option through which you can export result in local machine in pdf format & view that.


Application - Analytics
Analytics
Datasets Workflows Dashboard Interactive Dashboard Executions Schedules Share Credentials Connections
Workflow Executions
Search by name
HELP
REFRESH DELETE STOP KILL DELETE OLD WORKFLOW EXECUTIONS
Status: All Running Stopped Completed Failed Starting Stop Killed
ID WORKFLOW ID USER NAME NAME VIEW STATUS ACTION CATEGORY DESCRIPTION START TIME
141 14 sparkflows Transaction Data Analytics Completed Transaction Plots number of Transact... 2020-07-30 18:08:57 IST
140 326 sparkflows Telco Churn Prediction - D... Completed Data Profiling Dataset Profile - Fired m... 2020-07-30 17:05:39 IST
2 14 sparkflows Transaction Data Analytics Completed Transaction Plots number of Transact... 2020-07-14 15:53:35 IST


Application - Analytics
Analytics
Datasets Workflows Dashboard Interactive Dashboard Executions Schedules Share Credentials Connections
Workflow Executions
Search by name
HELP
REFRESH DELETE STOP KILL DELETE OLD WORKFLOW EXECUTIONS
Status: All Running Stopped Completed Failed Starting Stop Killed
ID WORKFLOW ID USER NAME NAME View Result STATUS ACTION CATEGORY DESCRIPTION START TIME
141 14 sparkflows Transaction Data Analytics Completed Transaction Plots number of Transact... 2020-07-30 18:08:57 IST
140 326 sparkflows Telco Churn Prediction - D... Completed Data Profiling Dataset Profile - Fired m... 2020-07-30 17:05:39 IST
2 14 sparkflows Transaction Data Analytics Completed Transaction Plots number of Transact... 2020-07-14 15:53:35 IST


Transaction Data Analytics Execution Result 2020-07-30 18:08:57 IST
EXPORT
Row Values
Row Values
id chain dept category company brand date productsize productmeasure purchasequantity purchaseamount
IntegerType IntegerType IntegerType IntegerType IntegerType IntegerType StringType DoubleType StringType IntegerType DoubleType
86246 205 7 707 1078778070 12564 2012-03-02 12.0 OZ 1 7.59
86246 205 63 6319 107654575 17876 2012-03-02 64.0 OZ 1 1.59
86246 205 97 9753 1022027929 0 2012-03-02 1.0 CT 1 5.99
The counts by column: brand
The counts by column: brand
COLUMN CHART BAR CHART LINE CHART
50


Transaction Data Analytics Execution Result 2020-07-30 18:08:57 IST
EXPORT
Row Values
Row Values
id chain dept category company brand date productsize productmeasure purchasequantity purchaseamount
IntegerType IntegerType IntegerType IntegerType IntegerType IntegerType StringType DoubleType StringType IntegerType DoubleType
86246 205 7 707 1078778070 12564 2012-03-02 12.0 OZ 1 7.59
86246 205 63 6319 107654575 17876 2012-03-02 64.0 OZ 1 1.59
86246 205 97 9753 1022027929 0 2012-03-02 1.0 CT 1 5.99
The counts by column: brand
The counts by column: brand
COLUMN CHART BAR CHART LINE CHART
Received Files
Transaction Dat...8_08_57 IST.pdf
C:\Users\NMBR\Downloads
Edit
Open
Transaction Data A....pdf

## 8.1.4 Scheduling

Fire allows you to schedule workflows by time to be executed.

### Scheduling Workflows

Fire allows you to schedule workflows to be run at regular intervals.

### Scheduling New Workflow

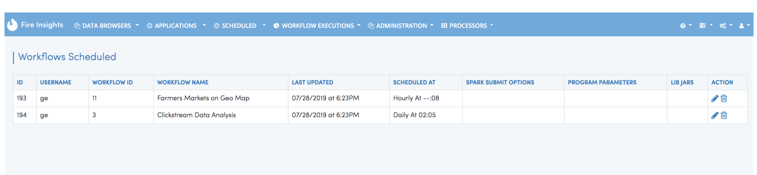
The workflows page displays the list of various workflows.

Under `Action` column, there is an icon to schedule any given workflow.

Clicking on the icon takes you to a page for creating new schedules for the workflow. Clicking on Schedule New Job for Workflow opens the dialog for creating a new schedule.


### Viewing Workflows Scheduled

Scheduled/By Time page displays the various workflows scheduled.



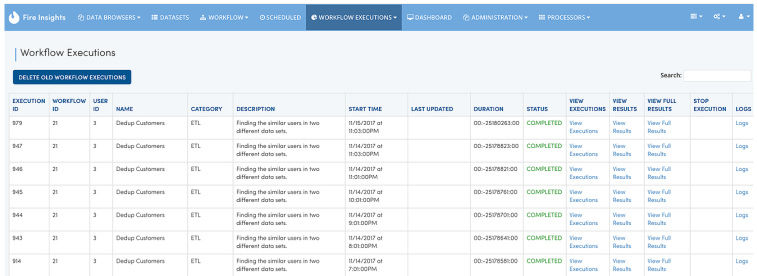

### Editing a Scheduled Workflow

You can edit a schedule by clicking on the edit icon, updating the new values and saving it.

### Viewing Results of Workflow Executions

When workflows are scheduled, they are executed by Fire at the specified schedule.

The results of the execution of the workflows can be viewed in the Workflow Executions Page. This allows us to view the results of past execution, logs of the run etc.




### Deleting a Scheduled Workflow

Go to the Scheduled/By Time page. It would display the list of scheduled workflows.

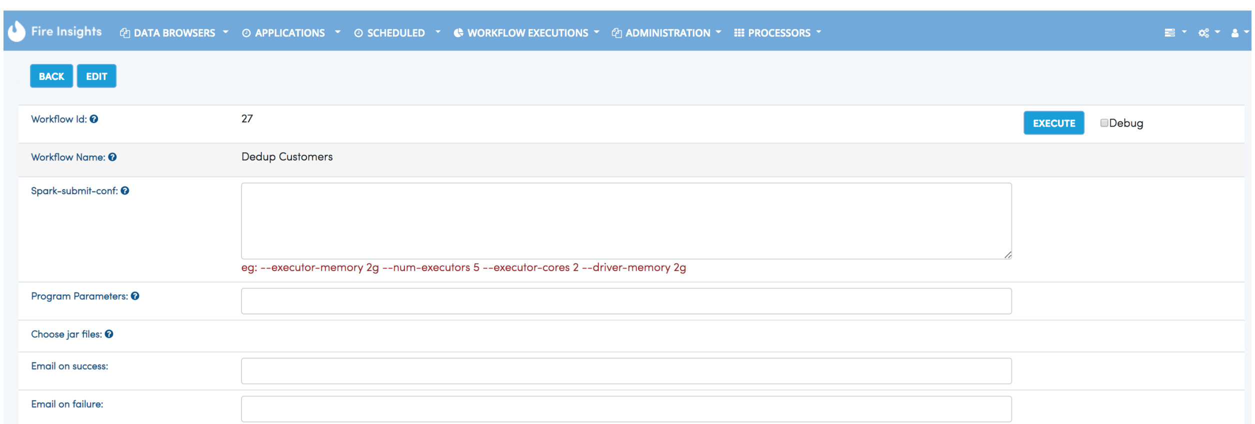
Click on the delete icon next to any schedule workflow to delete the schedule.

### Notifications & Alerts

Users in general like to be alerted when a job completes or fails, specially in Big Data where Jobs can run for hours together.

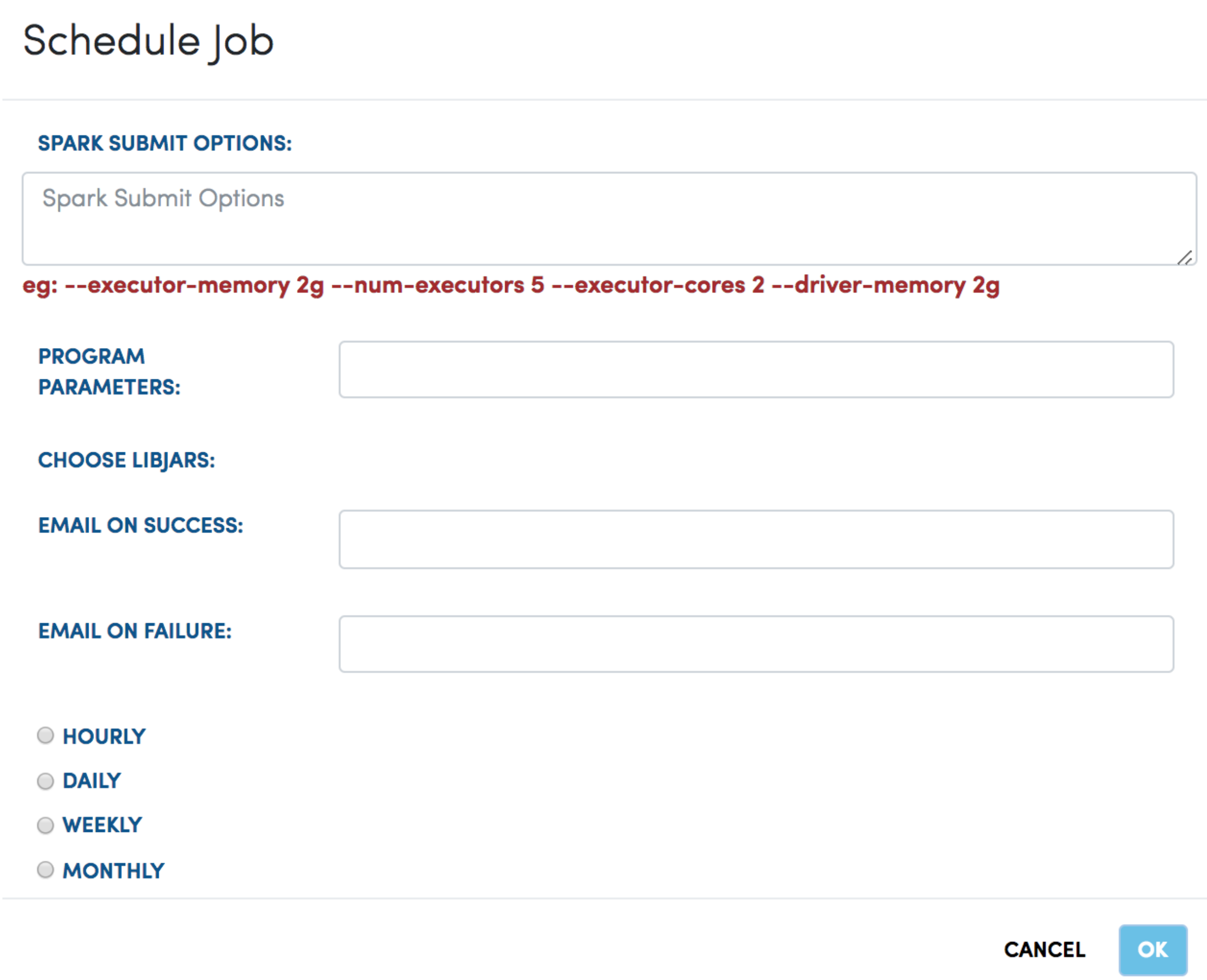
### Email Notifications/Alerts when Executing Workflows

When executing the workflows, you can specify email addresses for receiving emails when the workflow fails or succeeds.



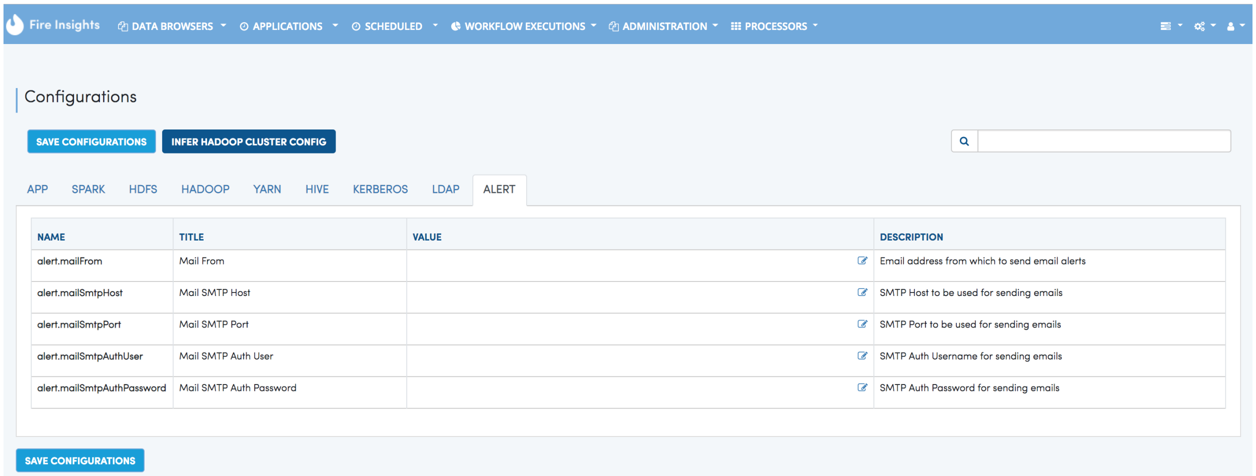

### Email Notifications/Alerts when Scheduling Workflows

When scheduling the workflows, you can specify email addresses for receiving emails when the workflow fails or succeeds.


Schedule Job
SPARK SUBMIT OPTIONS:
Spark Submit Options
eg: --executor-memory 2g --num-executors 5 --executor-cores 2 --driver-memory 2g
PROGRAM PARAMETERS:
CHOOSE LIBJARS:
EMAIL ON SUCCESS:
EMAIL ON FAILURE:
HOURLY
DAILY
WEEKLY
MONTHLY
CANCEL
OK

### SMTP Configurations

Administrator has to set up the SMTP configurations under Administration/Configuration




## Triggering Workflows by Event

Workflow Executions can be triggered by sending an event to a Kafka Topic.

Fire can be configured to poll for events from those topics.

### Use Case

The kind of use cases this can handle are:

- A job loads data into HIVE
- Now the job wants to trigger another workflow
- It pushes an event to a Kafka Topic to trigger the workflow

### Event Format

Events which are pushed to Kafka are string with the fields separated by `|` (pipe).

Below is the format of the event.

**Type|Value|Spark Submit Configs|Extra Jar Files|Program Parameters|Emails on Success|Emails on Failure**

- `Type` : Type determines the kind of data in the Value column
  - 0 : workflow id
  - 1 : workflow name
  - 2 : workflow uuid
- `Value` : This defines the value. Values are based on the Type used:
  - ID of the workflow

  - Name of the workflow
  - UUID of the workflow
- `Spark Submit Configs` : Extra Spark Submit configurations to be applied when running the Spark Job.
- `Extra Jar files` : Extra jar files to use in spark-submit.
- `Program parameters` : Extra program parameters if any.
  - Program Parameters are passed to the workflow. Example : `--var key1=value1`.
- `Email on Success` : email addresses to send Success email on Job Completion.
- `Email on Failure` : email addresses to send Failure email on Job Failure.

### Example Events

- 0|5| | | |success@sparkflows.io|failure@sparkflows.io

In the above example:

- 0 : Trigger by workflow id
- 5 : Workflow id to trigger
- success@sparkflows.io : Email address to send regarding success of the workflow
- failure@sparkflows.io : Email address to send regarding failure of the workflow

### Configuring Fire to listen for Events from the Kafka Topic

Fire has to be configured to listen for Events from the Kafka Topic. Each user can configure their own. The Jobs would be fired as a user who configured it.

## 8.1.5 Export / Import of Applications

Fire enables you to export your Applications and download them to your computer.

It then also enables you to import your Applications back into any instance of Fire.

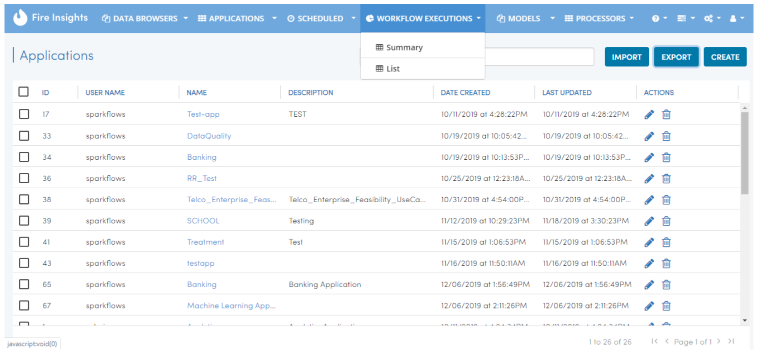
This is useful when you need to move/copy your Application from one environment to another.

### Exporting Applications

Fire allows you to export Applications and download them to your computer.

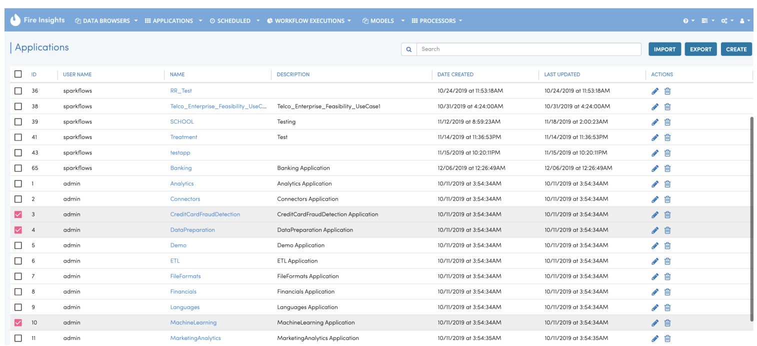
Below are the steps for exporting Applications in Fire.

### Go to the Applications Page

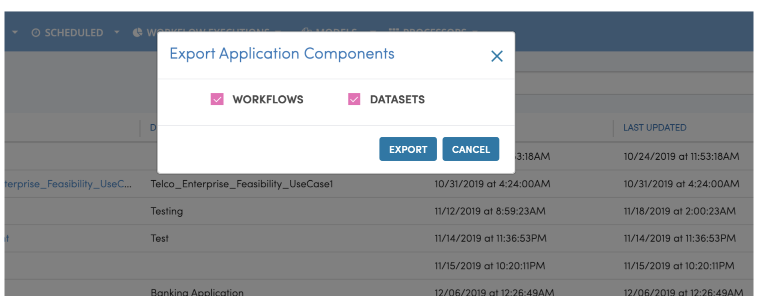



## Select the Applications you want to export

- Select the Applications you would like to export.
- Then click on the Export button.



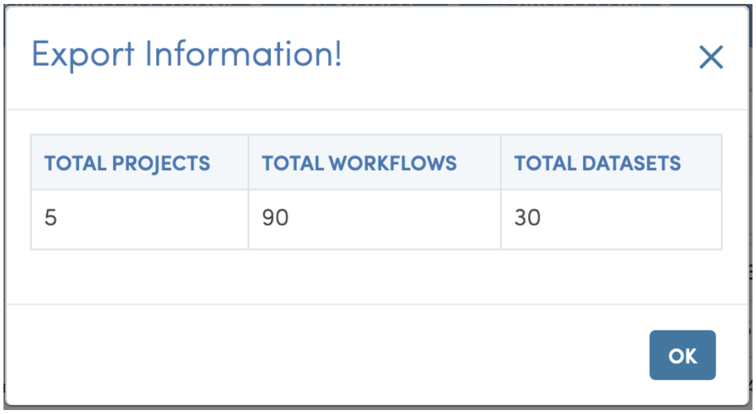

- In the dialog box which comes up, select whether you want to export workflows or datasets or both.




- Fire will now export the selected applications and download them to your computer.

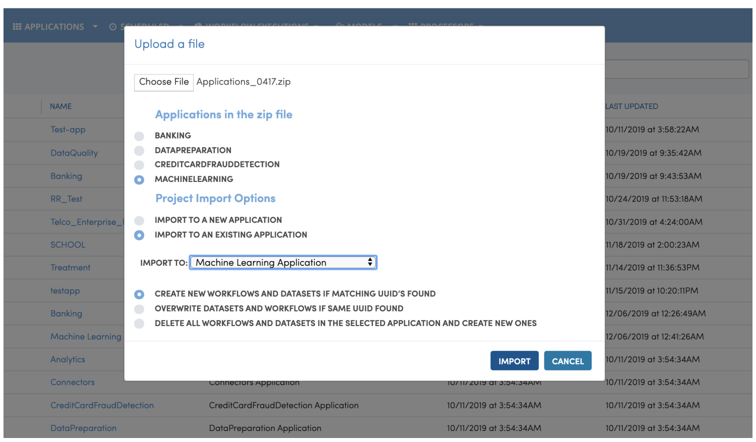
## Importing Applications

Fire allows you to import Applications. Below are the steps for importing Applications in Fire.




### Go to the Applications Page

- Click on the Import button.
- Choose the zip file from your computer to Import from. You would have previously downloaded this zip file from Fire during the export process.
- Select the name of the Application which you would like to import from the zip file. Fire would display all the available Applications in your zip file.




### Select the Options for importing the Application

There are two options when importing Applications:

- Import to a New Application
  - In this case, the selected Application would be imported as a new Application in Fire Insights.
- Import to an Existing Application

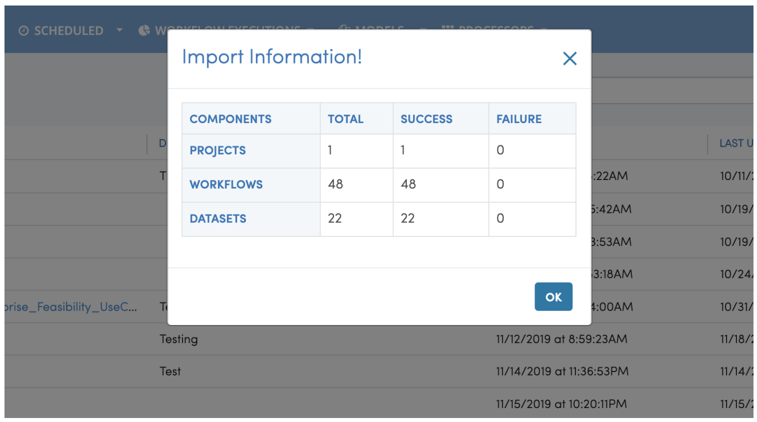
When importing to an existing Application, there are 3 possible methods to choose from:

- Create new workflows and datasets when matching UUID's found.
- Overwrite datasets and workflows if matching UUID found.

- Delete all workflows and datasets in the selected Application and create the imported workflows and datasets as new ones.

### On Success

On successful import of the Application into Fire Insights, the success dialog is displayed along with the details of the import.




## 8.1.6 Data profiling

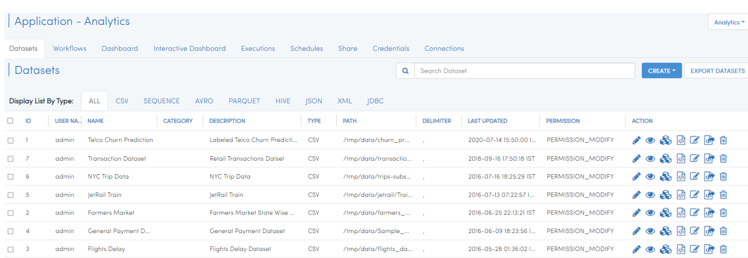
Fire Insights allows you to clean the datasets using dataset profile.

Below are the steps for Data Profiling in Fire.

### Go to the Applications Page

Go to application page where you need to create dataset or already have existing.

select `dataset` tab.



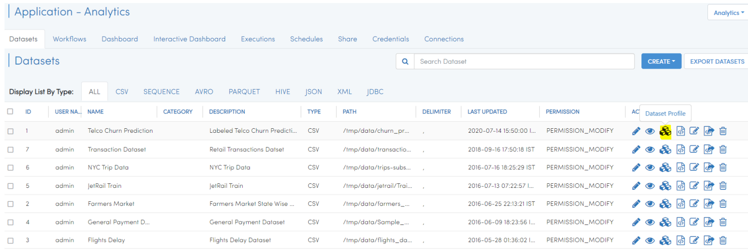

Select a dataset & under `action` icon choose Dataset profile.

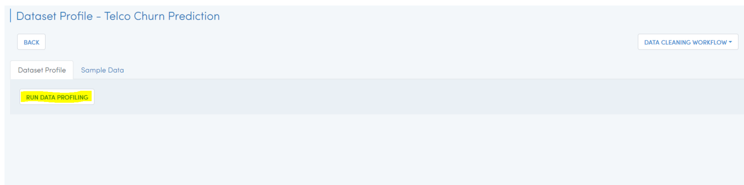
Once you Click on Dataset profile, it will take us to next page.

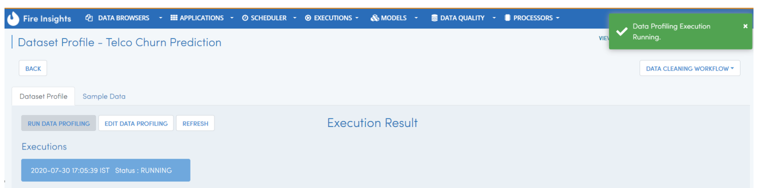
Click on `RUN DATA PROFILING` option

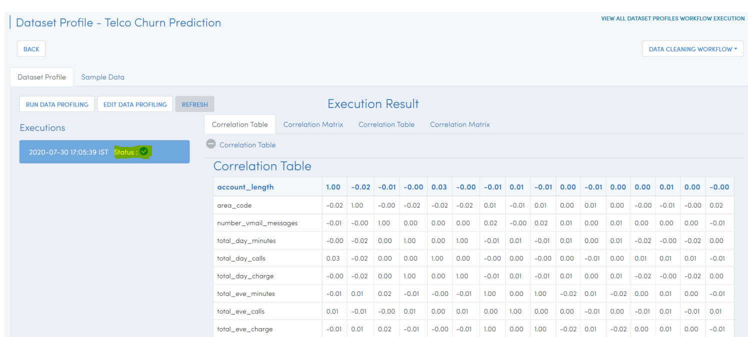
Once you click on above option, will get notifications about process is getting started.

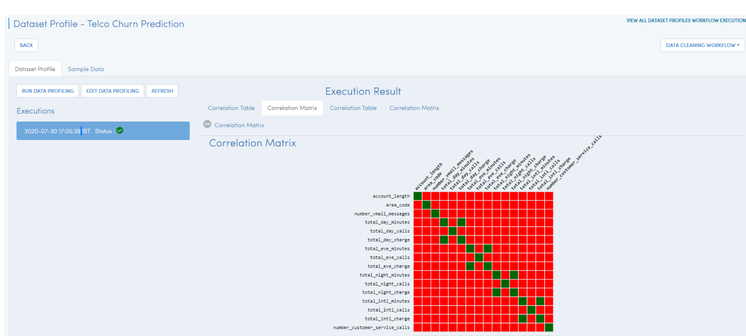
Once the `execution` process completed, after refresh the status will updated to green, if its completed and check its execution result in RHS


Application - Analytics
Datasets


Dataset Profile - Telco Churn Prediction


Dataset Profile - Telco Churn Prediction
Execution Result
Executions


Dataset Profile - Telco Churn Prediction
Execution Result
Executions
Correlation Table

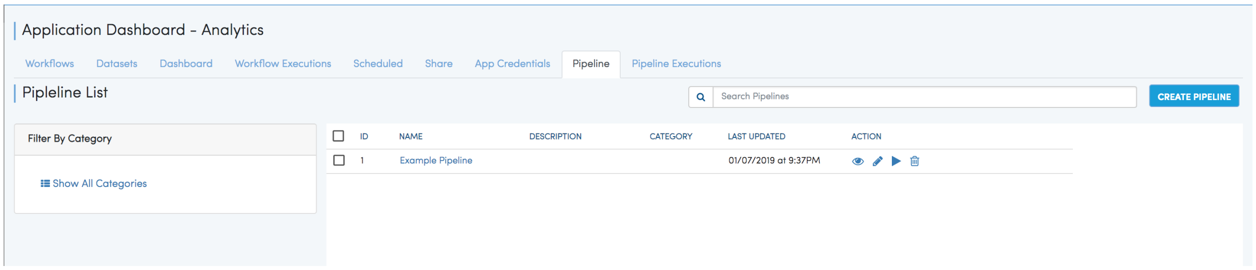

Dataset Profile - Telco Churn Prediction
Execution Result
Correlation Matrix

### 8.1.7 Pipeline

Fire supports Pipelines. Pipelines allow running workflows in a defined order.

#### Pipeline List

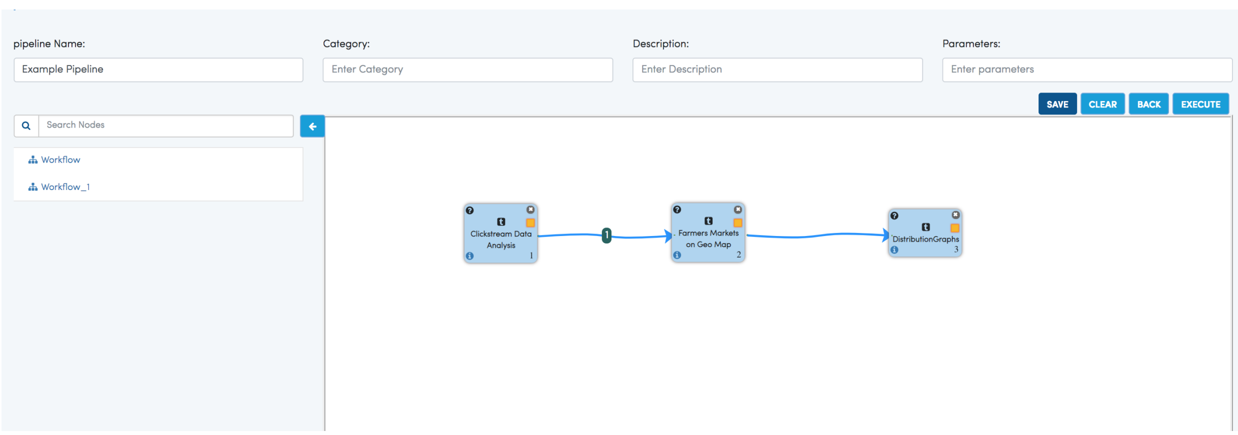
The Pipeline tab displays the list of Pipelines for the current Application.




#### Creating a Pipeline

Each Application now allows creating Pipelines.

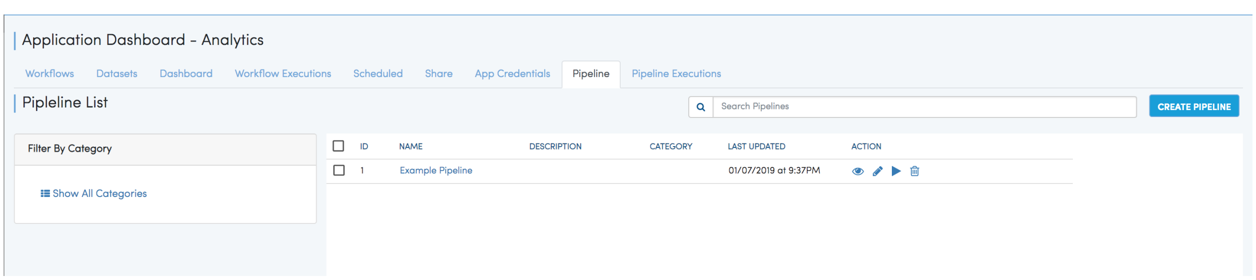
Below is an example Pipeline with 3 Workflows.



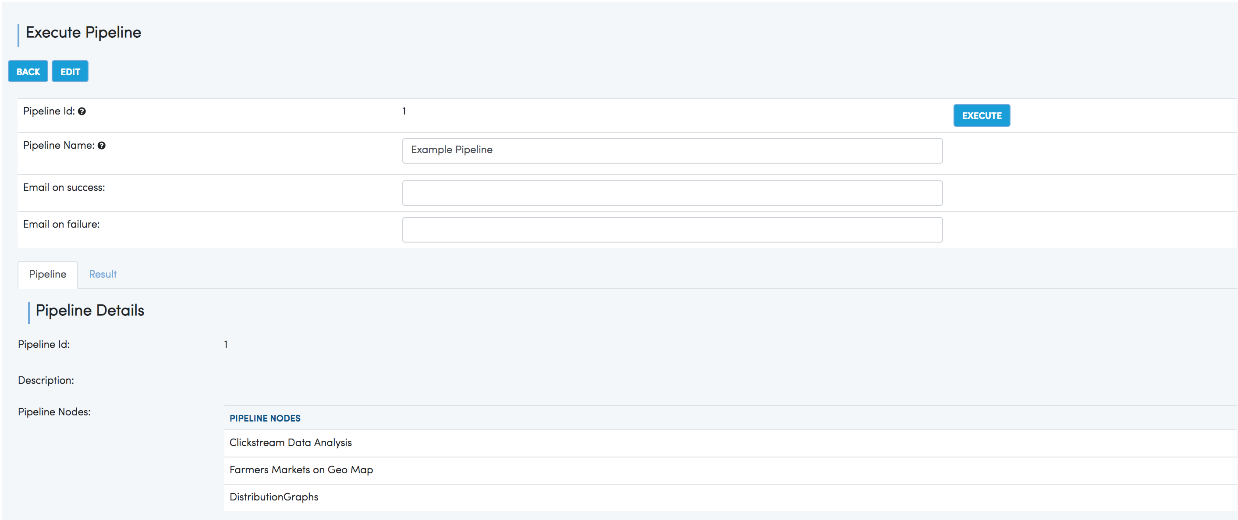

#### Executing a Pipeline

Pipelines can be executed like workflows. When a Pipeline is executed, its execution is submitted to Airflow.

The Pipeline tab displays the list of Pipelines for the current Application.

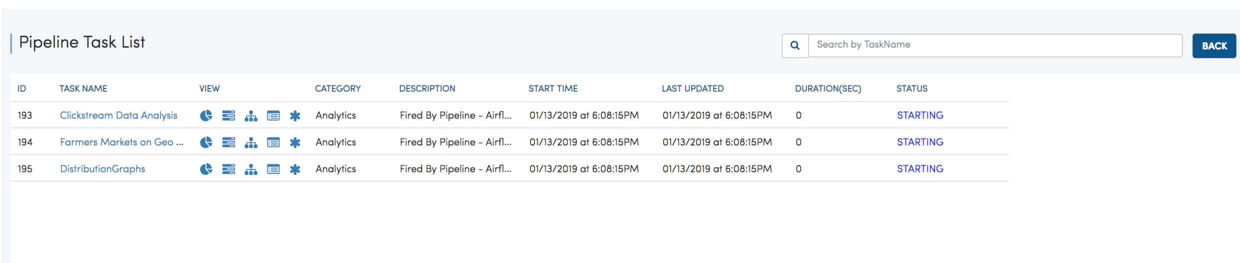


Clicking on the `Execute` Action icon opens the Pipeline Execute Page.




### Pipeline Execution

Once a Pipeline is fired, its details are visible in Pipeline Executions.




## 8.1.8 OCR with Tesseract

In order to run Tesseract, the below Installation steps have to be performed.

### Download & Install the Tesseract Language Data files

- Download and Install the tesseract language data files for Version 3.X on each of the worker nodes of the cluster:

```
https://github.com/tesseract-ocr/tessdata/releases
wget https://github.com/tesseract-ocr/tessdata/archive/3.04.00.tar.gz
```

- Install them in the same directory on each of the worker nodes:

```
git clone https://github.com/tesseract-ocr/tessdata.git
```

### Include TESSDATA_PREFIX in spark configs when submitting the job

- Include the following in spark submit configs when running workflows containing the OCR node:

```
--conf spark.executorEnv.TESSDATA_PREFIX=/home/ec2-user/tessdata
```

- Where the tesseract language data files are in `/home/ec2-user/tessdata` directory on each of the worker nodes

### Error if TESSDATA_PREFIX is not set correctly

If TESSDATA_PREFIX is not set, the spark program would run into the error below:

```
Error opening data file /Users/saudet/projects/bytedeco/javacpp-presets/tesseract/
↪cppbuild/macosx-x86_64/share/tessdata/eng.traineddata
Please make sure the TESSDATA_PREFIX environment variable is set to the parent␣
↪directory of your "tessdata" directory.
Failed loading language 'eng'
Tesseract couldn't load any languages!
```

The above error would be in the Job logs. If yarn is being used it would be in the yarn logs:

```
yarn logs -applicationId job_application_id
```

When the job is being executed, Fire displays the job_application_id in the browser.

CHAPTER 9

# Web App User Guide

## 9.1 Analytical Apps User Guide

### 9.1.1 Creating Analytics App

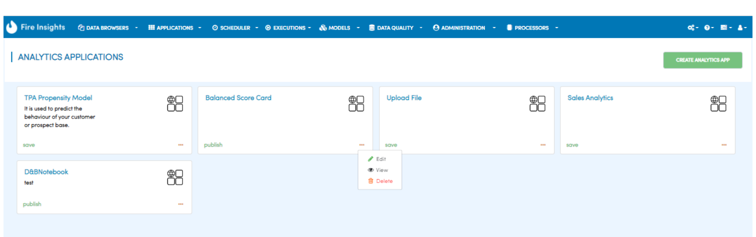
Fire Insights enables you to create Analytics Apps.

Below is the process for creating a new Analytics App.

- *Go to APPLICATIONS / ANALYTICS APPS*
- *Click on "Create Analytics App"*
- *Add mandatory fields i.e. "Name", "select notebook"*
- *Click on add stage button to add different stages*
- *Click "Save" Or "Publish"*

#### Go to Analytics Apps

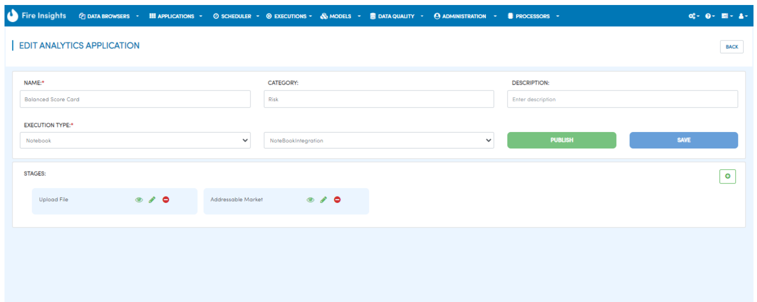
When you go to ANALYTICS APPS under APPLICATIONS all existing analytics app are displayed. Where you can EDIT, VIEW and DELETE existing analytics app.



### Click on Create Analytics App

Fill in the required fields as below.

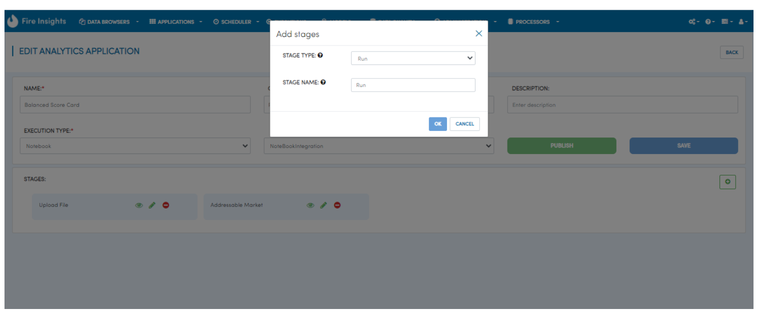
- *Name* : Name of the new analytics app
- *category* : Category of the new analytics app
- *Description* : Description of the new analytics app
- *Execution Type:* : Select execution type i.e notebook and select notebook from the available notebook list




"Save" or "Publish" the analytics app before adding stages.

## 9.1.2 Adding Stages

Click on "Add stages" button to add a new stage. Select stage type and enter the stage name.



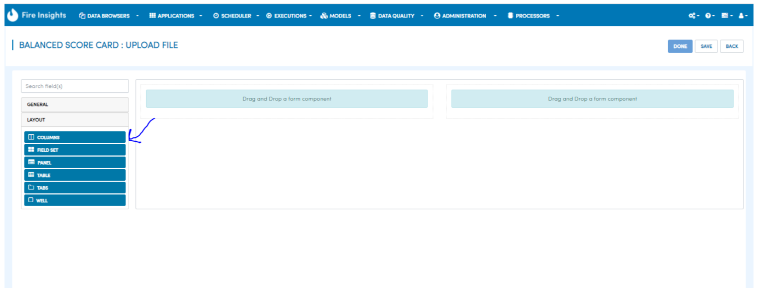

- You can rearrange the stages by dragging and dropping.
- You can EDIT, VIEW and REMOVE stages.

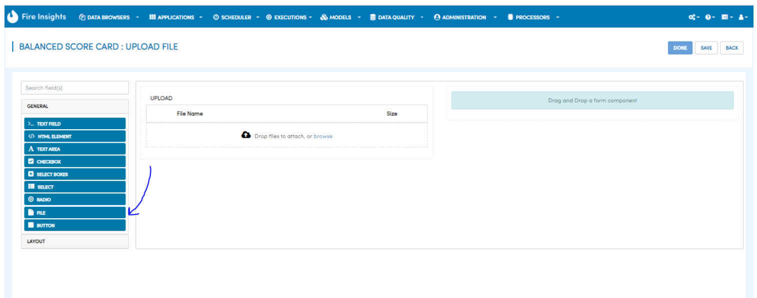
### Examples for adding various Stages

### 1 : Upload Stage

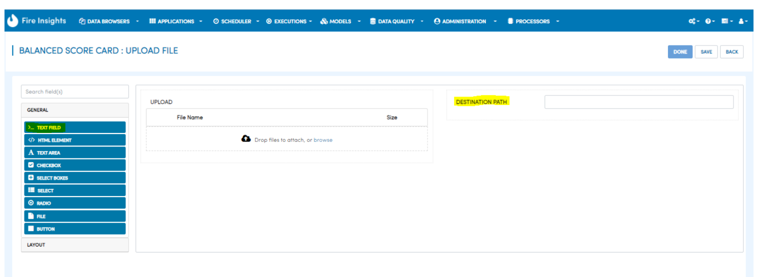
- In upload stage we will first add column component and divide in two columns

- In first column add file component to choose files to upload to databricks. In this component in File tab in "STORAGE" select "Base64"





- In other column we will add one textfield to add "DESTINATION PATH" where the browse file should get uploaded. Set its property name to `destinationPath`.



- Add upload button and set action to `event`. Set the button event name to `upload`.

- Also add next button to go to next stage and perform actions depending upon event. Set the event name as `next` for the next button.

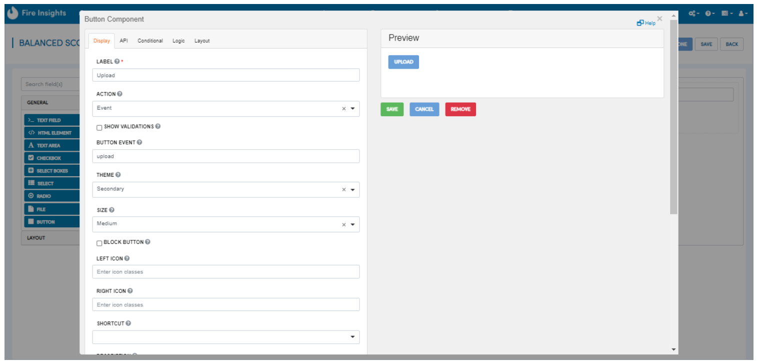
Click on "DONE" or "SAVE" to save added components for that stage

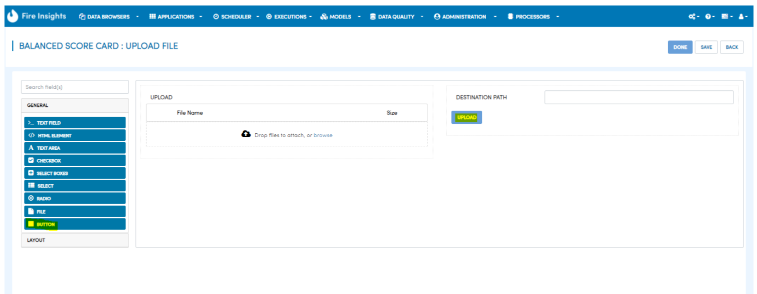
## 2 : Parameters Stage

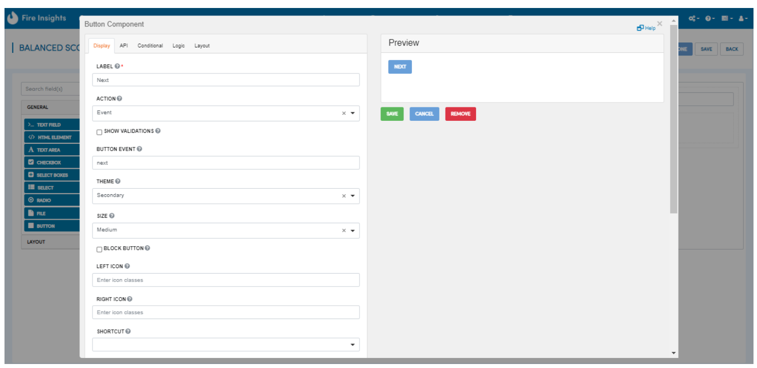
- In parameters stage we can add `select, text-field, select boxes, buttons` etc components

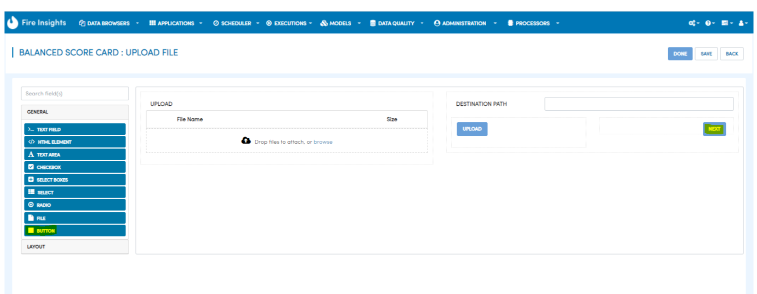
For example:

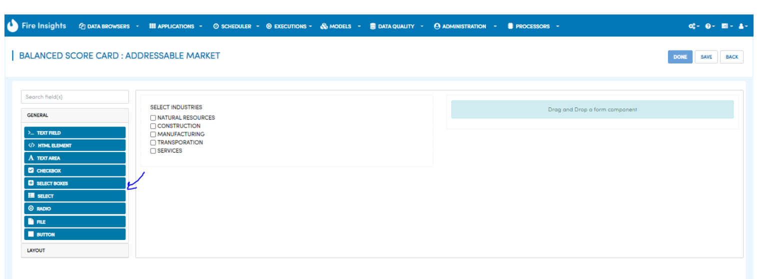
- First we will add column component and divide it in two columns
- Then, lets add select boxes example in first column by adding select boxes component. In this component in Data tab add all possible values you want to add.


Fire Insights
Button Component
Preview
BALANCED SCO

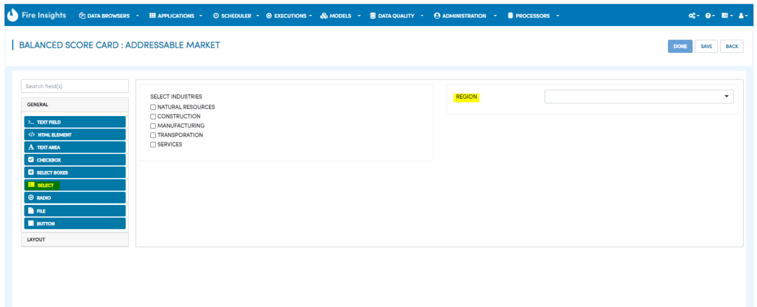

Fire Insights
BALANCED SCORE CARD : UPLOAD FILE
UPLOAD
DESTINATION PATH


Fire Insights
Button Component
Preview
BALANCED SCO

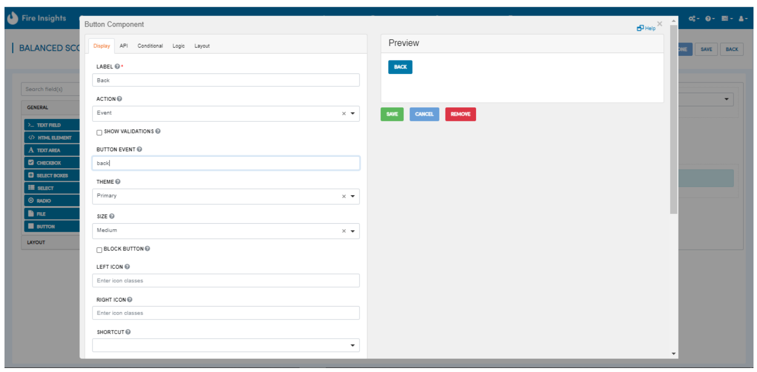

Fire Insights
BALANCED SCORE CARD : UPLOAD FILE
UPLOAD
DESTINATION PATH


Fire Insights
BALANCED SCORE CARD : ADDRESSABLE MARKET
SELECT INDUSTRIES
NATURAL RESOURCES
CONSTRUCTION
MANUFACTURING
TRANSPORATION
SERVICES

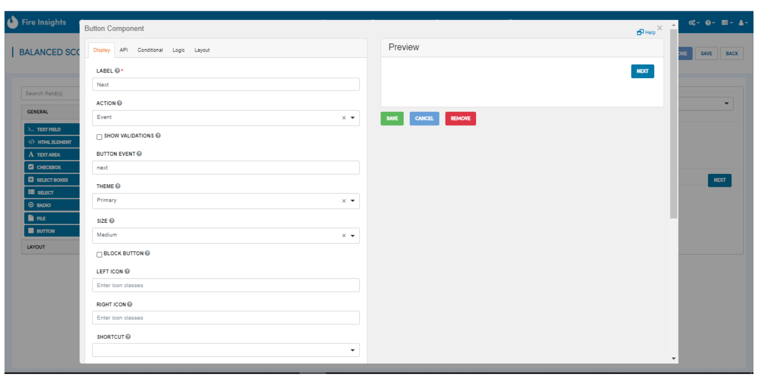
- Then, lets add select example in the second column by adding select component. In this component in Data tab add all possible values you want to add.



- Now, lets add column component in the bottom and divide into two columns for adding back and next button.
- Add back button in first column to go to back stage and perform actions depending upon event, where we will add event name as `back`.



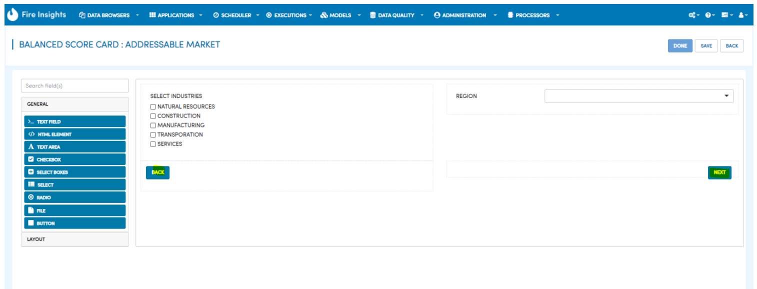

- Add next button in second column to go to next stage and perform actions depending upon event. Set its event name as `next`. We can also add CUSTOM CSS CLASS like `float-right, float-left` etc

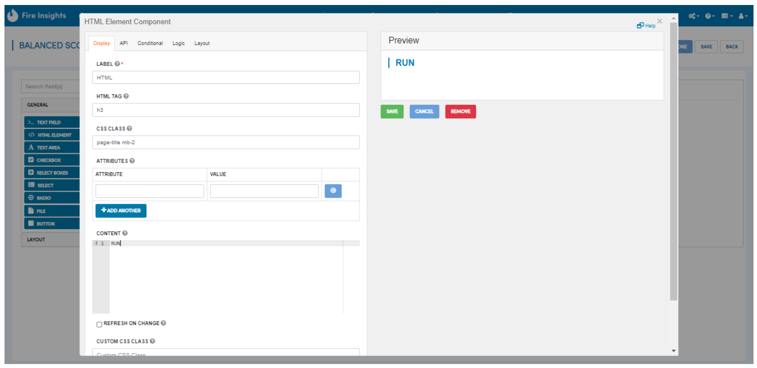



Click on "DONE" or "SAVE" to save the added components for that stage.

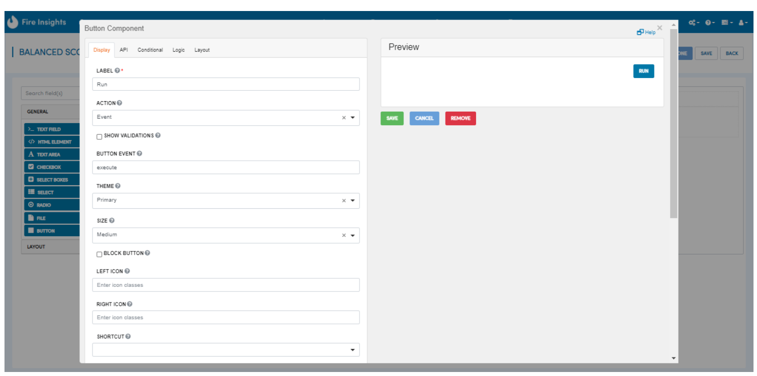
## 3 : Run Stage

- In run stage we will execute the notebook with all parameters added in the App.
- Let's first add title in page if needed with "html element" component like below.




- Now, lets add column component in the bottom and divide it into two columns for adding the `back` and `run` buttons.
- Add back button in first column to go to back stage and perform actions depending upon event. Set its event name as `back`.
- Add next button in second column to go to next stage and perform actions depending upon event. Set its add event name as `execute`. We can also can add CUSTOM CSS CLASS like float-right, float-left etc



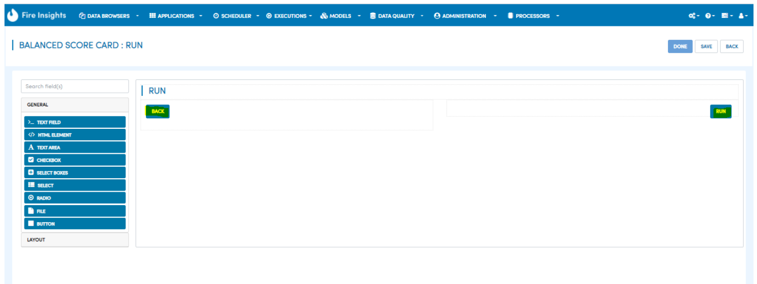

Click on "DONE" or "SAVE" to save added components for that stage

### 9.1.3 Integrating with Databricks Notebook

The Web App in Fire Insights can trigger a Notebook in Databricks.

Fire Insights passes 2 parameters to the Notebook:

- postback-url




- job-id

### Add wheel file to your Databricks Notebook

Add the wheel file to your Databricks Notebook. This is to enable using the Fire Insights API's for sending data to it.

### Outputing details to Fire Insights

The Databricks Notebook can output text, tables and charts to be dispalyed in Fire Insights.

Below are the examples for it.

### Create a RestWorkflowContext Object

First create a `RestWorkflowContext` for communicating with Fire Insights Server

```
jobId = dbutils.widgets.get("job-id")
webserverURL = dbutils.widgets.get("postback-url")

print(webserverURL)
print(jobId)

from fire_notebook.output.workflowcontext import RestWorkflowContext

restworkflowcontext = RestWorkflowContext(webserverURL, jobId)
```

### Outputing Text

Below is how to output text to Fire Insights

```
restworkflowcontext.outStr(9, "Test String")
```

### Outputing PySpark Dataframe as Table

The below code outputs the contents of PySpark Dataframe to Fire Insights as a table

```
from pyspark.sql.types import *

schema = StructType([StructField("c1", DoubleType())\
```

(continues on next page)

(continued from previous page)

```
                    ,StructField("c2", IntegerType())])
test_list = [[0.0, 2], [1.0, 4], [2.0, 8], [3.0, 16], [4.0, 32], [5.0, 64], [6.0,␣
↪128]]
df = spark.createDataFrame(test_list,schema=schema)
restworkflowcontext.outDataFrame(9, "PySpark Dataframe", df)
```

### Outputing Pandas Dataframe as Table

The below code outputs the contents of Pandas Dataframe to Fire Insights as a table

```
# list of strings
lst = ['Geeks', 'For', 'Geeks', 'is',
       'portal', 'for', 'Geeks']

# Calling DataFrame constructor on list
df = pd.DataFrame(lst, columns=['name'])
print(df)

restworkflowcontext.outPandasDataframe(9, "Names", df)
```

### Outputing CHART

Output the chart in fire by selecting x & y column and Different type of chartType: COLUMNCHART, BARCHART & LINECHART

from pyspark.sql.types import *

**schema = StructType([StructField("c1", DoubleType())** ,StructField("c2", IntegerType())])

test_list = [[0.0, 2], [1.0, 4], [2.0, 8], [3.0, 16], [4.0, 32], [5.0, 64], [6.0, 128]]

df = spark.createDataFrame(test_list,schema=schema)

**restworkflowcontext.outDataframeChart(title= "Example Chart", x_column = "c1", y_columns = ["c2"],**
chart_type ="LINECHART", df = df, numRowsToDisplay = 10)

### Outputing HTML

Below is how to output html to Fire Insights

```
htmlstr1 = "<h3>You can view HTML code in notebooks.</h3>"

restworkflowcontext.outHTML(9, title="Example HTML", text = htmlstr1)
```

### Outputing Plotly

Below is how to output plotly to Fire Insights

```
import plotly.graph_objs as go
import plotly
```

(continues on next page)

(continued from previous page)

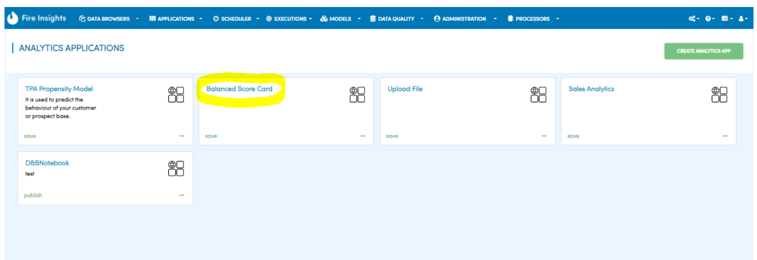
```
test = plotly.offline.plot([go.Scatter(x=[1, 2, 3], y=[3, 2, 6])],
                  output_type='div',
                 include_plotlyjs=False)
example_plotly = f'{test}'

restworkflowcontext.outPlotly(9, title="Example Plotly", text = example_plotly)
```

### 9.1.4 Running Analytics App

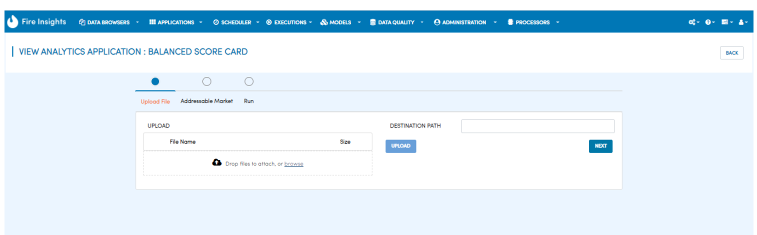
Once the Analytics App has been created, they can be executed.

Below are the steps for executing an Analytics App.

#### Click on Analytics App Name



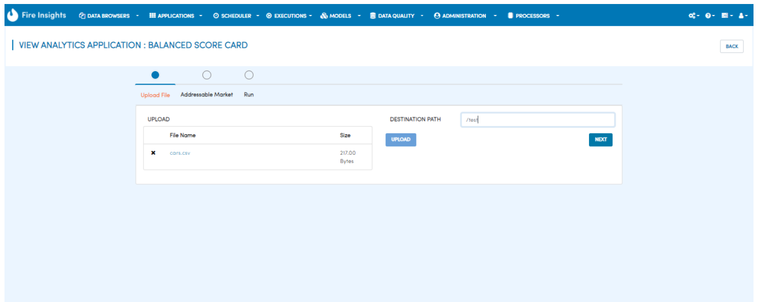

#### Go through the various Stages

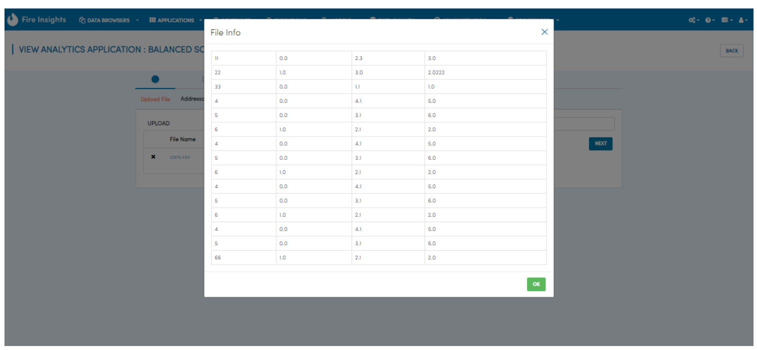


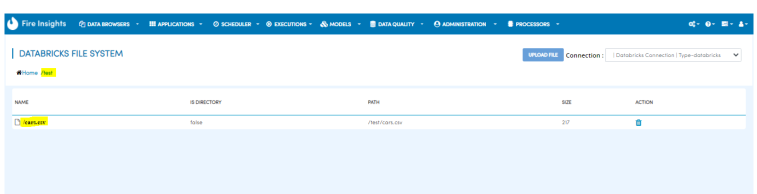

#### Examples of the various Stage Pages

#### 1 : Upload

- Browse files you want to upload to databricks.
- Add destination path of dbfs where you want to upload choose file.
- If added path is not there in dbfs then it will first create the folder in dbfs and then upload the file.
- Then, click on upload button to upload to DBFS and see the csv file data in tabular format.



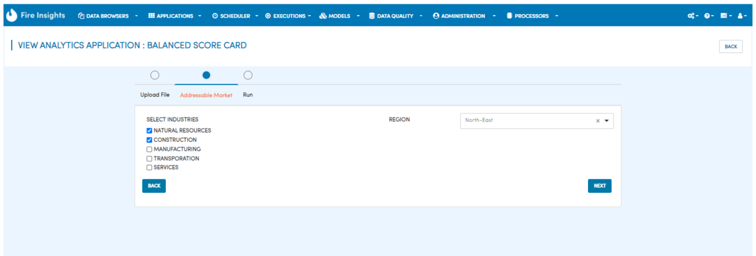




- You can browse dbfs and check if the file uploaded successfully.
- Click on "NEXT" button to go to next stage.

## 2 : Parameters

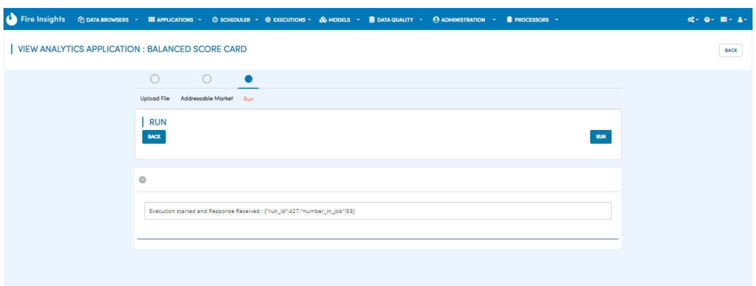
- Select the parameters of your interest
- If you click on "BACK" or "NEXT" button the selected value will remain as it is and you can change it if needed




- Click on "NEXT" button to move to next page

## 3 : Run

- In this stage you will execute the Analytics App with the added parameters in the earlier stages.
- You can click on back button and change the value and run Analytics App again.
- Click on "RUN" button to execute the app and view the results.



CHAPTER 10

# Data Science

## 10.1 Machine Learning User Guide

### 10.1.1 Feature Generation

Feature generation is the process of creating new features from one or multiple existing features, potentially for using in statistical analysis. This process adds new information to be accessible during the model construction and therefore hopefully result in more accurate model.

Table 1: Fire Insights provides a number of processors for Feature Generation. These include:

| Title | Description |
|---|---|
| DateToAge | Convert Date to Age |
| CaseWhen | Based on the value, convert it to another value |
| Scala | Write Scala code in Spark for generating new Features |
| SQL | Write SQL code for generating new features |
| StopWOrdRemover | Removes Stop Words |
| Tokenizer | Tokenizes a string into Tokens |
| OneHotEncoder | Applies one hot encoding |
| TF/IDF | Finds the TF and IDF |
| IndexString | Converts a column containg String to numeric values |

### 10.1.2 Feature Selection

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, in the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for several reasons:

- simplification of models to make them easier to interpret by researchers/users
- shorter training times

- to avoid the curse of dimensionality
- enhanced generalization by reducing overfitting (formally, reduction of variance)
- https://en.wikipedia.org/wiki/Feature_selection

Apache Spark has the following Feature Selectors. Fire Insights provides them as Processors to be easily used in the workflows:

### Feature Selection Processors in Fire Insights

Table 2: Apache Spark based Feature Selection Processors in Fire Insights

| Title | Description |
|---|---|
| VectorSlicer | VectorSlicer is a transformer that takes a feature vector and outputs a new feature vector with a sub-array of the original features. It is useful for extracting features from a vector column. VectorSlicer accepts a vector column with specified indices, then outputs a new vector column whose values are selected via those indices. |
| RFormula | RFormula selects columns specified by an R model formula. RFormula produces a vector column of features and a double or string column of label. Like when formulas are used in R for linear regression, string input columns will be one-hot encoded, and numeric columns will be cast to doubles. If the label column is of type string, it will be first transformed to double with StringIndexer. If the label column does not exist in the DataFrame, the output label column will be created from the specified response variable in the formula. |
| ChiSqSelector | ChiSqSelector stands for Chi-Squared feature selection. It operates on labeled data with categorical features. ChiSqSelector uses the Chi-Squared test of independence to decide which features to choose. It supports five selection methods: numTopFeatures, percentile, fpr, fdr, fwe |

More details regarding the Feature Selectors in Spark can be found at:

https://spark.apache.org/docs/2.2.0/ml-features.html#feature-selectors

- VectorSlicer
- RFormula
- ChiSqSelector

### VectorSlicer

VectorSlicer is a transformer that takes a feature vector and outputs a new feature vector with a sub-array of the original features. It is useful for extracting features from a vector column. VectorSlicer accepts a vector column with specified indices, then outputs a new vector column whose values are selected via those indices. There are two types of indices,

Integer indices that represent the indices into the vector, setIndices().

String indices that represent the names of features into the vector, setNames(). This requires the vector column to have an AttributeGroup since the implementation matches on the name field of an Attribute.

Specification by integer and string are both acceptable. Moreover, you can use integer index and string name simultaneously. At least one feature must be selected. Duplicate features are not allowed, so there can be no overlap between selected indices and names. Note that if names of features are selected, an exception will be thrown if empty input attributes are encountered.

#### RFormula

RFormula selects columns specified by an R model formula. Currently Spark supports a limited subset of the R operators, including '~', '.', ':', '+', and '-'. The basic operators are:

- ~ separate target and terms
- – concat terms, "+ 0" means removing intercept
- – remove a term, "- 1" means removing intercept
- : interaction (multiplication for numeric values, or binarized categorical values)
- . all columns except target

Suppose a and b are double columns, we use the following simple examples to illustrate the effect of RFormula:

- y ~ a + b means model y ~ w0 + w1 * a + w2 * b where w0 is the intercept and w1, w2 are coefficients.
- y ~ a + b + a:b - 1 means model y ~ w1 * a + w2 * b + w3 * a * b where w1, w2, w3 are coefficients.

RFormula produces a vector column of features and a double or string column of label. Like when formulas are used in R for linear regression, string input columns will be one-hot encoded, and numeric columns will be cast to doubles. If the label column is of type string, it will be first transformed to double with StringIndexer. If the label column does not exist in the DataFrame, the output label column will be created from the specified response variable in the formula.

#### ChiSqSelector

ChiSqSelector stands for Chi-Squared feature selection. It operates on labeled data with categorical features. ChiSqSelector uses the Chi-Squared test of independence to decide which features to choose. It supports five selection methods: numTopFeatures, percentile, fpr, fdr, fwe. * numTopFeatures chooses a fixed number of top features according to a chi-squared test. This is akin to yielding the features with the most predictive power. * percentile is similar to numTopFeatures but chooses a fraction of all features instead of a fixed number. * fpr chooses all features whose p-values are below a threshold, thus controlling the false positive rate of selection. * fdr uses the Benjamini-Hochberg procedure to choose all features whose false discovery rate is below a threshold. * fwe chooses all features whose p-values are below a threshold. The threshold is scaled by 1/numFeatures, thus controlling the family-wise error rate of selection. By default, the selection method is numTopFeatures, with the default number of top features set to 50. The user can choose a selection method using setSelectorType.

### 10.1.3 Clustering

Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

- https://en.wikipedia.org/wiki/Cluster_analysis

### Clustering Processors in Fire Insights

Table 3: Apache Spark based Clustering Processors in Fire Insights

| Title | Description |
|---|---|
| Gaussian Mixture | A Gaussian Mixture Model represents a composite distribution whereby points are drawn from one of k Gaussian sub-distributions, each with its own probability. The spark.ml implementation uses the expectation-maximization algorithm to induce the maximum-likelihood model given a set of samples. |
| KMeans | k-means is one of the most commonly used clustering algorithms that clusters the data points into a predefined number of clusters. The MLlib implementation includes a parallelized variant of the k-means++ method called kmeans\|\|. |
| LDA | LDA is implemented as an Estimator that supports both EMLDAOptimizer and OnlineLDAOptimizer, and generates a LDAModel as the base model. |

Table 4: H2O based Clustering Processors in Fire Insights

| Title | Description |
|---|---|
| KMeans | K-Means falls in the general category of clustering algorithms. Clustering is a form of unsupervised learning that tries to find structures in the data without using any labels or target values. Clustering partitions a set of observations into separate groupings such that an observation in a given group is more similar to another observation in the same group than to another observation in a different group. |

### Clustering Algorithms in Apache Spark

Apache Spark provides a number of Algorithms for Clustering.

https://spark.apache.org/docs/latest/ml-clustering.html

- K-means
- Latent Dirichlet allocation (LDA)
- Bisecting k-means
- Gaussian Mixture Model (GMM)
- Power iteration clustering (PIC)
- Streaming k-means

### K-means

https://spark.apache.org/docs/latest/ml-clustering.html#k-means k-means is one of the most commonly used clustering algorithms that clusters the data points into a predefined number of clusters. The MLlib implementation includes a parallelized variant of the k-means++ method called kmeans||. The implementation in spark.mllib has the following parameters:

k is the number of desired clusters. Note that it is possible for fewer than k clusters to be returned, for example, if there are fewer than k distinct points to cluster. - maxIterations is the maximum number of iterations to run. - initializationMode specifies either random initialization or initialization via k-means||. - runs This param has no effect since Spark 2.0.0. - initializationSteps determines the number of steps in the k-means|| algorithm. - epsilon determines the distance threshold within which we consider k-means to have converged. - initialModel is an optional set of cluster centers used for initialization. If this parameter is supplied, only one run is performed.

### Latent Dirichlet allocation (LDA)

https://spark.apache.org/docs/latest/ml-clustering.html#latent-dirichlet-allocation-lda

LDA is implemented as an Estimator that supports both EMLDAOptimizer and OnlineLDAOptimizer, and generates a LDAModel as the base model. Expert users may cast a LDAModel generated by EMLDAOptimizer to a DistributedLDAModel if needed.

Latent Dirichlet allocation (LDA) is a topic model which infers topics from a collection of text documents. LDA can be thought of as a clustering algorithm as follows:

- Topics correspond to cluster centers, and documents correspond to examples (rows) in a dataset.
- Topics and documents both exist in a feature space, where feature vectors are vectors of word counts (bag of words).
- Rather than estimating a clustering using a traditional distance, LDA uses a function based on a statistical model of how text documents are generated.

LDA supports different inference algorithms via setOptimizer function. EMLDAOptimizer learns clustering using expectation-maximization on the likelihood function and yields comprehensive results, while OnlineLDAOptimizer uses iterative mini-batch sampling for online variational inference and is generally memory friendly.

LDA takes in a collection of documents as vectors of word counts and the following parameters (set using the builder pattern):

- k: Number of topics (i.e., cluster centers)
- optimizer: Optimizer to use for learning the LDA model, either EMLDAOptimizer or OnlineLDAOptimizer
- docConcentration: Dirichlet parameter for prior over documents' distributions over topics. Larger values encourage smoother inferred distributions.
- topicConcentration: Dirichlet parameter for prior over topics' distributions over terms (words). Larger values encourage smoother inferred distributions.
- maxIterations: Limit on the number of iterations.
- checkpointInterval: If using checkpointing (set in the Spark configuration), this parameter specifies the frequency with which checkpoints will be created. If maxIterations is large, using checkpointing can help reduce shuffle file sizes on disk and help with failure recovery.

All of spark.mllib's LDA models support:

- describeTopics: Returns topics as arrays of most important terms and term weights
- topicsMatrix: Returns a vocabSize by k matrix where each column is a topic

### Bisecting k-means

Bisecting K-means can often be much faster than regular K-means, but it will generally produce a different clustering.

Bisecting k-means is a kind of hierarchical clustering. Hierarchical clustering is one of the most commonly used method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- Agglomerative: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Divisive: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

Bisecting k-means algorithm is a kind of divisive algorithms. The implementation in MLlib has the following parameters:

- k: the desired number of leaf clusters (default: 4). The actual number could be smaller if there are no divisible leaf clusters.
- maxIterations: the max number of k-means iterations to split clusters (default: 20)
- minDivisibleClusterSize: the minimum number of points (if >= 1.0) or the minimum proportion of points (if < 1.0) of a divisible cluster (default: 1)
- seed: a random seed (default: hash value of the class name)

### Gaussian mixture

A Gaussian Mixture Model represents a composite distribution whereby points are drawn from one of k Gaussian sub-distributions, each with its own probability. The spark.mllib implementation uses the expectation-maximization algorithm to induce the maximum-likelihood model given a set of samples. The implementation has the following parameters:

- k is the number of desired clusters.
- convergenceTol is the maximum change in log-likelihood at which we consider convergence achieved.
- maxIterations is the maximum number of iterations to perform without reaching convergence.
- initialModel is an optional starting point from which to start the EM algorithm. If this parameter is omitted, a random starting point will be constructed from the data.

### Power iteration clustering (PIC)

Power iteration clustering (PIC) is a scalable and efficient algorithm for clustering vertices of a graph given pairwise similarities as edge properties, described in Lin and Cohen, Power Iteration Clustering. It computes a pseudo-eigenvector of the normalized affinity matrix of the graph via power iteration and uses it to cluster vertices. spark.mllib includes an implementation of PIC using GraphX as its backend. It takes an RDD of (srcId, dstId, similarity) tuples and outputs a model with the clustering assignments. The similarities must be nonnegative. PIC assumes that the similarity measure is symmetric. A pair (srcId, dstId) regardless of the ordering should appear at most once in the input data. If a pair is missing from input, their similarity is treated as zero. spark.mllib's PIC implementation takes the following (hyper-)parameters:

- k: number of clusters
- maxIterations: maximum number of power iterations

- initializationMode: initialization model. This can be either "random", which is the default, to use a random vector as vertex properties, or "degree" to use normalized sum similarities.

#### Streaming k-means

When data arrive in a stream, we may want to estimate clusters dynamically, updating them as new data arrive. spark.mllib provides support for streaming k-means clustering, with parameters to control the decay (or "forgetfulness") of the estimates. The algorithm uses a generalization of the mini-batch k-means update rule. For each batch of data, we assign all points to their nearest cluster, compute new cluster centers, then update each cluster

### 10.1.4 Regression

Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features'). The most common form of regression analysis is linear regression, in which a researcher finds the line (or a more complex linear function) that most closely fits the data according to a specific mathematical criterion.

- https://en.wikipedia.org/wiki/Regression_analysis

### Apache Spark

Table 5: Apache Spark based Regression Processors in Fire Insights

| Title | Description |
|---|---|
| Linear regression | LinearRegression analysis is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables. |
| Generalized linear regression | Contrasted with linear regression where the output is assumed to follow a Gaussian distribution, generalized linear models (GLMs) are specifications of linear models where the response variable Yi follows some distribution from the exponential family of distributions |
| Decision tree regression | Decision trees and their ensembles are popular methods for the machine learning tasks of classification and regression. Decision trees are widely used since they are easy to interpret, handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions. |
| Random forest regression | Random forests are ensembles of decision trees. Random forests combine many decision trees in order to reduce the risk of overfitting. |
| Gradient-boosted tree regression | Gradient-Boosted Trees (GBTs) are ensembles of decision trees. GBTs iteratively train decision trees in order to minimize a loss function. |
| Survival regression | Survival Analysis is a set of statistical tools, which addresses questions such as 'how long would it be, before a particular event occurs'; in other words we can also call it as a 'time to event' analysis. |
| Isotonic regression | Isotonic regression is the technique of fitting a free-form line to a sequence of observations under the following constraints: the fitted free-form line has to be non-decreasing everywhere, and it has to lie as close to the observations as possible. |

### Regression Algorithms in Apache Spark

https://spark.apache.org/docs/latest/ml-classification-regression.html#regression

- Linear regression
- Decision tree regression
- Random Forest regression
- Gradient-boosted tree regression
- Survival regression
- Isotonic regression

### Scikit Learn

Table 6: Scikit Learn based Regression Processors in Fire Insights

| Title | Description |
|---|---|
| Ridge regression | Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares |
| Lasso regression | The Lasso is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients, effectively reducing the number of features upon which the given solution is dependent. |
| Gradient Boosting regression | GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function |
| Random forest regression | A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default). |

### Regression Algorithms in Scikit Learn

https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression-and-classification

- Ridge regression
- Lasso regression
- Gradient Boosting regression
- Random Forest regression

### Linear Regression

The interface for working with linear regression models and model summaries is similar to the logistic regression case.

When fitting LinearRegressionModel without intercept on dataset with constant nonzero column by "l-bfgs" solver, Spark MLlib outputs zero coefficients for constant nonzero columns. This behavior is the same as R glmnet but different from LIBSVM.

### Generalized linear regression

Contrasted with linear regression where the output is assumed to follow a Gaussian distribution, generalized linear models (GLMs) are specifications of linear models where the response variable Yi follows some distribution from the exponential family of distributions.

Spark's GeneralizedLinearRegression interface allows for flexible specification of GLMs which can be used for various types of prediction problems including linear regression, Poisson regression, logistic regression, and others.

### Decision tree regression

Decision trees are a popular family of classification and regression methods.

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

### Random Forest Regression

Random forests are a popular family of classification and regression methods.

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

### Gradient - boosted Tree Regression

Gradient-boosted trees (GBTs) are a popular regression method using ensembles of decision trees.

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

### Survival Regression

In spark.ml, we implement the Accelerated failure time (AFT) model which is a parametric survival regression model for censored data. It describes a model for the log of survival time, so it's often called a log-linear model for survival analysis. Different from a Proportional hazards model designed for the same purpose, the AFT model is easier to parallelize because each instance contributes to the objective function independently.

### Isotonic Regression

Isotonic regression or monotonic regression is the technique of fitting a free-form line to a sequence of observations under the following constraints: the fitted free-form line has to be non-decreasing (or non-increasing) everywhere, and it has to lie as close to the observations as possible.

Isotonic regression has applications in statistical inference. For example, one might use it to fit an isotonic curve to the means of some set of experimental results when an increase in those means according to some particular ordering is expected. A benefit of isotonic regression is that it is not constrained by any functional form, such as the linearity imposed by linear regression, as long as the function is monotonic increasing.

Another application is nonmetric multidimensional scaling, where a low-dimensional embedding for data points is sought such that order of distances between points in the embedding matches order of dissimilarity between points. Isotonic regression is used iteratively to fit ideal distances to preserve relative dissimilarity order.

Software for computing isotone (monotonic) regression has been developed for the R statistical package, the Stata statistical package and the Python programming language

### 10.1.5 Classification

In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs.

- https://en.wikipedia.org/w/index.php?search=Clssification+in+machine+learning&title=Special%3ASearch&go=Go&ns0=1

#### Apache Spark MLlib

Table 7: Apache Spark based Classification Processors in Fire Insights

| Title | Description |
|---|---|
| Logistic Regression | Logistic regression is a popular method to predict a categorical response. It is a special case of Generalized Linear models that predicts the probability of the outcomes. |
| Decision tree classifier | Decision trees and their ensembles are popular methods for the machine learning tasks of classification and regression. Decision trees are widely used since they are easy to interpret, handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions. |
| Random forest classifier | Random forests are ensembles of decision trees. Random forests combine many decision trees in order to reduce the risk of overfitting. |
| Gradient-boosted tree classifier | Gradient-Boosted Trees (GBTs) are ensembles of decision trees. GBTs iteratively train decision trees in order to minimize a loss function. |
| Multilayer perceptron classifier | Multilayer perceptron classifier (MLPC) is a classifier based on the feedforward artificial neural network. |
| Naive Bayes | Naive Bayes classifiers are a family of simple probabilistic, multiclass classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between every pair of features. |

#### Classification Algorithms in Spark MLlib

https://spark.apache.org/docs/latest/ml-classification-regression.html#classification

- Logistic Regression
- Decision tree classifier
- Random forest classifier
- Gradient-boosted tree classifier
- Multilayer perceptron classifier
- Linear Support Vector Machine
- One-vs-Rest classifier
- Naive Bayes

### Scikit Learn

Table 8: Scikit Learn based Classification Processors in Fire Insights

| Title | Description |
|---|---|
| Logistic Regression Classifier | In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. |
| Gradient Boosting classifier | GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n_classes _ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. |
| Random forest classifier | A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default). |

### Classification Algorithms in Scikit Learn

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

- Logistic Regression
- Gradient-boosting classifier
- Random Forest classifier

Fire Insights provides processors for the above Algorithms.

### Logistic Regression

Logistic regression is a popular method to predict a categorical response. It is a special case of Generalized Linear models that predicts the probability of the outcomes. In spark.ml logistic regression can be used to predict a binary outcome by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial logistic regression. Use the family parameter to select between these two algorithms, or leave it unset and Spark will infer the correct variant.

Multinomial logistic regression can be used for binary classification by setting the family param to "multinomial". It will produce two sets of coefficients and two intercepts.

When fitting LogisticRegressionModel without intercept on dataset with constant nonzero column, Spark MLlib outputs zero coefficients for constant nonzero columns. This behavior is the same as R glmnet but different from LIBSVM.

### Decision tree classifier

Decision tree learning is one of the predictive modeling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value.

Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making).

### Random forest classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

### Gradient-boosted tree classifier

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The idea of gradient boosting originated in the observation that boosting can be interpreted as an optimization algorithm on a suitable cost function. Explicit regression gradient boosting algorithms were subsequently developed simultaneously with the more general functional gradient boosting perspective. It later introduced the view of boosting algorithms as iterative functional gradient descent algorithms. That is, algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

### Multilayer perceptron classifier

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to refer to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation). Multilayer perceptrons are sometimes colloquially referred to as “vanilla” neural networks, especially when they have a single hidden layer.

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

### Naive Bayes

In machine learning, naïve Bayes classifiers are a family of simple “probabilistic classifiers” based on applying Bayes’ theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models.

It remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression,which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

## 10.1.6 Prediction

Prediction is to identify data points purely on the description of another related data value. It is not necessarily related to future events but the used variables are unknown. Prediction derives the relationship between a thing you know and a thing you need to predict for future reference.

Prediction refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether or not a customer will churn in 30 days. The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.

The word "prediction" can be misleading. In some cases, it really does mean that you are predicting a future outcome, such as when you're using machine learning to determine the next best action in a marketing campaign. Other times, though, the "prediction" has to do with, for example, whether or not a transaction that already occurred was fraudulent. In that case, the transaction already happened, but you're making an educated guess about whether or not it was legitimate, allowing you to take the appropriate action.

### What is Prediction?

- Predicting the identity of one thing based purely on the description of another, related thing
- Not necessarily future events, just unknowns
- Based on the relationship between a thing that you can know and a thing you need to predict

### Why are Predictions Important?

Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things – customer churn likelihood, possible fraudulent activity, and more. These provide the business with insights that result in tangible business value. For example, if a model predicts a customer is likely to churn, the business can target them with specific communications and outreach that will prevent the loss of that customer.

### Predictor => Predicted

- When building a predictive model, you have data covering both
- When using one, you have data describing the predictor and you want it to tell you the predicted value

### Usual Examples

- Predicting levels of sales that will result from a price change or advert.
- Predicting whether or not it will rain based on current humidity
- Predicting the colour of a pottery glaze based on a mixture of base pigments
- Predicting how far up the charts a single will go
- Predicting how much revenue a book of debt will bring

### Techniques

Most prediction techniques are based on mathematical models:

- Simple statistical models such as regression
- Non-linear statistics such as power series
- Neural networks, RBFs, etc
- All based on fitting a curve through the data, that is, finding a relationship from the predictors to the predicted

## 10.1.7 Model Evaluation

Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data.

### Evaluation Processors in Fire Insights

Table 9: Apache Spark based Evaluation Processors in Fire Insights

| Title | Description |
|---|---|
| NodeRegressionEvaluator | Evaluator for regression, which expects two input columns: prediction and label. Regression analysis is used when predicting a continuous output variable from a number of independent variables. |
| NodeBinaryClassificationEvaluator | Evaluator for binary classification, which expects two input columns: rawPrediction and label. Binary classifiers are used to separate the elements of a given dataset into one of two possible groups (e.g. fraud or not fraud) and is a special case of multiclass classification. |
| NodeMulticlassClassificationEvaluator | Evaluator for multiclass classification, which expects two input columns: score and label. A multiclass classification describes a classification problem where there are M>2 possible labels for each data point (the case where M=2 is the binary classification problem) |

- https://heartbeat.fritz.ai/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f

Machine learning continues to be an increasingly integral component of our lives, whether we're applying the techniques to research or business problems. Machine learning models ought to be able to give accurate predictions in order to create real value for a given organization.

While training a model is a key step, how the model generalizes on unseen data is an equally important aspect that should be considered in every machine learning pipeline. We need to know whether it actually works and, consequently, if we can trust its predictions. Could the model be merely memorizing the data it is fed with, and therefore unable to make good predictions on future samples, or samples that it hasn't seen before?

In this article, we explain the techniques used in evaluating how well a machine learning model generalizes to new, previously unseen data. We'll also illustrate how common model evaluation metrics are implemented for classification and regression problems using Python.

### Model Evaluation Techniques

The above issues can be handled by evaluating the performance of a machine learning model, which is an integral component of any data science project.

Methods for evaluating a model's performance are divided into 2 categories: namely, holdout and Cross-validation. Both methods use a test set (i.e data not seen by the model) to evaluate model performance. It's not recommended to use the data we used to build the model to evaluate it. This is because our model will simply remember the whole training set, and will therefore always predict the correct label for any point in the training set. This is known as overfitting.

#### Holdout

The purpose of holdout evaluation is to test a model on different data than it was trained on. This provides an unbiased estimate of learning performance.

In this method, the dataset is randomly divided into three subsets:

1)Training set is a subset of the dataset used to build predictive models.

2)Validation set is a subset of the dataset used to assess the performance of the model built in the training phase. It provides a test platform for fine-tuning a model's parameters and selecting the best performing model. Not all modeling algorithms need a validation set.

3)Test set, or unseen data, is a subset of the dataset used to assess the likely future performance of a model. If a model fits to the training set much better than it fits the test set, overfitting is probably the cause.

The holdout approach is useful because of its speed, simplicity, and flexibility. However, this technique is often associated with high variability since differences in the training and test dataset can result in meaningful differences in the estimate of accuracy.

#### Cross-Validation

Cross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis.

The most common cross-validation technique is k-fold cross-validation, where the original dataset is partitioned into k equal size subsamples, called folds. The k is a user-specified number, usually with 5 or 10 as its preferred value. This is repeated k times, such that each time, one of the k subsets is used as the test set/validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get the total effectiveness of our model.

For instance, when performing five-fold cross-validation, the data is first partitioned into 5 parts of (approximately) equal size. A sequence of models is trained. The first model is trained using the first fold as the test set, and the remaining folds are used as the training set. This is repeated for each of these 5 splits of the data and the estimation of accuracy is averaged over all 5 trials to get the total effectiveness of our model. As can be seen, every data point gets to be in a test set exactly once and gets to be in a training set k-1 times. This significantly reduces bias, as we're using most of the data for fitting, and it also significantly reduces variance, as most of the data is also being used in the test set. Interchanging the training and test sets also adds to the effectiveness of this method.

- https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

#### Model Evaluation in Fire Insights

### 10.1.8 Model Persistence

Save / Load Model allows you to save your model to files and load them later in order to make predictions.

Fire Insights allows you to save the ML Model created. The ML Models can be loaded in the same or other workflows to be used for scoring. The ML Models can also be downloaded from HDFS Browse Page.

The ML models can be saved into the following locations:

- HDFS : when Fire Insights is connected to a Hadoop Cluster
- S3 : when Fire is configured and connected to AWS.
- Local Machine FileSystem : when Fire is running in local mode

In order to save onto S3, the model path can be provided as `s3://models/priceprediction`

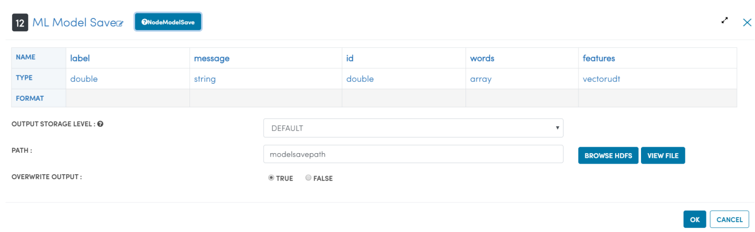
## Persisting SparkML Models

### Spark ML Models

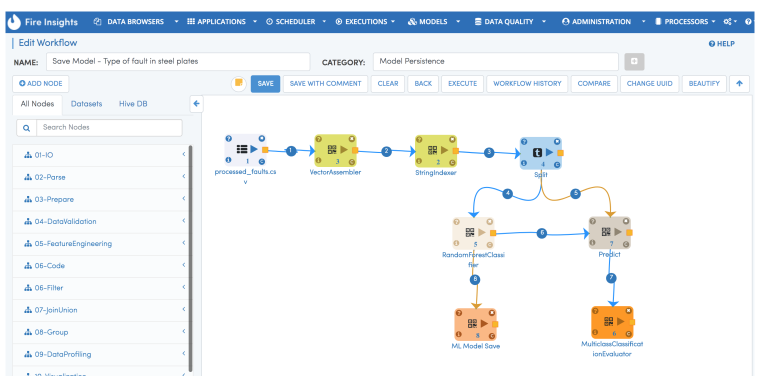
Spark ML models are saved into a directory with multiple files in it. Fire Insights has processors for saving and loading the Spark ML models.

### Save Model processor

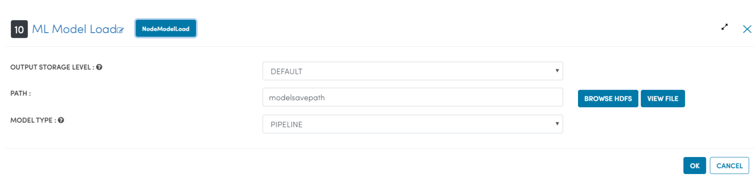
NodeModelSave processor, saves the given Apache Spark ML model at the given location.



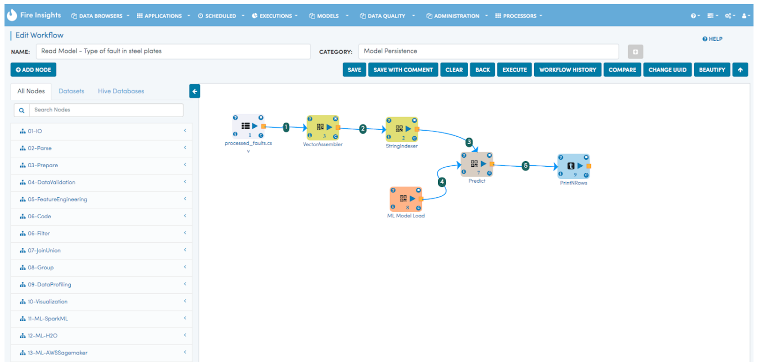

### ML Save Workflow




### Load Model processor



### ML Load Workflow



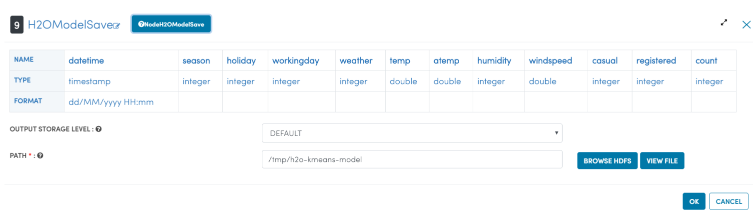
### Persisting H2O Models

### H2O Models

H2O Models can be saved in binary format or in MOJO format. Fire Insights has processors for saving and reading them back.

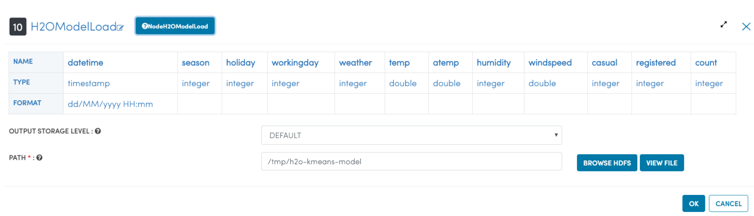
### Save H2o Model processor

H2OModelSave Processor saves the H2O model at the specified path in the binary format.




### Load H2o Model processor

H2OModelLoad Processor loads the H2O model in binary format from the specified path.

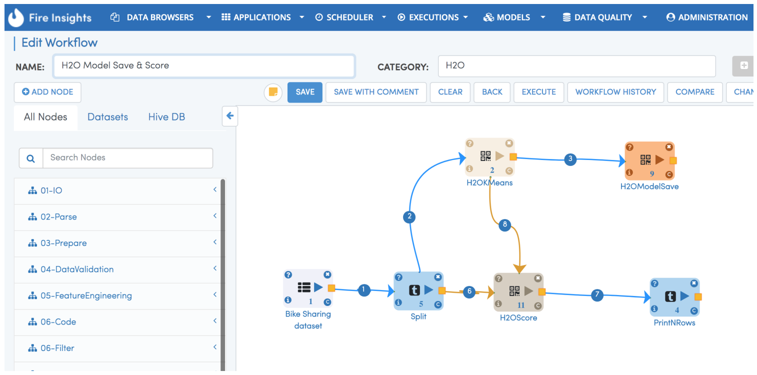



More details of saving and loading the H2O Models is available here:

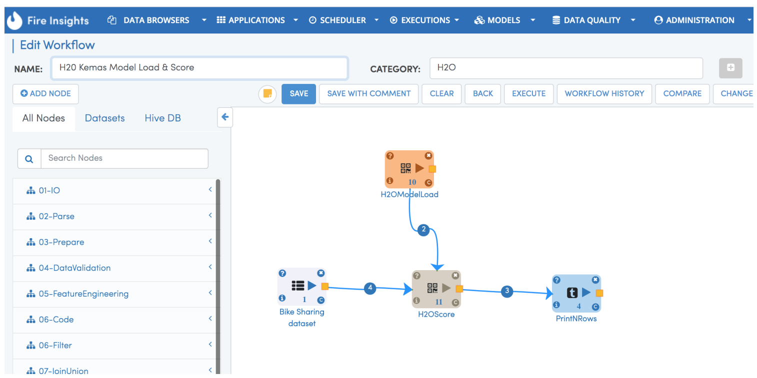
http://docs.h2o.ai/h2o/latest-stable/h2o-docs/save-and-load-model.html

### Save and Load H2O Workflow

Below is a workflow, which saves the generated H2O model on the file system.




Below is a workflow, which load back the saved model and used in batch scoreing.




### Persisting Scikit Learn Models

Scikit-Learn models are persisted with pickle. Fire Insights has processors for saving and loading the pickle files.

More details of the pickle format is available here:

https://scikit-learn.org/stable/modules/model_persistence.html

## 10.1.9 Model Serving

Fire Insights allows you to save your models. These models can be saved to:

- HDFS : when running on a Hadoop Cluster
- S3 : when running on AWS
- ADLS : when running on Azure
- Local file system : when running on your laptop or independent machine

Once these models are saved, they can be served in various ways.

### Scoring with Workflows

Fire Insights enables you to build workflows. Workflows provide for reading data, transforming them and also creating machine learning models. Fire Insights supports a number of ML frameworks including Scikit Learn, H2O, Spark ML, Keras etc.

Models built with the workflows can be saved onto the File System. The models can then be scored with another workflow.

### Data Preparation and Scoring Environments

The workflows built with Fire Insights can run on a variety of environments. These include:

- Standalone machine
- AWS - EMR
- Azure - HDInsights
- Databricks
- Cloudera

In any of these environments, Fire Insights does not need to be installed for model scoring. When running on Standalone machine, scoring can be performed with running java/python using the supplied jar/wheel files and the workflow json.

When running on clusters, scoring can be performed with spark-submit using the supplied jar/wheel files and the workflow json.

### Workflow Patterns for Scoring Models

There are a few patterns by which Fire Insights enables Data Preparation/Feature Engineering and Model Scoring.

- One workflow for Data Preparation/Feature Engineering, another for Model Training and the third for Model Scoring
- One workflow for Data Preparation/Feature Engineering plus Model Training. And another workflow for Data Preparation/Feature Engineering plus Model Scoring.

### Using 3 Workflows

In this pattern, one workflow is built to read in the input datasets, perform Data Preparation and also Feature Engineering. This workflow prepares the input datasets to be used for Training and also Scoring and saves it to the File System.

The second workflow reads in the prepared data, builds the model and then save it to the File System.

The third workflow also reads in the prepared data, reads in the ML model and then scores the input data. The result of scoring can be saved to the File System, Relational Database, Cassandra, MongoDB, HIVE etc.

### Using 2 workflows

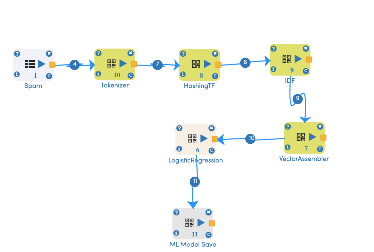
In this pattern, one workflow is built to read in the input datasets, perform Data Preparation/Feature Engineering and then finally build the ML Model.

For the second workflow, the first workflow is cloned with one click, and the model nodes are removed from the workflow. They are replaced with nodes which read in the model and then score the datasets.

### Serving Spark MLlib Models

Fire Insights creates Apache Spark MLlib models. These models get saved as files on the File System.

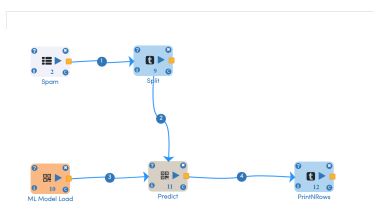
NoveModelSave saves the Spark ML models as files. It uses the Spark interfaces to save the model.




Once the SparkML model is saved, they can be loaded and used in scoring. Fire Insights enables saving both Spark ML models and pipelines.

### Batch Model Scoring:

By using NodeModelLoad & selecting the particular type of model to be loaded, the model would be loaded in the workflow and it can be used for scoring the input data.




### Online Scoring with Kafka and Spark Streaming:

Scalable messaging platform like Kafka to send newly acquired data to a long running Spark Streaming process. The Spark process can then make a new prediction based on the new data.

### Serving H2O Models

H2O allows you to persist the models you have built to either a Plain Old Java Object (POJO) or a Model ObJect, Optimized (MOJO).

Fire Insights has the following processors for persisting the H2O Models.

- H2OMojoSave
- H2OModelSave

Once the H2O model is saved, they can be used for serving.

H2O-generated MOJO and POJO models are intended to be easily embeddable in any Java environment. The only compilation and runtime dependency for a generated model is the h2o-genmodel.jar file produced as the build output of these packages.

We can use our H2OModelLoad or H2OMojoLoad to make a batch prediction, real-time prediction using Spark Streaming, Kafka or Storm. Or you can expose your model as a REST API.

https://h2o-release.s3.amazonaws.com/h2o/rel-ueno/2/docs-website/h2o-docs/pojo-quick-start.html

### Serving H2O MOJO models

The below page on the H2O website gives details on serving a MOJO model.

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/productionizing.html#step-2-compile-and-run-the-mojo

### Serving H2O POJO models

The details for serving a POJO models is described in this page.

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/productionizing.html#building-a-pojo

```
import java.io.*;
import hex.genmodel.easy.RowData;
import hex.genmodel.easy.EasyPredictModelWrapper;
import hex.genmodel.easy.prediction.*;

public class main {
 private static String modelClassName = "gbm_pojo_test";

 public static void main(String[] args) throws Exception {
   hex.genmodel.GenModel rawModel;
   rawModel = (hex.genmodel.GenModel) Class.forName(modelClassName).newInstance();
   EasyPredictModelWrapper model = new EasyPredictModelWrapper(rawModel);

   RowData row = new RowData();
   row.put("Year", "1987");
   row.put("Month", "10");
   row.put("DayofMonth", "14");
   row.put("DayOfWeek", "3");
   row.put("CRSDepTime", "730");
   row.put("UniqueCarrier", "PS");
   row.put("Origin", "SAN");
   row.put("Dest", "SFO");

   BinomialModelPrediction p = model.predictBinomial(row);
   System.out.println("Label (aka prediction) is flight departure delayed: " + p.
↪label);
   System.out.print("Class probabilities: ");
   for (int i = 0; i < p.classProbabilities.length; i++) {
     if (i > 0) {
       System.out.print(",");
     }
     System.out.print(p.classProbabilities[i]);
   }
   System.out.println("");
```

(continues on next page)

(continued from previous page)

```
  }
}
```

Useful links:

https://medium.com/spikelab/building-a-machine-learning-application-using-h2o-ai-67ce3681df9c

### Serving AWS SageMaker models

When the SageMaker models are built in Fire Insights, SageMaker automatically provides a REST endpoint for online scoring of the models.

The details for it are available here:

- https://aws.amazon.com/blogs/machine-learning/creating-a-machine-learning-powered-rest-api-with-amazon-api-gateway-mapp
- https://aws.amazon.com/blogs/machine-learning/call-an-amazon-sagemaker-model-endpoint-using-amazon-api-gateway-and-av

### Serving Scikit Learn Models

Fire Insights provides the following processors for persisting the Scikit Learn models as pickle files:

- SaveAsPickle

Once the Scikit Learn model is saved, they can be used for serving.

The details for Scikit Learn Model Persistence is available here:

- https://scikit-learn.org/stable/modules/model_persistence.html

### Serving Tensorflow Models

Fire Insights provides the following processors for persisting the Tensorflow models:

- NodeSaveKerasModel
- NodeLoadKerasModel

### Integration with MLflow

Fire Insights integrates deeploy with Apache MLflow.

Fire Insights can be configured to output the models to MLflow.

CHAPTER 11

# Time Series

## 11.1 Time Series Analysis

Time series analysis is a statistical technique that deals with time series data, or trend analysis. Time series data means that data is in a series of particular time periods or intervals.

https://www.statisticssolutions.com/time-series-analysis/

Fire Insights provides a number of features for Time Series Analysis.

### 11.1.1 Time Series Feature Engineering

Fire Insights provides a number of Processors for Feature Engineering of Time Series Data. These include:

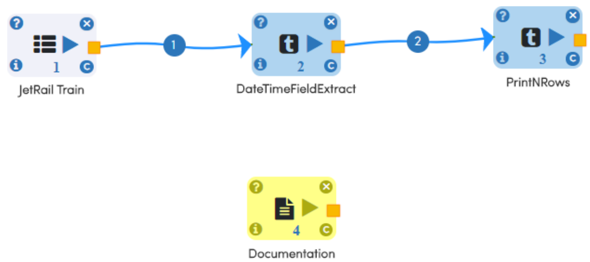
Table 1: Update New features where needed

| Features | Description |
|---|---|
| DateTimeFieldExtract | Extracts year, month, day of month, hour, minute, second and week of year from timestamp/date columns |
| Days to holiday | Days remaining for next holiday |
| Days from holiday | Days passed after holiday |
| Time-segmentation | Divide data in morning, afternoon, evening, night to get more idea about time based pattern |
| MovingWindowingFunctions | Calculates the moving values using the given function |
| WindowingAnalytics | Implements window functions is mainly through the operators rolling and expanding |
| Exponential Moving Average (EMA) | The Exponential Moving Average (EMA) assigns a greater weight to the most recent price observations. While it assigns lesser weight to past data, it is based on a recursive formula that includes in its calculation all the past data in our price series. |

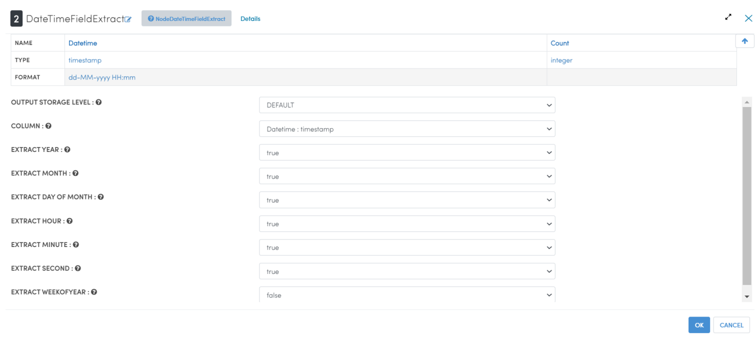
#### DateTimeFieldExtract

Below is the sample workflows which contains `DateTimeFieldExtract` processor in Fire Insights.

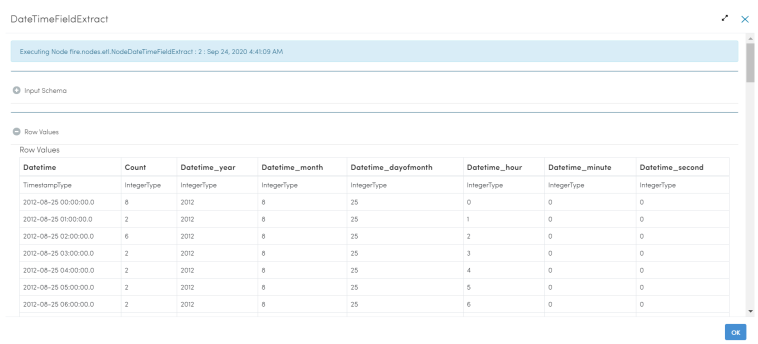
It reads the JetRail Train dataset & use DateTimeFieldExtract processor which create New DataFrame by extracting Date & Time field and print the result.




DateTimeFieldExtract processor Configuration:




Output result of `DateTimeFieldExtract` processor:



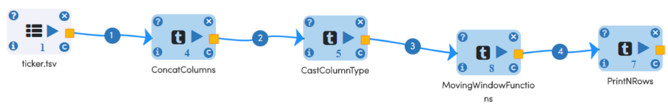

## MovingWindowingFunctions

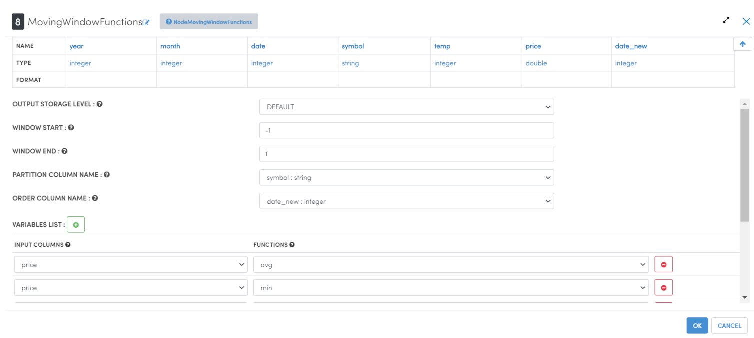
Below is the sample workflows which contains `MovingWindowingFunctions` processor in Fire Insights.

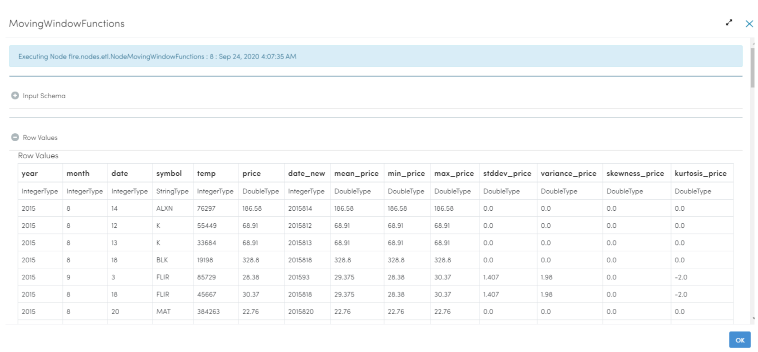
It reads the ticker dataset, concatenate the input column, casting specified column to new data type, use MovingWindowingFunctions processor which calculates the moving value of selected function of input column and print the result.

MovingWindowingFunctions processor Configuration:

Output result of `MovingWindowingFunctions` processor:


ticker.tsv
ConcatColumns
CastColumnType
MovingWindowFunctions
PrintNRows


MovingWindowFunctions
OUTPUT STORAGE LEVEL :
DEFAULT
WINDOW START :
-1
WINDOW END :
1
PARTITION COLUMN NAME :
symbol : string
ORDER COLUMN NAME :
date_new : integer
VARIABLES LIST :
INPUT COLUMNS
FUNCTIONS
price
avg
price
min
OK
CANCEL


MovingWindowFunctions
Executing Node fire.nodes.etl.NodeMovingWindowFunctions : 8 : Sep 24, 2020 4:07:35 AM
Input Schema
Row Values
Row Values
year month date symbol temp price date_new mean_price min_price max_price stddev_price variance_price skewness_price kurtosis_price
IntegerType IntegerType IntegerType StringType IntegerType DoubleType IntegerType DoubleType DoubleType DoubleType DoubleType DoubleType DoubleType DoubleType
2015 8 14 ALXN 76297 186.58 2015814 186.58 186.58 186.58 0.0 0.0 0.0 0.0
2015 8 12 K 55449 68.91 2015812 68.91 68.91 68.91 0.0 0.0 0.0 0.0
2015 8 13 K 33684 68.91 2015813 68.91 68.91 68.91 0.0 0.0 0.0 0.0
2015 8 18 BLK 19198 328.8 2015818 328.8 328.8 328.8 0.0 0.0 0.0 0.0
2015 9 3 FLIR 85729 28.38 201593 29.375 28.38 30.37 1.407 1.98 0.0 -2.0
2015 8 18 FLIR 45667 30.37 2015818 29.375 28.38 30.37 1.407 1.98 0.0 -2.0
2015 8 20 MAT 384263 22.76 2015820 22.76 22.76 22.76 0.0 0.0 0.0 0.0
OK

### 11.1.2 Time Series Visualizations

Fire Insights provides a number of Processors for the visualization of the time series data.

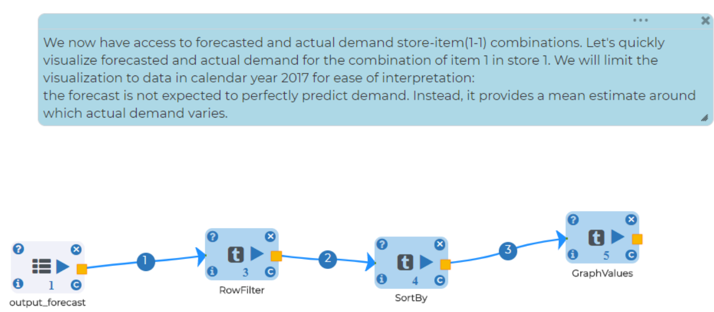
Table 2: Update New features where needed

| Charts | Description |
|---|---|
| Line | Perfect for series of data points to form a continuous line. Example - Represent Daily sales data. |
| Bar | Bar charts are a fundamental visualization for comparing values between groups of data. Best way to represent Categorical data. |
| Scatter | Scatter plots are used to observe relationships between variables. |
| Histogram | Histograms are a type of graph that shows the distribution of a dataset. They graph the percentage or the number of instances of different categories. |
| Pie | Illustrate the percentage breakdown of a small number of data points, then they can be very effective. |

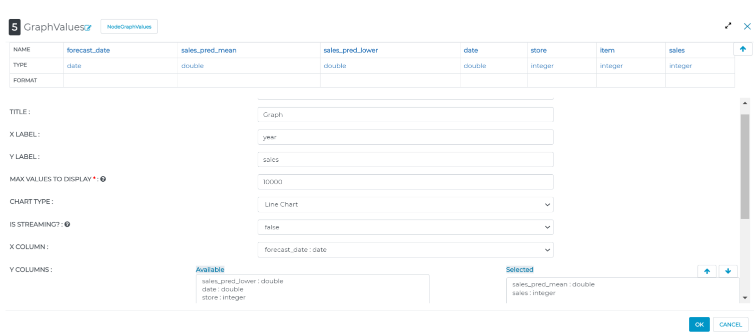
#### Charts : LineChart

Perfect for series of data points to form a continuous line. Example - Represent Daily sales data

Below is the sample workflows which contains `Time Series data` and visualize using line chart in Fire Insights.

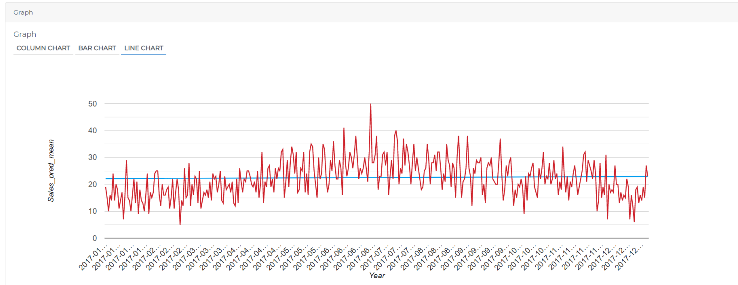


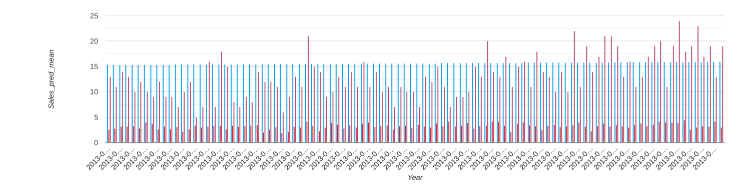

Configurations for visualization processors in Fire Insight: * Set number of columns want to represent on y axis with respect to x axis * Set chart type based on data type



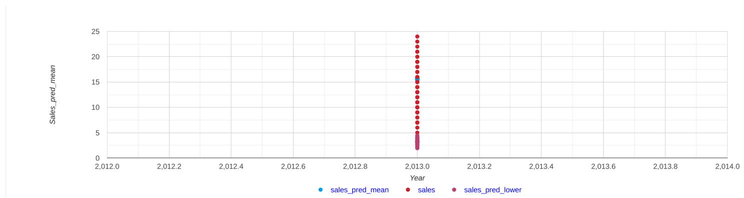

Output result of `Visualization` processor:

#### Charts : BarChart




### Charts : Scatter




Row Values

Row Values

| forecast_date | sales_pred_mean | sales | sales_pred_lower |
|---|---|---|---|
| DateType | DoubleType | IntegerType | DoubleType |
| 2013-01-01 | 15.322 | 13 | 2.487 |
| 2013-01-02 | 15.328 | 11 | 2.771 |
| 2013-01-03 | 15.335 | 14 | 3.16 |
| 2013-01-04 | 15.341 | 13 | 3.121 |
| 2013-01-05 | 15.347 | 10 | 3.235 |
| 2013-01-06 | 15.354 | 12 | 2.803 |
| 2013-01-07 | 15.36 | 10 | 3.949 |
| 2013-01-08 | 15.366 | 9 | 3.739 |

## 11.1.3 Time Series Modeling

Fire Insights provides a number of Processors for Time Series Modeling. These include:

Table 3: Update New features where needed

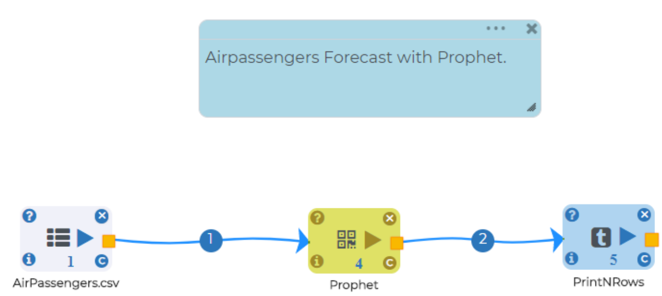
| Models | Description |
|---|---|
| Prophet | Prophet is a procedure for predicting time series data based on an additive or multiplicative model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It is best for time series that have strong seasonal effects and several seasons of historical data. Prophet is robust model to missing data and shifts in the trend, and able to handles outliers. For more: https://facebook.github.io/prophet/ |
| Arima | ARIMA is a model which is used for predicting future trends on a time series data. It is model that form of regression analysis. For more: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average |
| XGBoost | XGBoost is gradient boosting algorithm. It is also known as 'regularized boosting' technique - seeks a goot bias-variant trade-off to reduce overfitting allows cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. For more: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/xgboost.html#limitations |
| LSTM | LSTM is special kind of recurrent neural network that is capable of learning long term dependencies in data. This is achieved because the recurring module of the model has a combination of four layers interacting with each other. This is a great benefit in time series forecasting, where classical linear methods can be difficult to adapt to multivariate or multiple input forecasting problems. For more: https://www.tensorflow.org/tutorials/structured_data/time_series |

### Prophet

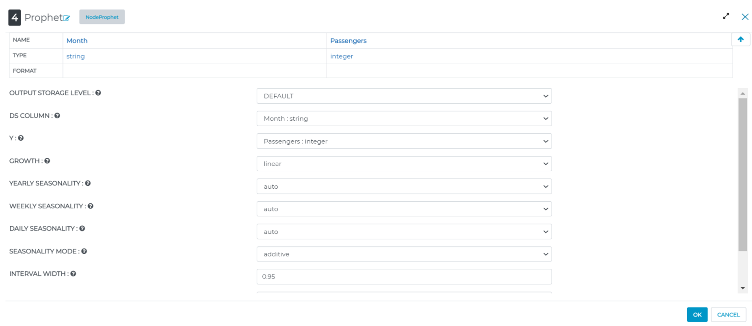
Below is the sample workflows which contains `Prophet` processor in Fire Insights.

> **Equation - y(t)=g(t)+s(t)+h(t)+t**,
>
> where:
>
> - Trend g(t): models non-periodic changes
> - Seasonality s(t): represents periodic changes
> - Holidays component h(t): contributes information about holidays and events

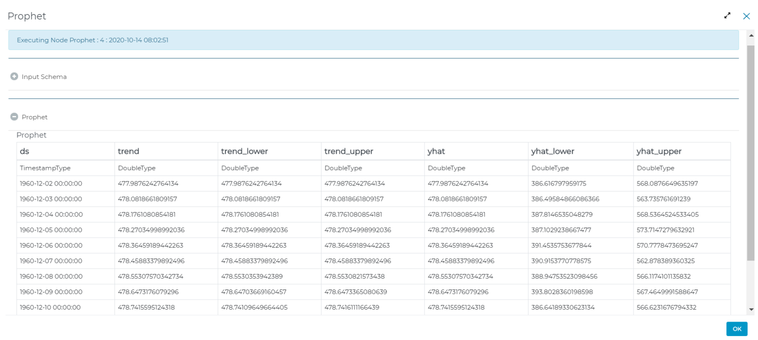
It reads the AirPassengers dataset & use Prophet processor which forecasting of univariate time series data and print the result.




Prophet processor Configuration:




Output result of `Prophet` processor:




## ARIMA

Below is the sample workflows which contains `ARIMA` processor in Fire Insights.

- AR (Autoregression): A changing variable that regresses on its own lagged/prior values.
- I (Integrated): Differencing of raw observations to allow for the time series to become stationary
- MA (Moving average): Dependency between an observation and a residual error from a moving average model

In terms of y, the general forecasting equation is:

**ŷt = $\mu$ + 1 yt-1 +……… + p yt-p — $\theta$1et-1 -……… - $\theta$qet-q**,

where: * $\mu \rightarrow$ constant

- 1 yt-1 +…+ p yt-p $\rightarrow$ AR terms (lagged values of y)
- $\theta$1et-1 -……… - $\theta$qet-q $\rightarrow$ MA terms (lagged errors)

It reads the AirPassengers dataset & use ARIMA processor which Forecast the airline passengers count, generate a new column with unique index/value for each row in dataset and print the result.

ARIMA processor Configuration:

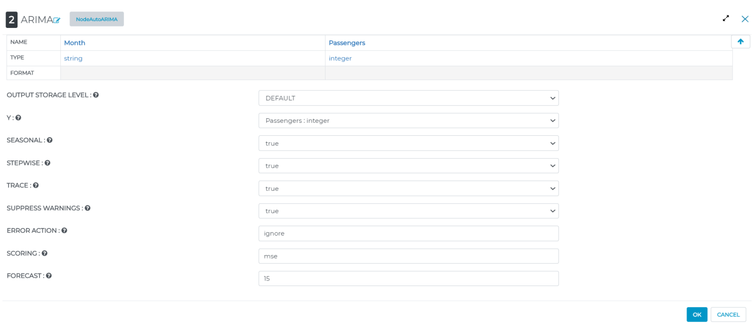
Output result of `ARIMA` processor:

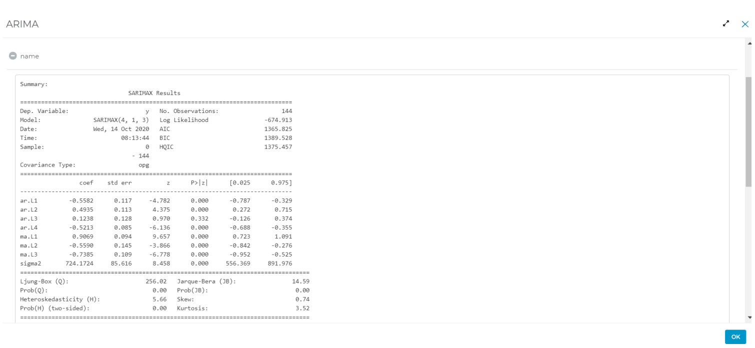
## H2OXGBoost

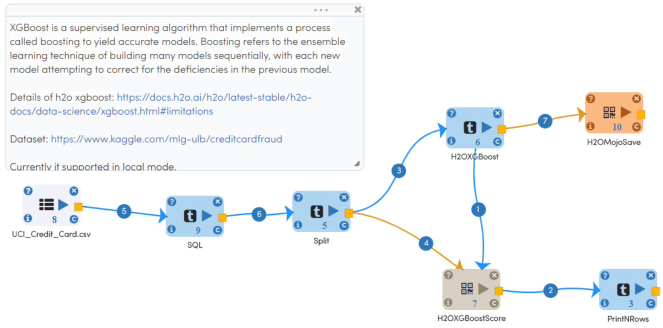
Below is the sample workflows which contains `H2OXGBoost` processor in Fire Insights.

It reads the UCI_Credit_Card dataset & use H2OXGBoost processor supervised learning algorithm that implements a process called boosting to yield accurate models and save the model in s3 location.

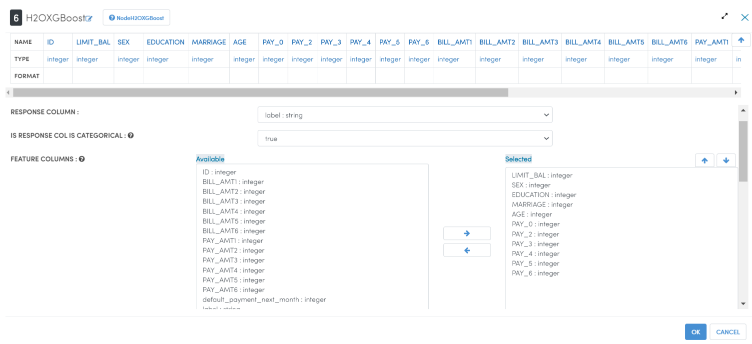
time-series/../../_assets/ml_userguide/arima.PNG



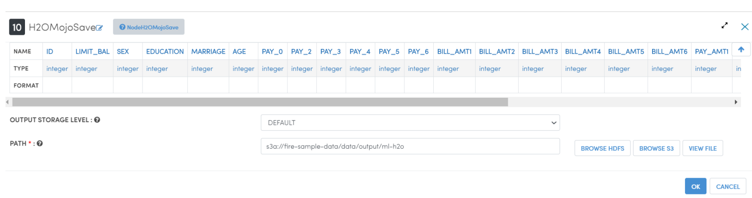





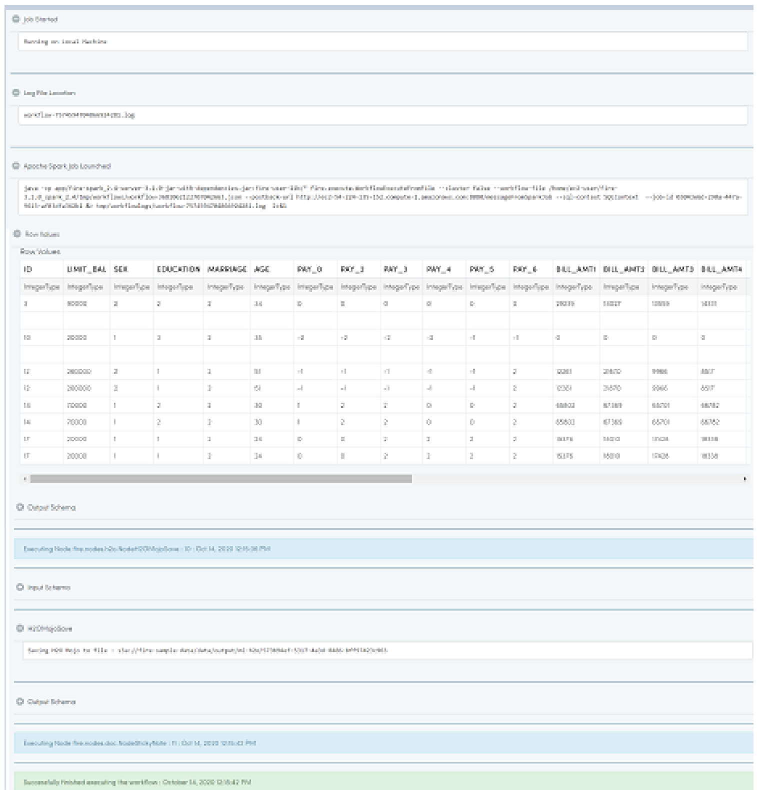

H2OXGBoost processor Configuration:

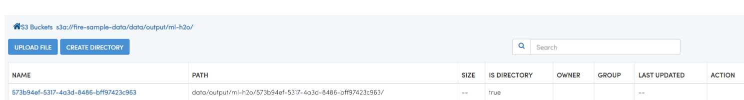



H2OMojoSave processor Configuration:




On successful submission of the job, the model get saved to specified locations, you can just view the model at specified location.





# CHAPTER 12

# Tutorials

## 12.1 Tutorials

### 12.1.1 Reading - Writing Data

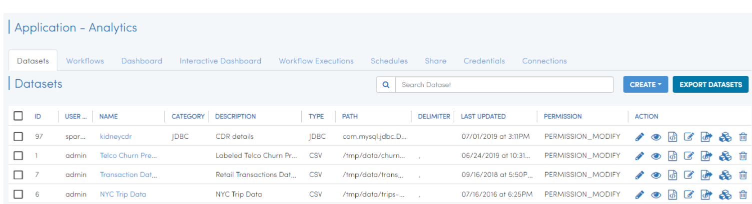
#### Creating Dataset for CSV Files

When working with data in Fire Insights, the first step is to create a dataset that you plan to process subsequently. Dataset is a wrapper around your data which makes it easy to handle it in Sparkflows workbench.

When datasets are created, Fire Insights automatically infers the schema using Spark-CSV library from Databricks.

#### Datasets List

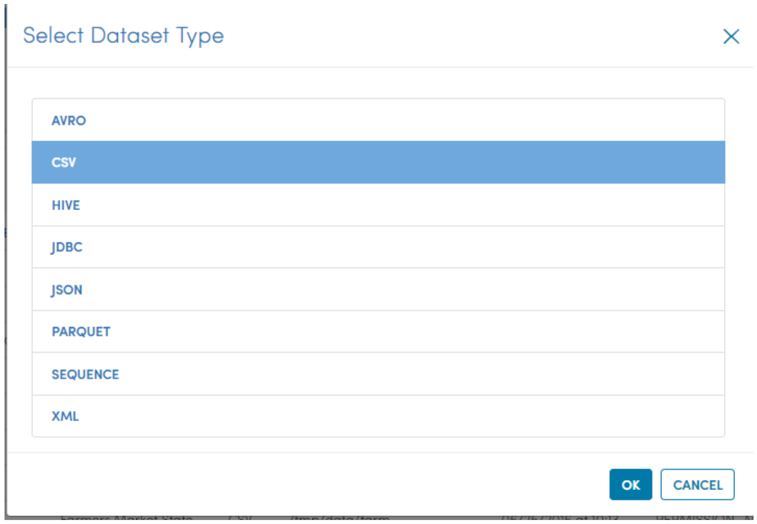
When you open any application, all existing Datasets specific to the application are displayed in the Datasets tab.




#### Dataset Creation

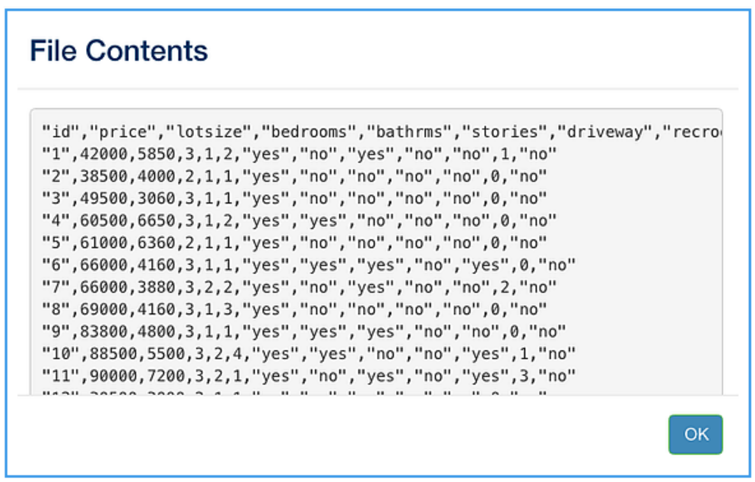
#### Choose type of Dataset to Create

Navigate to the "Datasets" tab in your application. Click on the "Create" button and choose "Dataset". In the pop-up choose "CSV" and then click "OK".




### Dataset Details

Clicking "OK" will take you to Dataset Details page where you can enter information about your dataset. In the screenshot below, we create a dataset from a housing.csv file. It is a comma separated file with a header row specifying the names of the various columns.

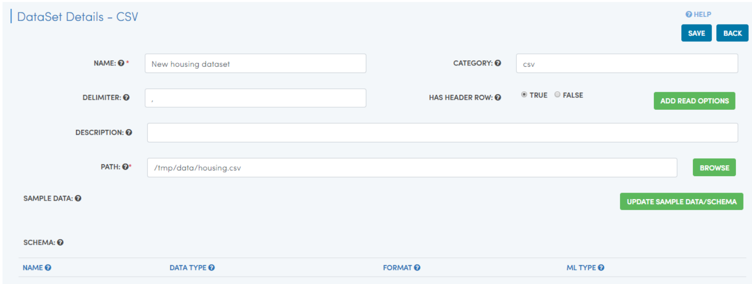


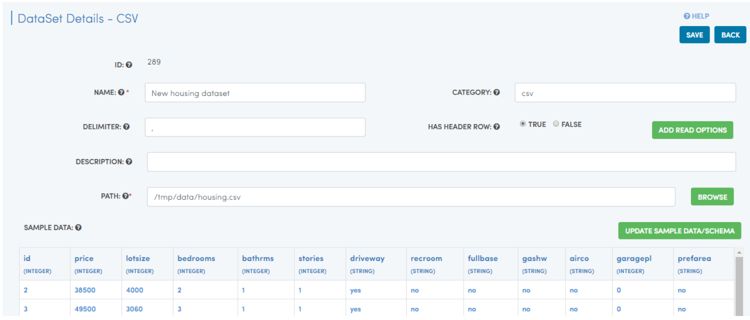

For the housing.csv file, we will fill in the required fields as below.

We specified a name for the dataset we are creating. 'Header' is set to true indicating that the file has a header row, field delimiter is comma and we also specified the path to the file.

### Update Sample data/schema

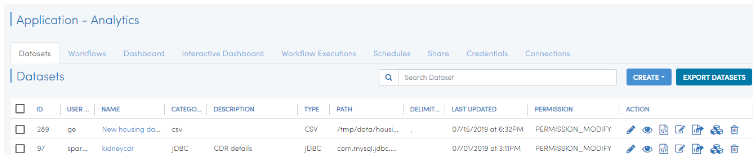
Once we have specified the above, we hit the 'Update Sample data/schema' button. This brings up the sample data, infers the schema and displays it. We can change the column names and also the data types. Format column is used for specifying the format for date/time fields.





### Save the Dataset

Clicking the 'Save' button creates the new dataset. The dataset is now ready for use in any workflow within the specific application.

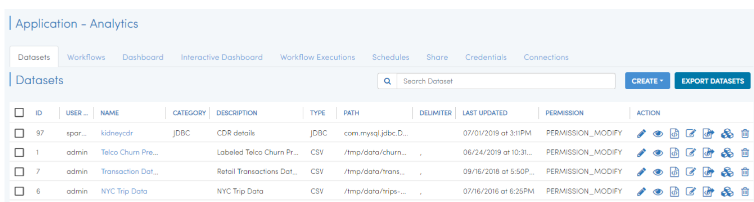



### Creating Dataset for AVRO Files

When working with data in Fire Insights, the first step is to create a dataset that you plan to process subsequently. Dataset is a wrapper around your data which makes it easy to handle it in Sparkflows workbench.

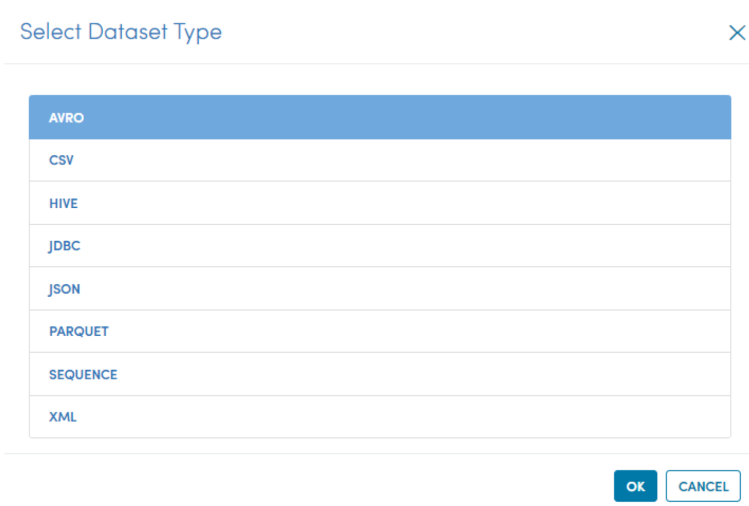
When datasets are created, Fire Insights automatically infers the schema using Spark-Avro library.

### Datasets

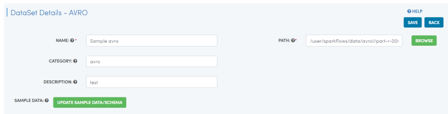



### Dataset Creation

Navigate to the "Datasets" tab in your application where you want to create a new dataset. Click on the "Create" button and choose "Dataset". In the pop-up choose "AVRO" and then click "OK".

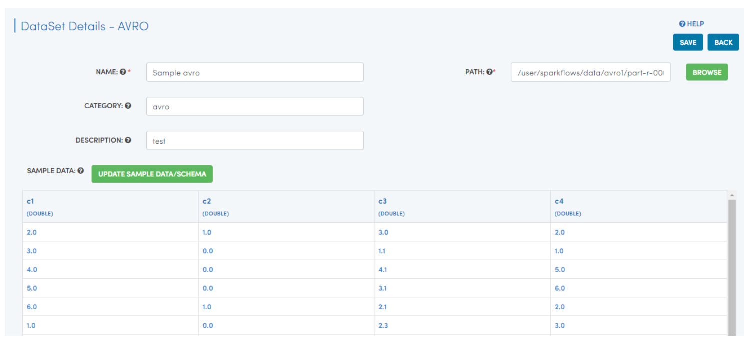



Clicking "OK" will take you to Dataset Details page where you can enter information about your dataset. In the screenshot below, we create a dataset from a sample.avro file.

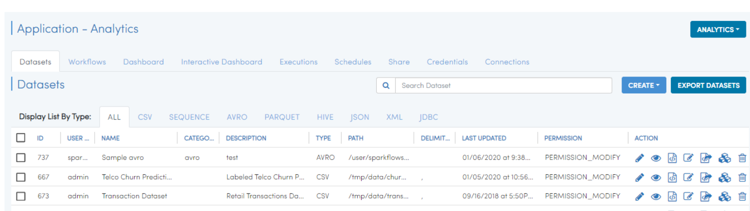


We specified a name, category, description & path of avro file for the dataset we are creating.

Once we have specified the above, we hit the 'Update Sample data/schema' button. This brings up the sample data, infers the schema and displays it. We can change the column names and also the data types. Format column is used for specifying the format for date/time fields.


Clicking the 'Save' button saves the new avro dataset. The avro dataset is now ready for use in any workflow within the specific application.

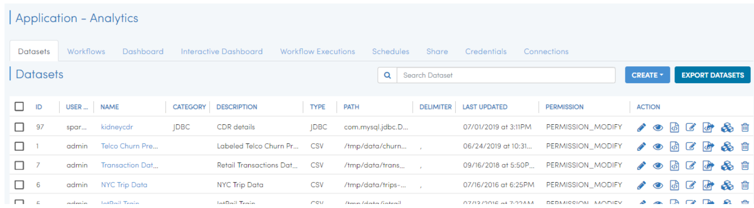



## Creating Dataset for JSON Files

When working with data in Fire Insights, the first step is to create a dataset that you plan to process subsequently. Dataset is a wrapper around your data which makes it easy to handle it in Sparkflows workbench.

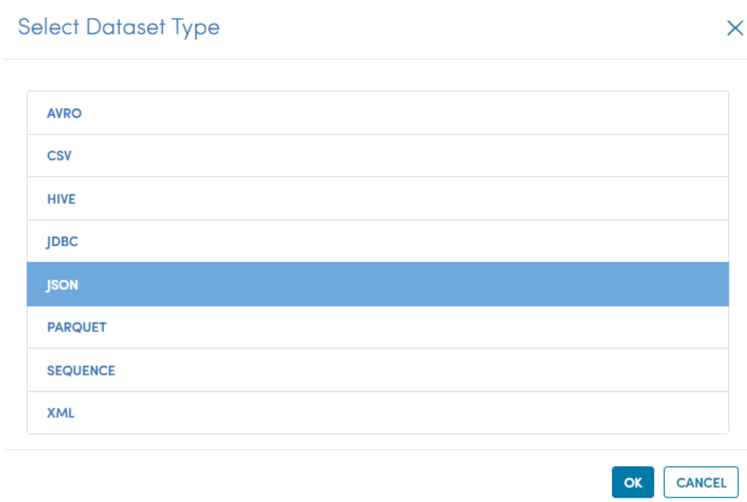
When datasets are created, Fire Insights automatically infers the schema using Spark-Json library.

### Datasets

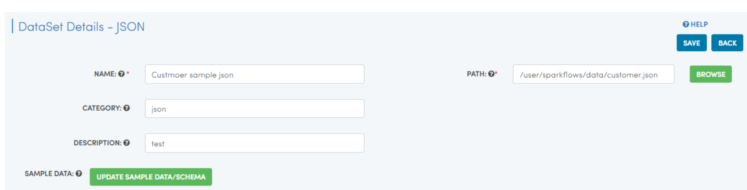



### Dataset Creation

Navigate to the "Datasets" tab in your application where you want to create a new dataset. Click on the "Create" button and choose "Dataset". In the pop-up choose "JSON" and then click "OK".




Clicking "OK" will take you to Dataset Details page where you can enter information about your dataset. In the screenshot below, we create a dataset from a customer.json file.



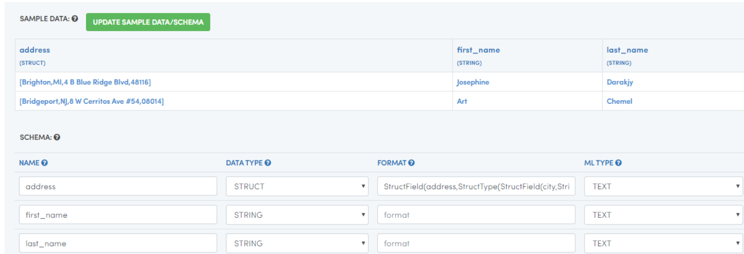

We specified a name, category, description & path of json file for the dataset we are creating.

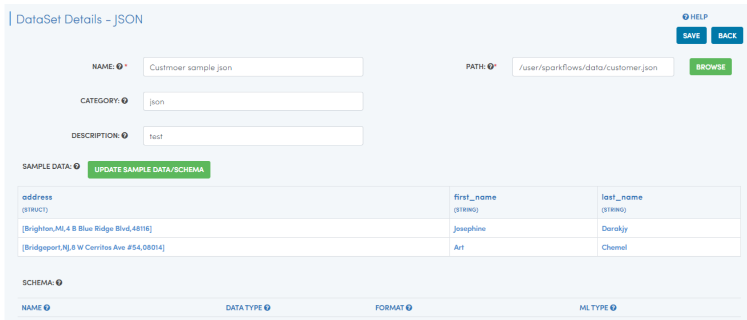
Once we have specified the above, we hit the 'Update Sample data/schema' button. This brings up the sample data, infers the schema and displays it. We can change the column names and also the data types. Format column is used for specifying the format for date/time fields.

Clicking the 'Save' button saves the new json dataset. The json dataset is now ready for use in any workflow within the specific application.

### Creating Dataset for Parquet Files

Fire insights supports reading from several file formats including Parquet files. Parquet files have schema embedded in them. Fire Insights is able to extract schema of Parquet files automatically.




### Datasets

The existing datasets are displayed in the DataSets page of specific application.

### Dataset Creation

Navigate to the "Datasets" tab in your application where you want to create a new dataset. Click on the "Create" button and choose "Dataset". We now create a dataset for people.parquet. It is a parquet file.

In the 'Create DataSet' page fill in the required fields as below.

Specify the name of the dataset you are creating.

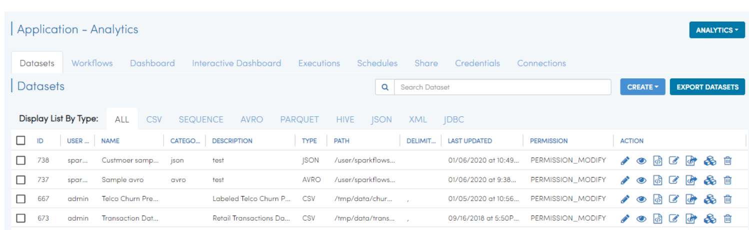
After specifying name and path, click the 'Update Sample data schema' button. This brings up the sample data, extracts the schema and displays it. Below we see that there are 2 fields : age and name. Age is of type integer and name is of type string.

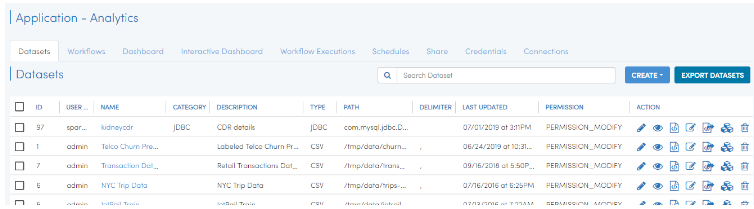
Clicking the 'Save' button creates the new DataSet for us.

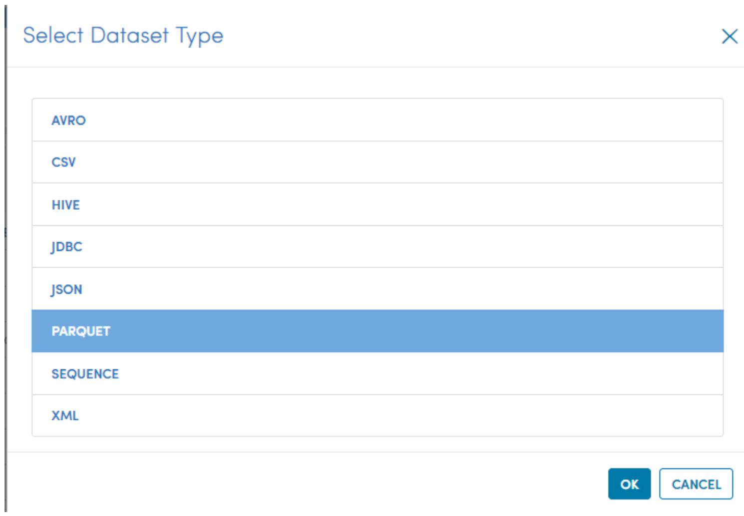
Now you are ready to use the dataset in your workflows.

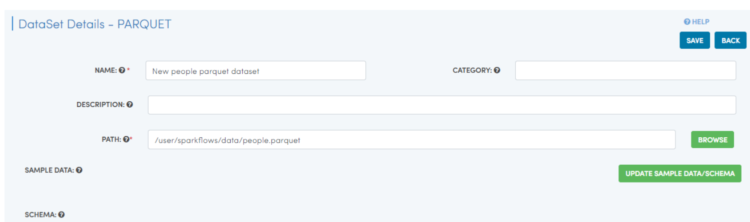
### Creating Dataset from MySQL Table

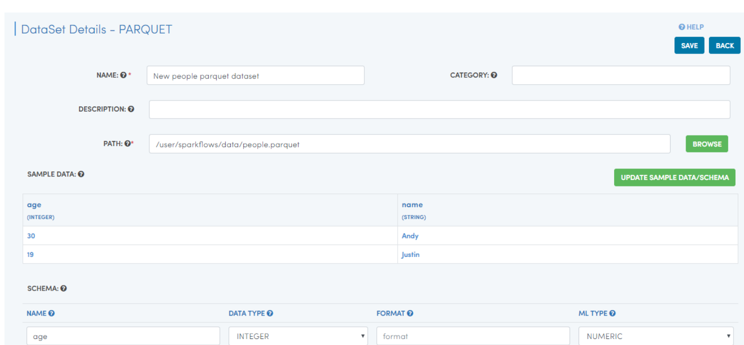
When working with data in Fire Insights, the first step is to create a dataset that you plan to process subsequently. Dataset is a wrapper around your data which makes it easy to handle it in Sparkflows workbench.

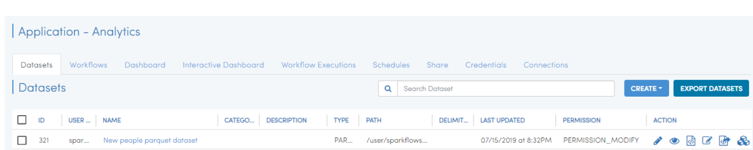



Application - Analytics
Datasets
Workflows
Dashboard
Interactive Dashboard
Workflow Executions
Schedules
Share
Credentials
Connections
Datasets
Search Dataset
CREATE
EXPORT DATASETS


Select Dataset Type
AVRO
CSV
HIVE
JDBC
JSON
PARQUET
SEQUENCE
XML
OK
CANCEL


DataSet Details - PARQUET
HELP
SAVE
BACK
NAME:
New people parquet dataset
CATEGORY:
DESCRIPTION:
PATH:
/user/sparkflows/data/people.parquet
BROWSE
SAMPLE DATA:
UPDATE SAMPLE DATA/SCHEMA
SCHEMA:

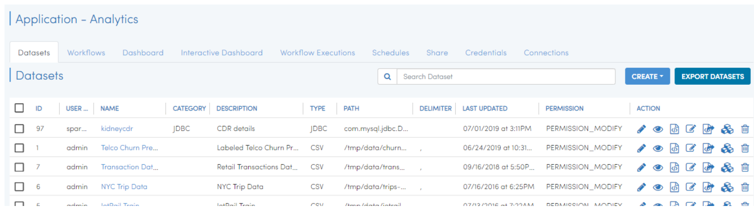

DataSet Details - PARQUET
HELP
SAVE
BACK
NAME:
New people parquet dataset
CATEGORY:
DESCRIPTION:
PATH:
/user/sparkflows/data/people.parquet
BROWSE
SAMPLE DATA:
UPDATE SAMPLE DATA/SCHEMA
age
(INTEGER)
name
(STRING)
30
Andy
19
Justin
SCHEMA:
NAME
DATA TYPE
FORMAT
ML TYPE
age
INTEGER
format
NUMERIC


Application - Analytics
Datasets
Workflows
Dashboard
Interactive Dashboard
Workflow Executions
Schedules
Share
Credentials
Connections
Datasets
Search Dataset
CREATE
EXPORT DATASETS
New people parquet dataset
PERMISSION_MODIFY

When datasets are created, Fire Insights automatically infers the schema of the dataset.

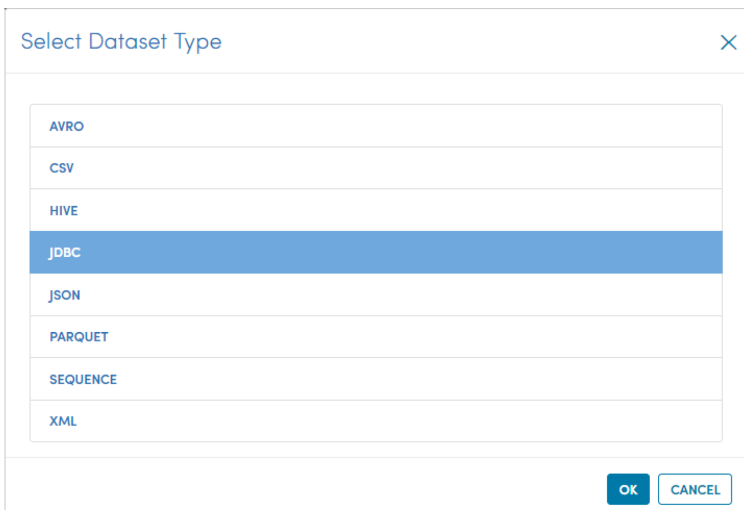
### Datasets

When you open any application, all existing datasets specific to the application are displayed in the Datasets tab.



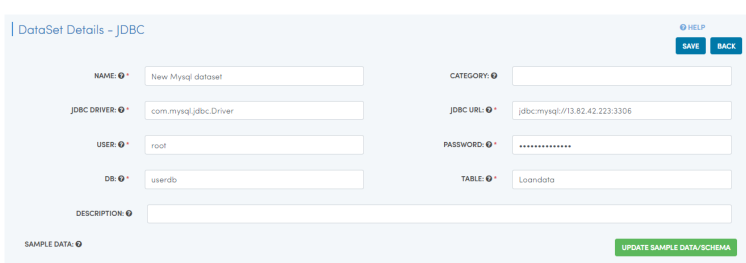

### Dataset Creation

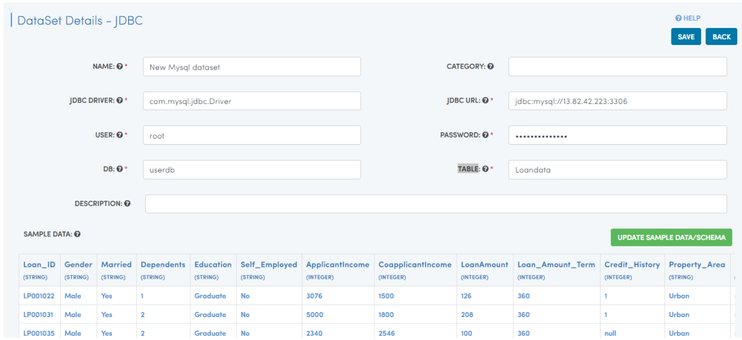
Navigate to the "Datasets" tab in your application where you want to create a new dataset. Click on the "Create" button and choose "Dataset". In the pop-up choose "JDBC" and then click "OK".



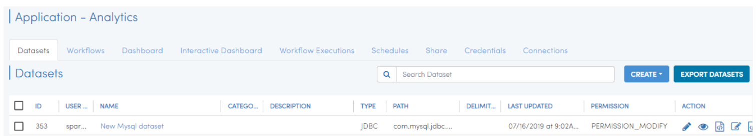

Specify the name of the dataset you are creating and other required parameters such as JDBC DRIVER, JDBC URL, USER, PASSWORD, DB, & TABLE etc.

Once you have filled in required information, hit 'Update Sample data/schema' button. This brings up sample data, infers the schema and displays it. You can change column names and data types as needed. Format column is used for specifying the format of date/time fields.






Clicking the 'Save' button creates the new dataset that can be used in any workflow or Interactive dashboard within the specific application.




### Reading from RDBMS in Workflow

Fire has JDBC Processors for reading from JDBC sources or writing to JDBC sinks.

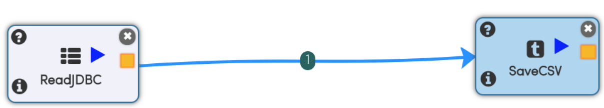
In order to connect to a JDBC source like MySQL/Oracle/DB2 etc. the JDBC driver needs to be installed in Fire Insights.

Use the steps here for installing the corresponding JDBC driver for your RDBMS:

- http://docs.sparkflows.io/en/latest/operating/installing-jdbc-drivers.html

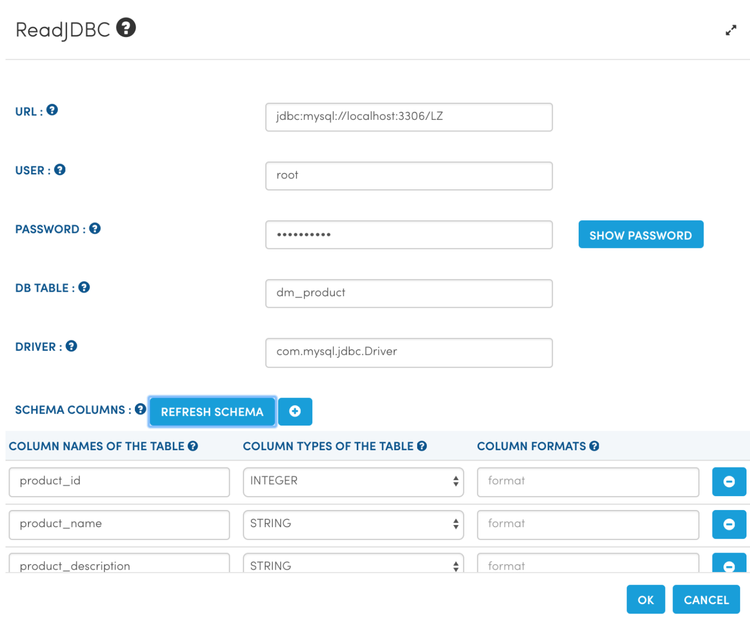
### Workflow for reading from MySQL

Below is a workflow which reads data from MySQL and saves to a CSV file. It reads in the data from the `dm_product` table in MySQL and saves it to a CSV file.



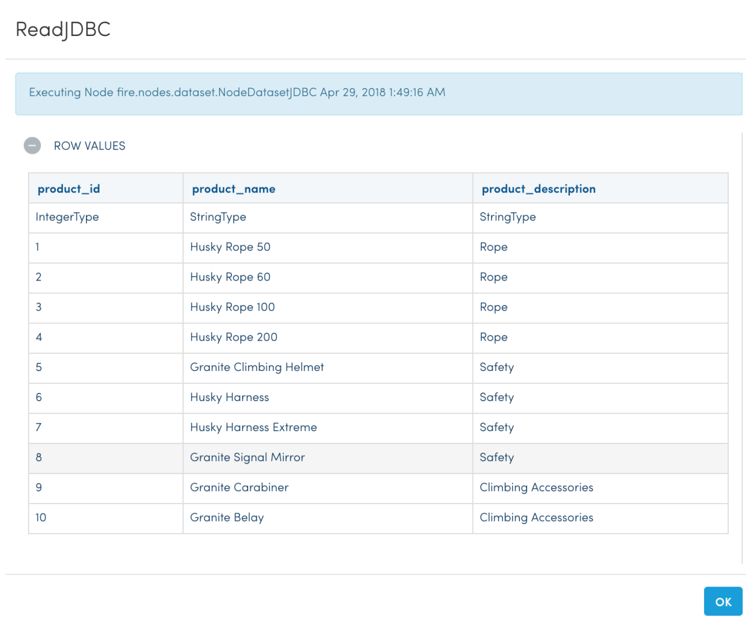

### JDBC Processor Configuration

Below are the configuration details of the JDBC Processor. It uses the provided user for reading from the MySQL database. On clicking on *Refresh Schema*, Fire gets the schema of the table in MySQL and populates the entries.




### Results of reading from MySQL table

The below screenshot displays some of the records read from the MySQL table by Fire.

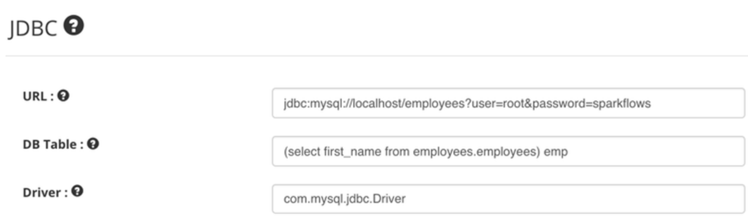



| product_id | product_name | product_description |
|---|---|---|
| IntegerType | StringType | StringType |
| 1 | Husky Rope 50 | Rope |
| 2 | Husky Rope 60 | Rope |
| 3 | Husky Rope 100 | Rope |
| 4 | Husky Rope 200 | Rope |
| 5 | Granite Climbing Helmet | Safety |
| 6 | Husky Harness | Safety |
| 7 | Husky Harness Extreme | Safety |
| 8 | Granite Signal Mirror | Safety |
| 9 | Granite Carabiner | Climbing Accessories |
| 10 | Granite Belay | Climbing Accessories |

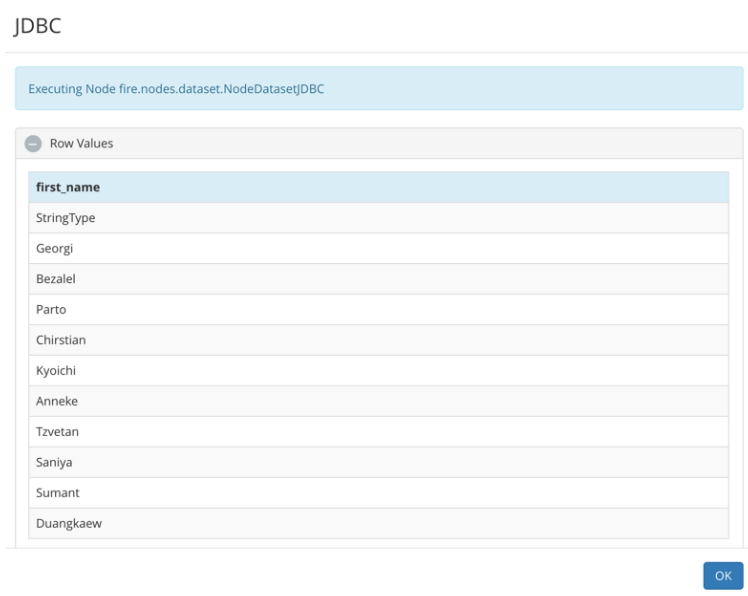
### Specifying a sub-query

In the configuration of the JDBC node, for `db_table` anything that is valid in a FROM clause of a SQL query can be used. For example, instead of a full table we could also use a subquery in parentheses.

More details are available on the Spark Guide : https://spark.apache.org/docs/1.6.0/sql-programming-guide.html#jdbc-to-other-databases




Above we have specified a subquery which selects only the ‘first_name’ from the employees table.




### JDBC Drivers

Below are the JDBC URL’s for some databases:

- MySQL : com.mysql.jdbc.Driver
- PostgreSQL : org.postgresql.Driver
- Oracle : oracle.jdbc.driver.OracleDriver

### Example JDBC URL

Below are some examples of JDBC URL for reading from Relational sources:

- MySQL : jdbc:mysql://localhost:3306/mydb
- PostgreSQL : jdbc:postgresql://localhost:5432/mydb

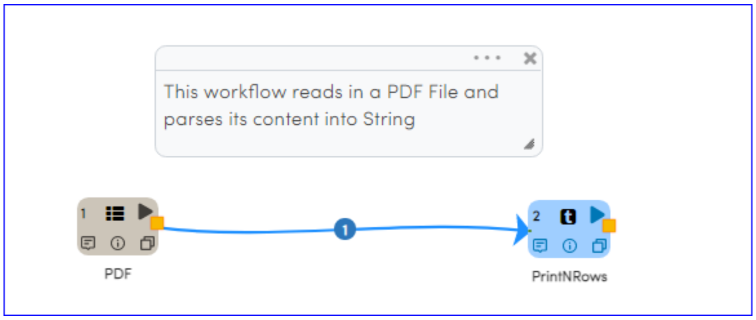
### Read PDF File

This workflow reads in PDF file from the given location. It then parses its content and creates DataFrame then prints the results.

### Workflow

Below is the workflow that shows:

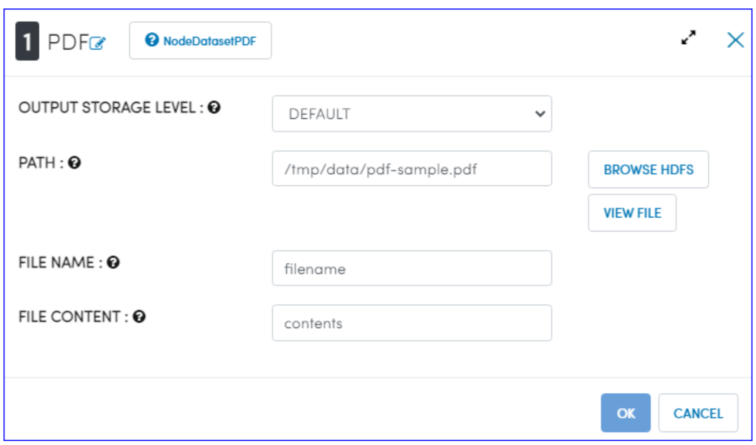
- How to read in PDF file from the given location and create the DataFrame from it
- Prints the result




### Reading And Parsing PDF File

`DatasetPDF` processor uses the passed location to download PDF file, parse its content into string and create the DataFrame.

### Processor Configuration




### Processor Output

### Prints the Results

It prints the result onto the screen.

### Reading and Writing from ElasticSearch

Elastic Search is often used for indexing, searching and analyzing datasets. Fire Insights makes it easy to read data from Elastic Search, clean it and transform it as needed.

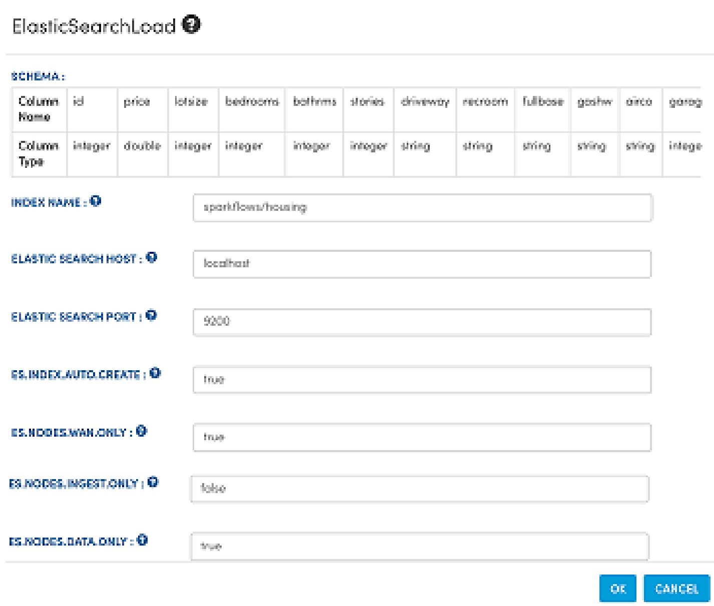

Elasticsearch-hadoop provides native integration between Elasticsearch and Apache Spark. In the example below we will first load data from HDFS into Elastic Search and then read it back into Apache Spark from Elastic Search.

If your data is already in Elastic Search, skip to "Workflow for Reading data from Elastic Search".

### Loading data into Elastic Search

Create a new empty workflow. Drag and drop the source dataset from which you want to load data into Elastic Search. If you don't have a dataset for the source data, create one.

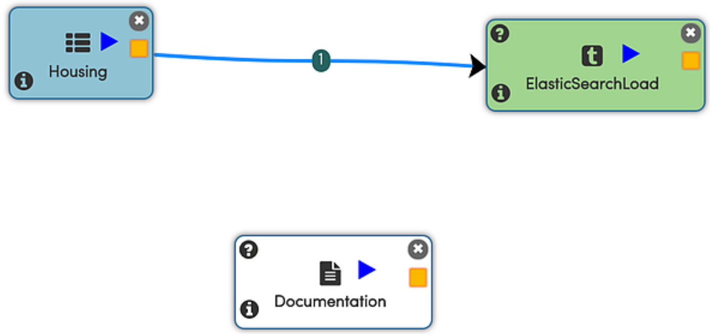
Once the source processor is on the workflow canvas, drag and drop "SaveElasticSearch" processor in the workflow. Configure your Elastic Search processor in the dialog box shown below.




After configuring "SaveElasticSearch" processor, connect your data source processor to Elastic Search processor.

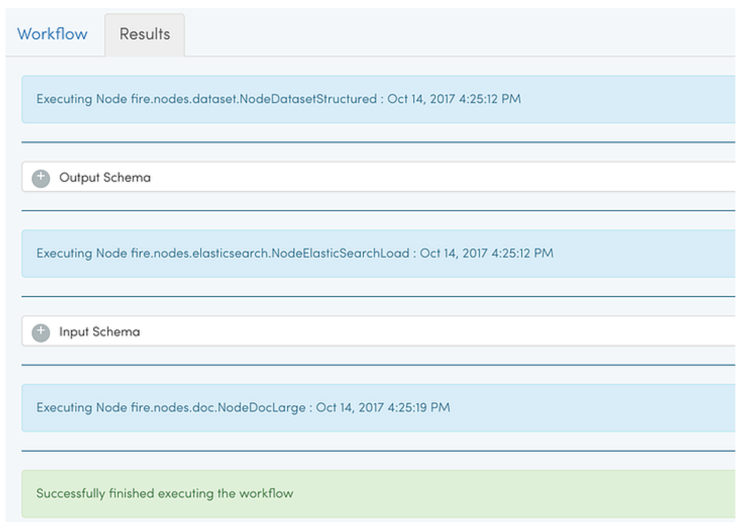
The example workflow below reads a Housing dataset which is in CSV format from HDFS. The 'SaveElasticSearch' takes in the incoming data and loads it into the Elastic Search Index 'sparkflows/housing'.

### Note: Documentation processor is just for documentation purposes.




### Workflow Execution

When the example workflow above is executed, it reads in the dataset from HDFS and saves it into Elastic Search.




### Reading data from Elastic Search

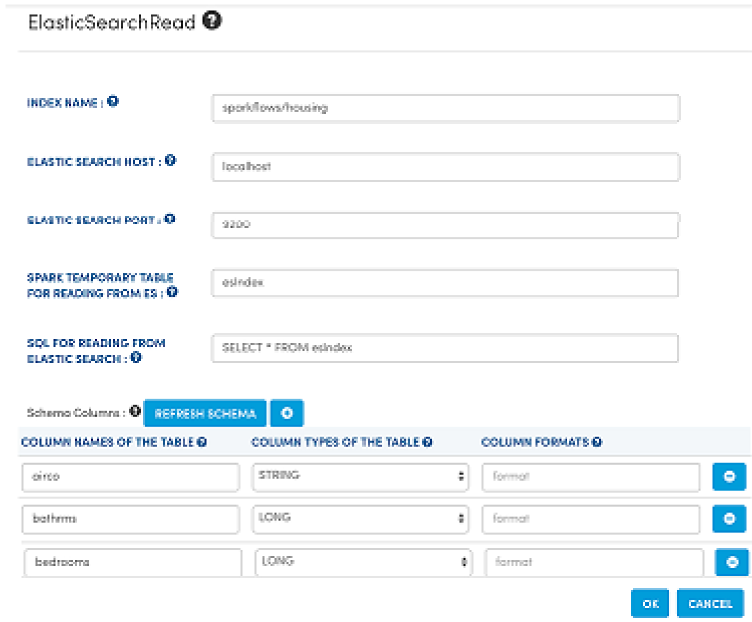
Reading data from Elastic Search is easy. Drag and drop 'ReadElasticSearch' process into your workflow and configure it. The screenshot below shows the dialog box for the Elastic Search Read processor.

In the dialog above, 'Refresh Schema' button infers the schema of the index. Thus it is able to pass down the output schema to the next processor making it easy to build workflows.

The SQL field specifies the SQL to be used for reading from Elastic Search. It allows you to limit the columns of interest, and apply where clauses etc.

The Elastic Search processor understands the SQL and translates it into the appropriate QueryDSL. The connector pushes down the operations directly to the source, where the data is efficiently filtered out so that only the required data is streamed back to Spark. This significantly increases the query performance and minimizes the CPU, memory and I/O operations on both Spark and Elastic Search clusters.

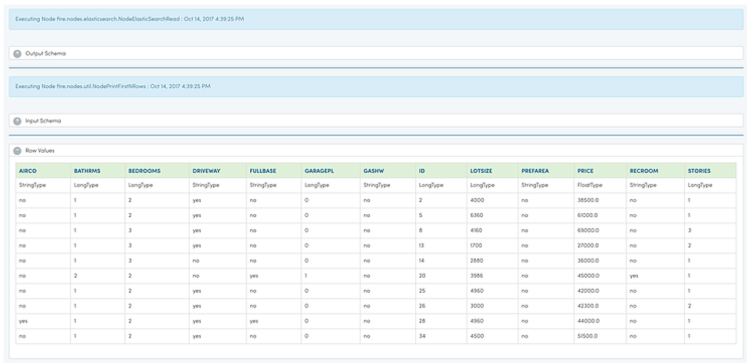
The example workflow below reads the data from the sparkflows/housing index in Elastic Search and prints out the first few lines.


ElasticSearchRead
INDEX NAME :
sparkflows/housing
ELASTIC SEARCH HOST :
localhost
ELASTIC SEARCH PORT :
9200
SPARK TEMPORARY TABLE FOR READING FROM ES :
esIndex
SQL FOR READING FROM ELASTIC SEARCH :
SELECT * FROM esIndex
Schema Columns :
REFRESH SCHEMA
COLUMN NAMES OF THE TABLE
COLUMN TYPES OF THE TABLE
COLUMN FORMATS
STRING
bathrms
LONG
format
bedrooms
LONG
OK
CANCEL


ElasticSearchRead
1
PrintNRows

### Workflow Execution

When the example workflow above is executed, it reads in the index from Elastic Search and displays the first few lines.


Executing Node fire.nodes.elasticsearch.NodeElasticSearchRead : Oct 14, 2017 4:39:25 PM

Output Schema

Executing Node fire.nodes.util.NodePrintFirstNRows : Oct 14, 2017 4:39:25 PM

Input Schema

Row Values

| AIRCO | BATHRMS | BEDROOMS | DRIVEWAY | FULLBASE | GARAGEPL | GASHW | ID | LOTSIZE | PREFAREA | PRICE | RECROOM | STORIES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| StringType | LongType | LongType | StringType | StringType | LongType | StringType | LongType | LongType | StringType | FloatType | StringType | LongType |
| no | 1 | 2 | yes | no | 0 | no | 2 | 4000 | no | 38500.0 | no | 1 |
| no | 1 | 2 | yes | no | 0 | no | 5 | 6360 | no | 61000.0 | no | 1 |
| no | 1 | 3 | yes | no | 0 | no | 8 | 4160 | no | 69000.0 | no | 3 |
| no | 1 | 3 | yes | no | 0 | no | 13 | 1700 | no | 27000.0 | no | 2 |
| no | 1 | 3 | no | no | 0 | no | 14 | 2880 | no | 36000.0 | no | 1 |
| no | 2 | 2 | no | yes | 1 | no | 20 | 3986 | no | 45000.0 | yes | 1 |
| no | 1 | 2 | yes | no | 0 | no | 25 | 4960 | no | 42000.0 | no | 1 |
| no | 1 | 2 | yes | no | 0 | no | 26 | 3000 | no | 42300.0 | no | 2 |
| yes | 1 | 2 | yes | yes | 0 | no | 28 | 4960 | no | 44000.0 | no | 1 |
| no | 1 | 2 | yes | no | 0 | no | 34 | 4500 | no | 51500.0 | no | 1 |

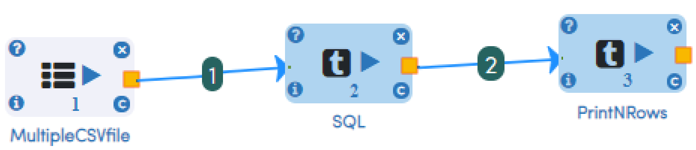
### Processing multiple files

This workflow reads in multiple files available in specific directory. It then filters and calculates number of bedrooms with specific prices and then prints the results.

### Workflow

Below is the workflow. It does the following:

- Reads multiple csv files available in specific directory.
- Filters it to calculate number of bedrooms with specific prices.
- Prints the results.

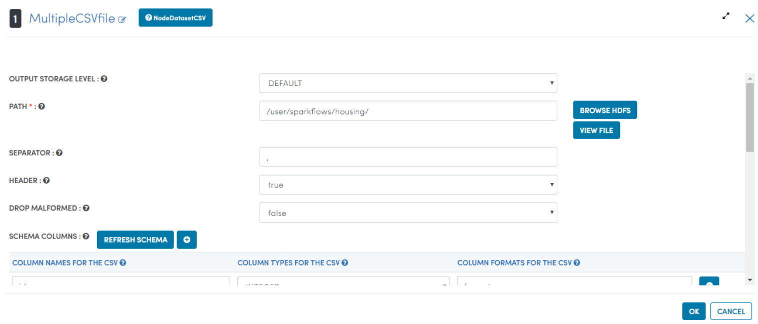


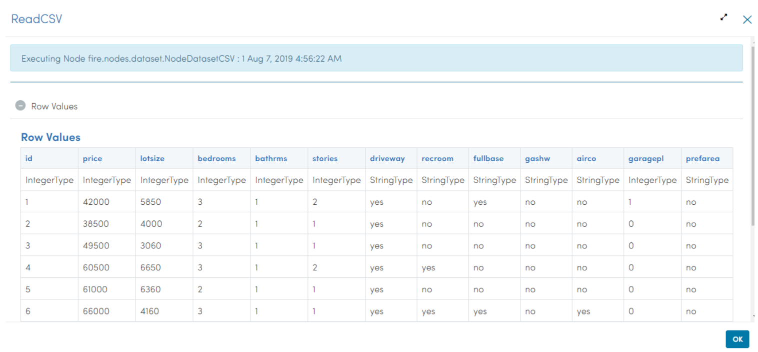

### Reading CSV files

It reads multiple CSV files available in specific directory using ReadCSV processor.

### Processor Configuration

### Processor Output

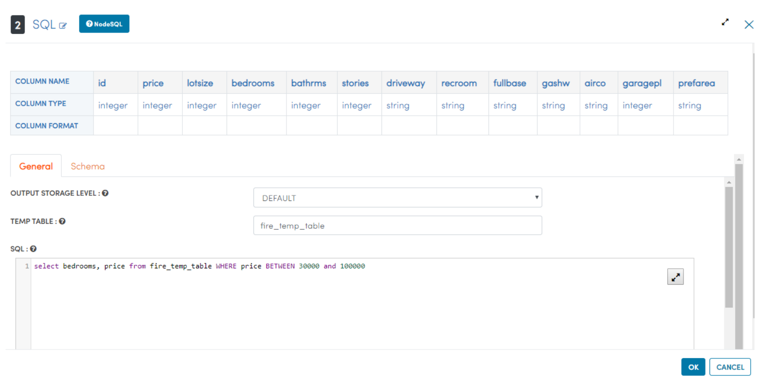



### Filter its data

It then filters to calculate number of bedrooms with specific prices using SQL processor.

### Processor Configuration

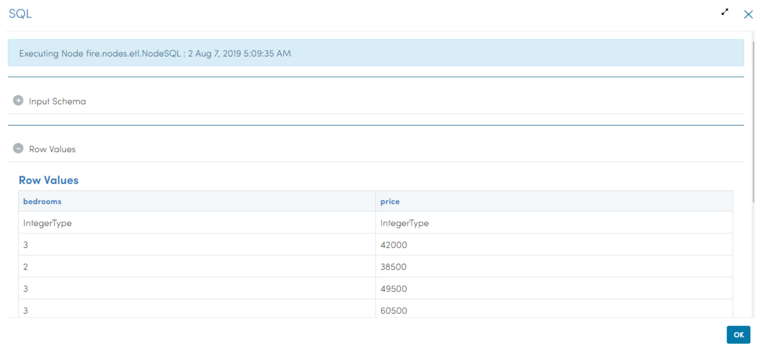


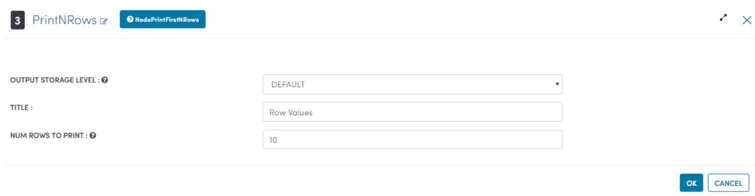

### Processor Output

### Print the results

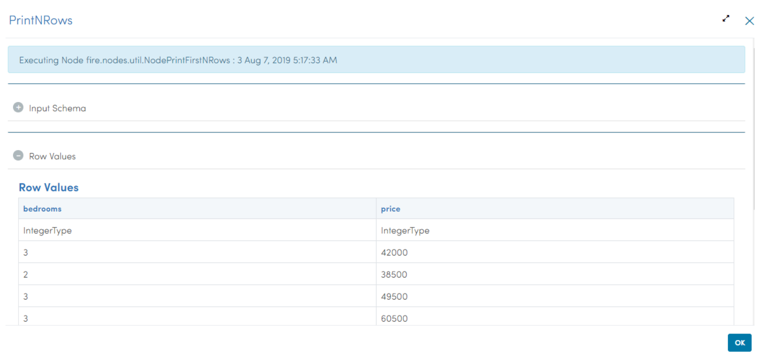
It will print the results with the output required after filter aggregation.

### Processor Configuration




### Processor Output




### Saving Data to HIVE

As par of your data pipeline or workflow, you might want to save data to HIVE after it has been read from a data source, cleaned and transformed. After data is saved in HIVE it can be read from another workflow or accessed through BI tools such as Tableau.

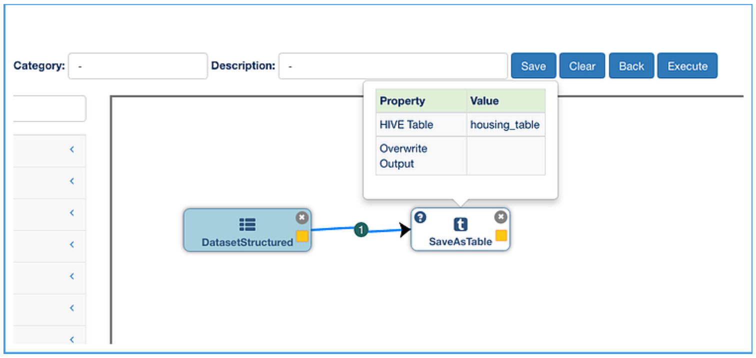
### Cluster vs Standalone Mode

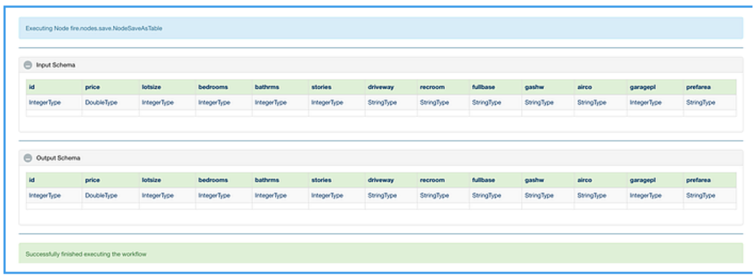
In your workflow, drag and drop a "SaveAsHIVETable" processor. Configure the processor to save your data into HIVE as a table which can be read later.

Note: Fire Insights can run in cluster mode or in the standalone mode. These settings are in Administration/Configuration.When connecting to HIVE, Sparkflows must be running in cluster mode on an edge node of a Hadoop cluster. HIVE settings have to be correctly set under Administration/Configuration-> app.runOnCluster.

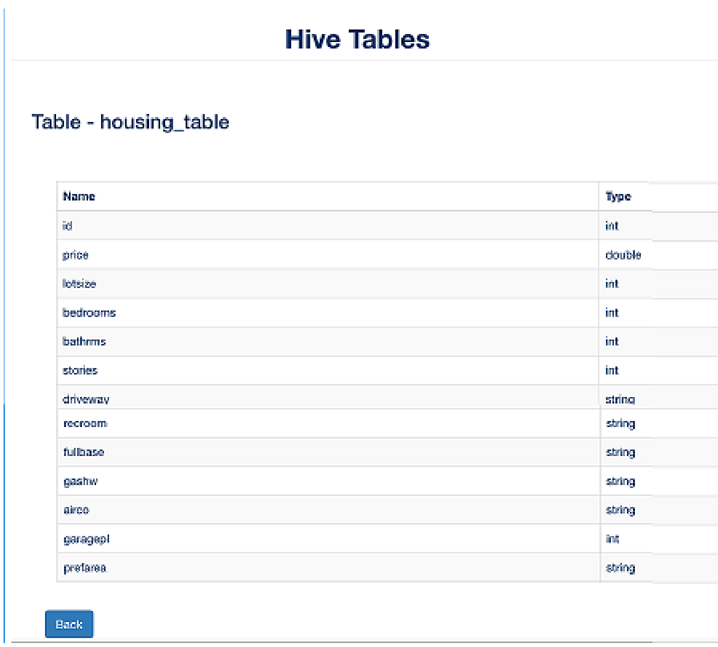
The example workflow below, contains "SaveAsHIVETable" processor. It reads Housing dataset and saves it into the HIVE 'housing_table'.

When the example workflow is executed, data is written into HIVE table 'housing_table'.



The ‘housing_table’ gets created with the schema of the Housing Dataset.

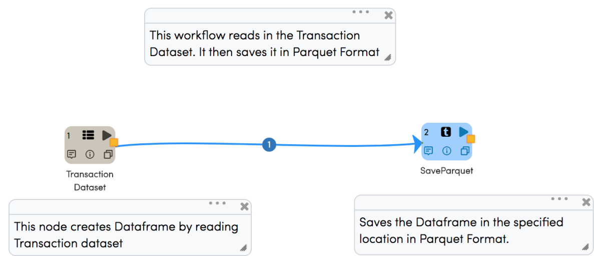



## Writing to Parquet Files

Fire Insights enables you to write your Dataframe to Parquet Files.

### Workflow for writing to Parquet file

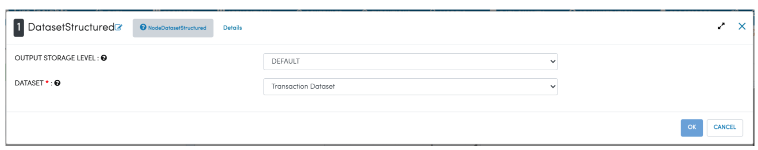
Below is a workflow example which reads in transaction data. It then writes it out to Parquet files.




### DatasetStructured Processor

Node `DatasetStructured` creates a Dataframe of your dataset named `Transaction Dataset` by reading data from HDFS, HIVE etc. which had been defined earlier in Fire by using the Dataset feature.

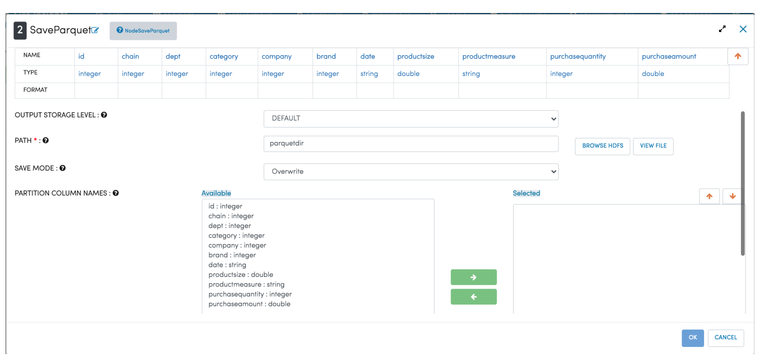
As a user you have to select the Dataset of your interest as shown below.




### SaveParquet Processor

`SaveParquet` processor saves the incoming DataFrame into the specified path in Parquet Format. When running on Hadoop, Parquet files gets saved into HDFS.

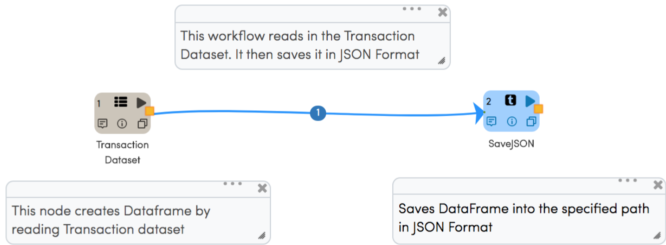
The DataFrame might be written as multiple part files in the specified folder, depending on the size and partition of the DataFrame.




## Writing to JSON Files

Fire Insights enables you to write your DataFrame to JSON Files.

### Workflow for writing to JSON file

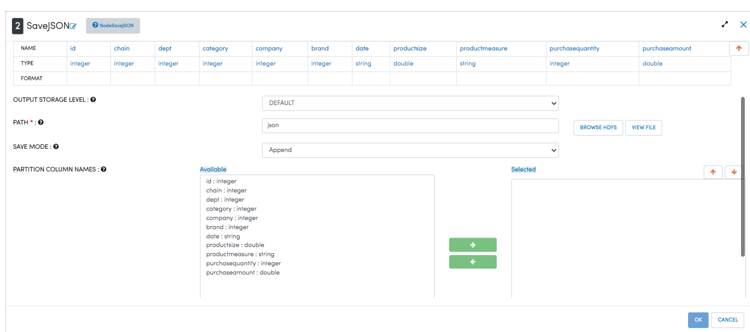



### Reading From Dataset

Node `TransactionDataset` creates DataFrame of your dataset named 'Transaction Dataset' by reading data from HDFS, HIVE etc. which had been defined earlier in Fire by using the Dataset feature. As a user you just have to select the Dataset of your interest and configure the details as shown below.


### SaveJSON Processor Configuration

Node `SaveJSON` saves DataFrame into the specified path in JSON Format. When running on Hadoop, JSON files gets saved into HDFS.




### Reading and Writing from MongoDB

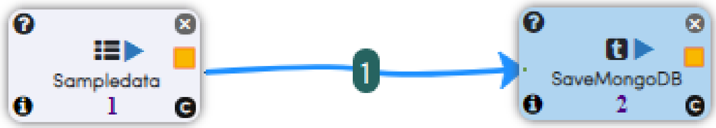
MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need. Here we are loading data from HDFS and Saving it into MongoDB.

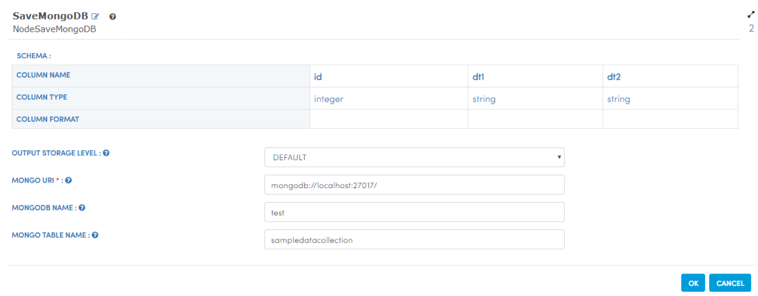
### Workflow for Loading data into MongoDB

The below workflow reads in the Sample Dataset which is in CSV format from HDFS.

It then saves the data into MongoDB.

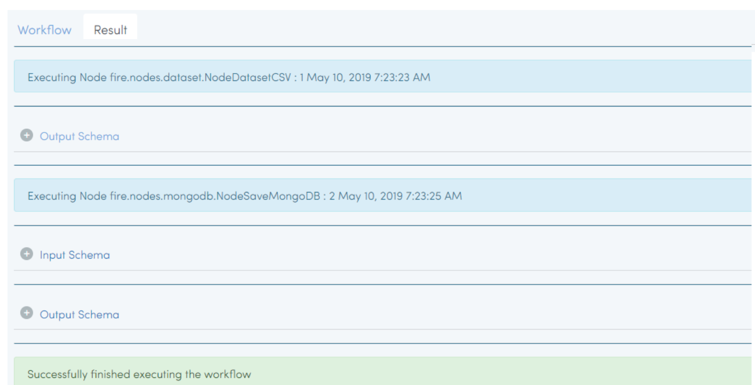
The below diagram shows the dialog box for the SaveMongoDB Processor.




### Workflow Execution

When we execute the Workflow, it reads in the dataset from HDFS and loads it into MongoDB.




### Workflow for Reading data from MongoDB

The below workflow reads Data in MongoDB.It then prints the data.

The below diagram shows the dialog box for the ReadMongoDB Processor.

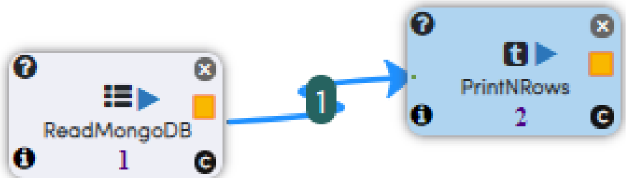
In the above dialog, the 'Refresh Schema' button infers the schema of the collections. Thus it is able to pass down the output schema to the next Processor making it easy for us to build the workflow.

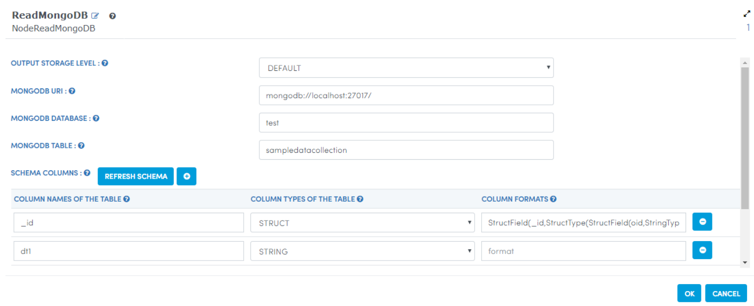
### Workflow Execution

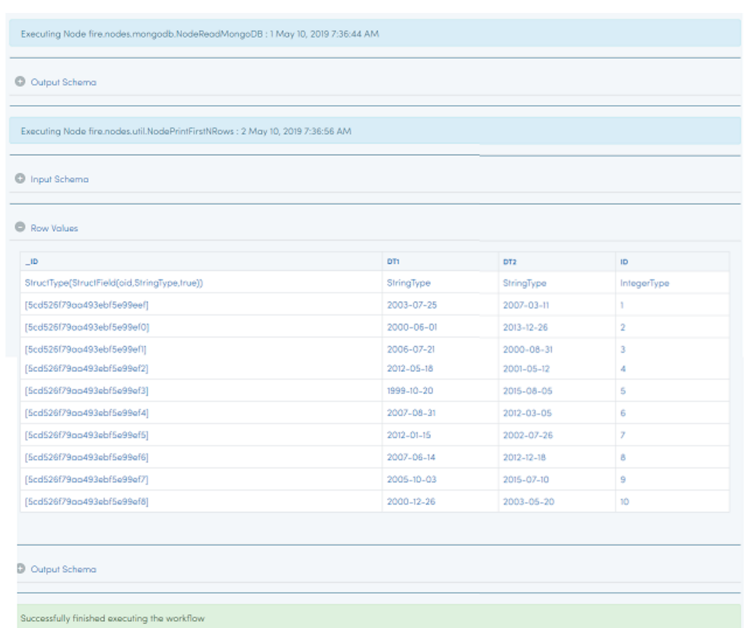
When we execute the Workflow, it reads in the Sample collection from MongoDB and displays the first few lines.

We see that the Sample data records we wrote to MongoDB in the first workflow is read back now.

## 12.1.2 Data Exploration


ReadMongoDB
1
1
PrintNRows
2


ReadMongoDB
NodeReadMongoDB
OUTPUT STORAGE LEVEL :
DEFAULT
MONGODB URI :
mongodb://localhost:27017/
MONGODB DATABASE :
test
MONGODB TABLE :
sampledatacollection
SCHEMA COLUMNS :
REFRESH SCHEMA
COLUMN NAMES OF THE TABLE
COLUMN TYPES OF THE TABLE
COLUMN FORMATS
_id
STRUCT
StructField(_id,StructType(StructField(oid,StringTyp
dt1
STRING
format
OK
CANCEL

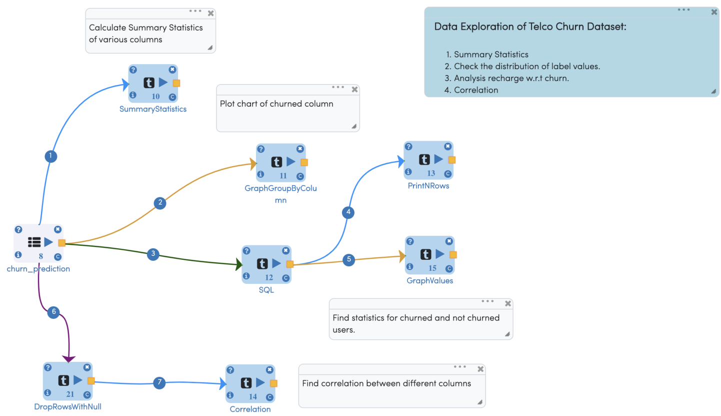

Executing Node fire.nodes.mongodb.NodeReadMongoDB : 1 May 10, 2019 7:36:44 AM
Output Schema
Executing Node fire.nodes.util.NodePrintFirstNRows : 2 May 10, 2019 7:36:56 AM
Input Schema
Row Values
_ID
DT1
DT2
ID
StructType(StructField(oid,StringType,true))
StringType
StringType
IntegerType
[5cd526f79aa493ebf5e99eef] 2003-07-25 2007-03-11 1
[5cd526f79aa493ebf5e99ef0] 2000-06-01 2013-12-26 2
[5cd526f79aa493ebf5e99ef1] 2006-07-21 2000-08-31 3
[5cd526f79aa493ebf5e99ef2] 2012-05-18 2001-05-12 4
[5cd526f79aa493ebf5e99ef3] 1999-10-20 2015-08-05 5
[5cd526f79aa493ebf5e99ef4] 2007-08-31 2012-03-05 6
[5cd526f79aa493ebf5e99ef5] 2012-01-15 2002-07-26 7
[5cd526f79aa493ebf5e99ef6] 2007-06-14 2012-12-18 8
[5cd526f79aa493ebf5e99ef7] 2005-10-03 2015-07-10 9
[5cd526f79aa493ebf5e99ef8] 2000-12-26 2003-05-20 10
Output Schema
Successfully finished executing the workflow

### Telco Churn Data Exploration

Data Profiling is extremely helpful in understanding the data. Fire Insights provides a number of processors for users to profile their data.

Workflow for Data Profiling

Below is a workflow which profiles the Telco Churn Dataset.




### Input Telco Churn Data

The input dataset looks like below:

| state | account_length | area_code | phone_number | intl_plan | voice_mail_plan | number_vmail_messages | today_day_minutes | today_day_calls | today_day_change | total_eve_minutes |
|---|---|---|---|---|---|---|---|---|---|---|
| StringType | DoubleType | DoubleType | StringType | StringType | StringType | DoubleType | DoubleType | DoubleType | DoubleType | DoubleType |
| KS | 128.0 | 415.0 | 382-4657 | no | yes | 25.0 | 265.1 | 110.0 | 45.07 | 197.4 |
| OH | 107.0 | 415.0 | 371-7191 | no | yes | 26.0 | 161.6 | 123.0 | 27.47 | 195.5 |
| NJ | 137.0 | 415.0 | 358-1921 | no | no | 0.0 | 243.4 | 114.0 | 41.38 | 121.2 |
| OH | 84.0 | 408.0 | 375-9999 | yes | no | 0.0 | 299.4 | 71.0 | 50.9 | 61.9 |
| OK | 75.0 | 415.0 | 330-6626 | yes | no | 0.0 | 166.7 | 113.0 | 28.34 | 148.3 |
| AL | 118.0 | 510.0 | 391-8027 | yes | no | 0.0 | 223.4 | 98.0 | 37.98 | 220.6 |
| MA | 121.0 | 510.0 | 355-9993 | no | yes | 24.0 | 218.2 | 88.0 | 37.09 | 348.5 |
| MO | 147.0 | 415.0 | 329-9001 | yes | no | 0.0 | 157.0 | 79.0 | 26.69 | 103.1 |
| LA | 117.0 | 408.0 | 335-4719 | no | no | 0.0 | 184.5 | 97.0 | 31.37 | 351.6 |
| WV | 141.0 | 415.0 | 330-8173 | yes | yes | 37.0 | 258.6 | 84.0 | 43.96 | 222.0 |
| IN | 65.0 | 415.0 | 329-6603 | no | no | 0.0 | 129.1 | 137.0 | 21.95 | 228.5 |

### Workflow Execution Result

When the above workflow is executed, it produces the below results. The good thing about Fire Insights is that the Data Profiling runs in a distributed fashion. So, whatever the number of records in the input dataset, it scales seamlessly.

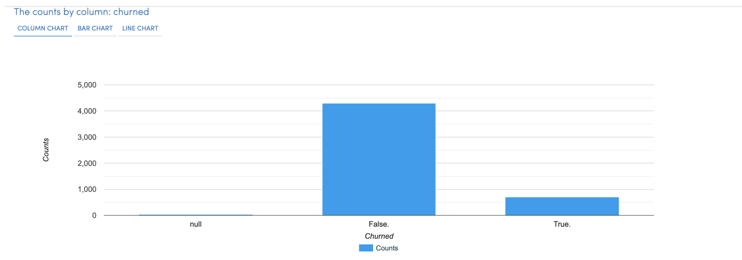
### Summary Statistics

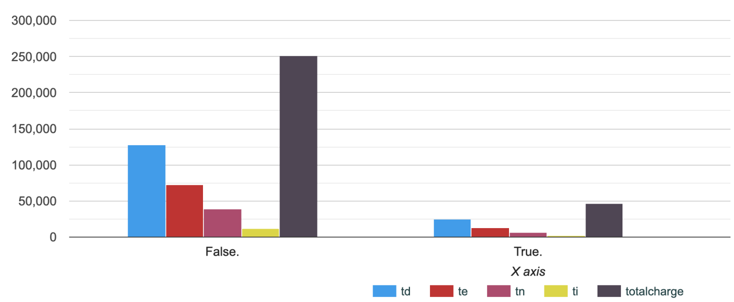
### Counts by Churned Column

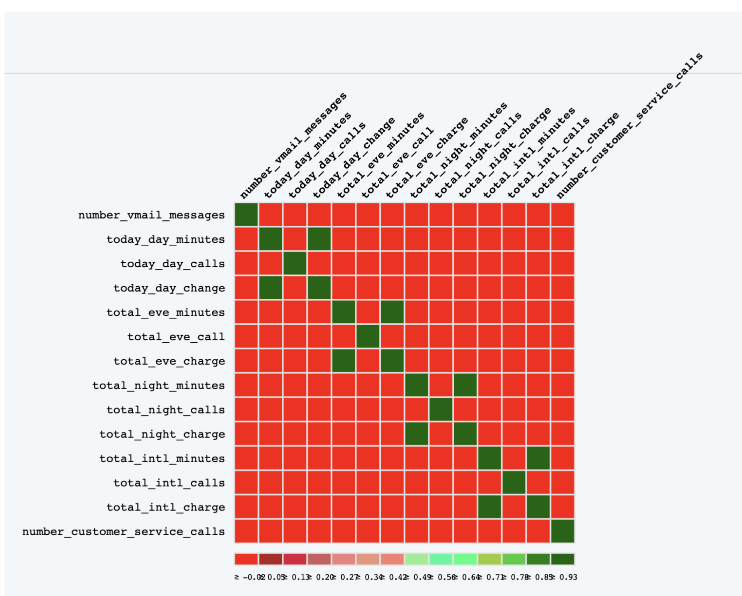
### Graph of counts of various attributes for Churned and Not Churned customers

Summary

| summary | number_vmail_messages | today_day_minutes | today_day_calls | today_day_change | total_eve_minutes | total_eve_call | total_eve_charge | total_night_minutes | total_night_call |
|---|---|---|---|---|---|---|---|---|---|
| count | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 |
| mean | 7.755 | 180.289 | 100.029 | 30.65 | 200.637 | 100.191 | 17.054 | 200.392 | 99.919 |
| min | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 25_percentile | 0.0 | 143.7 | 87.0 | 24.43 | 166.3 | 87.0 | 14.14 | 166.9 | 87.0 |
| 50_percentile | 0.0 | 180.1 | 100.0 | 30.62 | 201.0 | 100.0 | 17.09 | 200.4 | 100.0 |
| 75_percentile | 17.0 | 216.2 | 113.0 | 36.75 | 234.1 | 114.0 | 19.9 | 234.7 | 113.0 |
| max | 52.0 | 351.5 | 165.0 | 59.76 | 363.7 | 170.0 | 30.91 | 395.0 | 175.0 |
| stdev | 13.546 | 53.895 | 19.831 | 9.162 | 50.551 | 19.826 | 4.297 | 50.528 | 19.959 |
| variance | 183.505 | 2904.639 | 393.276 | 83.944 | 2555.435 | 393.09 | 18.463 | 2553.057 | 398.349 |



### Correlation Matrix

## 12.1.3 Machine Learning

### Telco Churn Prediction

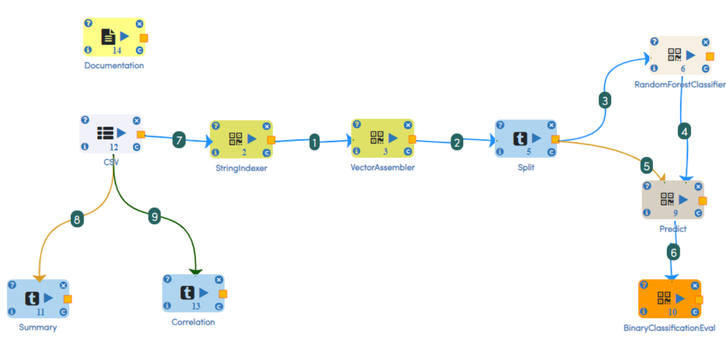
Fire Insights enable us to create a Random Forest Model to predict churn and evaluate the results.

The dataset is artificial Churn Data based on claims, similar to real world. It is taken from the following location.

- https://www.sgi.com/tech/mlc/db/
- https://www.sgi.com/tech/mlc/db/churn.all
- https://www.sgi.com/tech/mlc/db/churn.name

```
KS, 128, 415, 382-4657, no, yes, 25, 265.1, 110, 45.07, 197.4, 99, 16.78, 244.7, 91, 11.01, 10, 3, 2.7,
OH, 107, 415, 371-7191, no, yes, 26, 161.6, 123, 27.47, 195.5, 103, 16.62, 254.4, 103, 11.45, 13.7, 3,
NJ, 137, 415, 358-1921, no, no, 0, 243.4, 114, 41.38, 121.2, 110, 10.3, 162.6, 104, 7.32, 12.2, 5, 3.29
OH, 84, 408, 375-9999, yes, no, 0, 299.4, 71, 50.9, 61.9, 88, 5.26, 196.9, 89, 8.86, 6.6, 7, 1.78, 2, F
OK, 75, 415, 330-6626, yes, no, 0, 166.7, 113, 28.34, 148.3, 122, 12.61, 186.9, 121, 8.41, 10.1, 3, 2.7
AL, 118, 510, 391-8027, yes, no, 0, 223.4, 98, 37.98, 220.6, 101, 18.75, 203.9, 118, 9.18, 6.3, 6, 1.7,
MA, 121, 510, 355-9993, no, yes, 24, 218.2, 88, 37.09, 348.5, 108, 29.62, 212.6, 118, 9.57, 7.5, 7, 2.0
MO, 147, 415, 329-9001, yes, no, 0, 157, 79, 26.69, 103.1, 94, 8.76, 211.8, 96, 9.53, 7.1, 6, 1.92, 0,
LA, 117, 408, 335-4719, no, no, 0, 184.5, 97, 31.37, 351.6, 80, 29.89, 215.8, 90, 9.71, 8.7, 4, 2.35, 1
WV, 141, 415, 330-8173, yes, yes, 37, 258.6, 84, 43.96, 222, 111, 18.87, 326.4, 97, 14.69, 11.2, 5, 3.0
IN, 65, 415, 329-6603, no, no, 0, 129.1, 137, 21.95, 228.5, 83, 19.42, 208.8, 111, 9.4, 12.7, 6, 3.43,
RI, 74, 415, 344-9403, no, no, 0, 187.7, 127, 31.91, 163.4, 148, 13.89, 196, 94, 8.82, 9.1, 5, 2.46, 0,
IA, 168, 408, 363-1107, no, no, 0, 128.8, 96, 21.9, 104.9, 71, 8.92, 141.1, 128, 6.35, 11.2, 2, 3.02, 1
MT, 95, 510, 394-8006, no, no, 0, 156.6, 88, 26.62, 247.6, 75, 21.05, 192.3, 115, 8.65, 12.3, 5, 3.32,
IA, 62, 415, 366-9238, no, no, 0, 120.7, 70, 20.52, 307.2, 76, 26.11, 203, 99, 9.14, 13.1, 6, 3.54, 4,
NY, 161, 415, 351-7269, no, no, 0, 332.9, 67, 56.59, 317.8, 97, 27.01, 160.6, 128, 7.23, 5.4, 9, 1.46,
ID, 85, 408, 350-8884, no, yes, 27, 196.4, 139, 33.39, 280.9, 90, 23.88, 89.3, 75, 4.02, 13.8, 4, 3.73,
VT, 93, 510, 386-2923, no, no, 0, 190.7, 114, 32.42, 218.2, 111, 18.55, 129.6, 121, 5.83, 8.1, 3, 2.19,
VA, 76, 510, 356-2992, no, yes, 33, 189.7, 66, 32.25, 212.8, 65, 18.09, 165.7, 108, 7.46, 10, 5, 2.7, 1
TX, 73, 415, 373-2782, no, no, 0, 224.4, 90, 38.15, 159.5, 88, 13.56, 192.8, 74, 8.68, 13, 2, 3.51, 1,
FL, 147, 415, 396-5800, no, no, 0, 155.1, 117, 26.37, 239.7, 93, 20.37, 208.8, 133, 9.4, 10.6, 4, 2.86,
CO, 77, 408, 393-7984, no, no, 0, 62.4, 89, 10.61, 169.9, 121, 14.44, 209.6, 64, 9.43, 5.7, 6, 1.54, 5,
AZ, 130, 415, 358-1958, no, no, 0, 183, 112, 31.11, 72.9, 99, 6.2, 181.8, 78, 8.18, 9.5, 19, 2.57, 0, F
SC, 111, 415, 350-2565, no, no, 0, 110.4, 103, 18.77, 137.3, 102, 11.67, 189.6, 105, 8.53, 7.7, 6, 2.08
VA, 132, 510, 343-4696, no, no, 0, 81.1, 86, 13.79, 245.2, 72, 20.84, 237, 115, 10.67, 10.3, 2, 2.78, 0
NE, 174, 415, 331-3698, no, no, 0, 124.3, 76, 21.13, 277.1, 112, 23.55, 250.7, 115, 11.28, 15.5, 5, 4.1
WY, 57, 408, 357-3817, no, yes, 39, 213, 115, 36.21, 191.1, 112, 16.24, 182.7, 115, 8.22, 9.5, 3, 2.57,
```

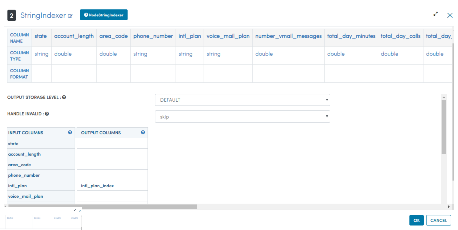
Below is the workflow you can use for creating the model for Churn Prediction.

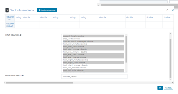



The workflow performs the following steps:

- Reads in the dataset from a tab separated file
- Applies StringIndexer on the field "intl_plan"
- Applies VectorAssembler on the fields we want to model on
- Splits the dataset into (.8, .2)
- Performs Random Forest Classification

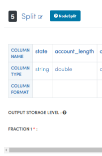
- Performs prediction using the model generated on the remaining 20% dataset
- Finally evaluates the prediction results



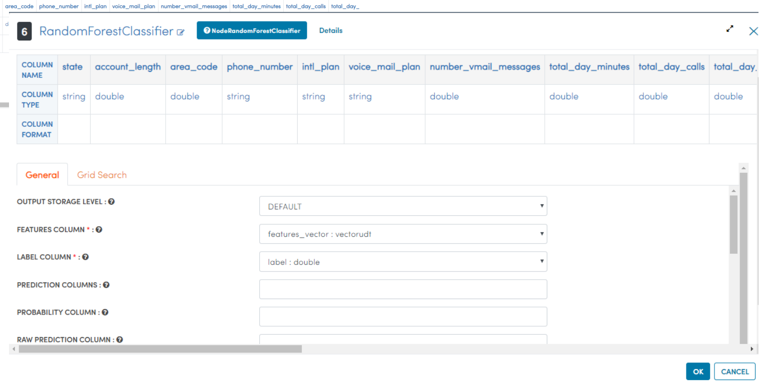


In the VectorAssembler, select the fields you want to include in the model. Only the numeric fields are displayed as VectorAssembler supports only the numeric fields.

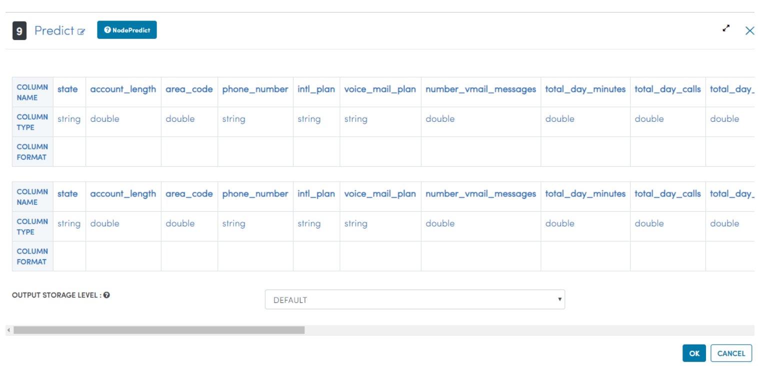
You can split the dataset into training and test datasets. We split it into (.8, .2)



You can use a RandomForestClassifier for predicting churn. We use 20 trees.




You can predict using the model on the test dataset.

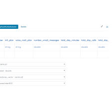



You can evaluate the quality of our results.

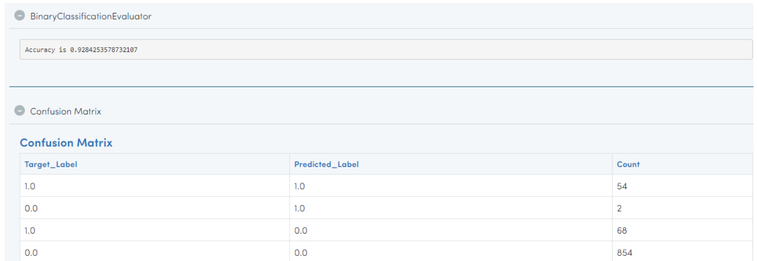
Next, You can execute the workflow.

From the evaluator You get the following results:

## Bike Rental Prediction



This workflow reads in a dataset.It then Predicts the number of bikes to be rented in any given hour.



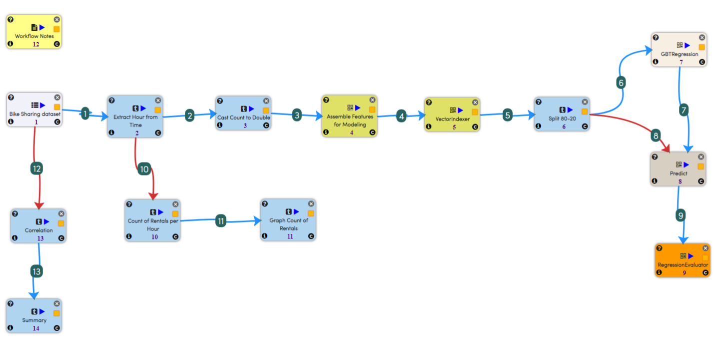

Confusion Matrix

| Target_Label | Predicted_Label | Count |
|---|---|---|
| 1.0 | 1.0 | 54 |
| 0.0 | 1.0 | 2 |
| 1.0 | 0.0 | 68 |
| 0.0 | 0.0 | 854 |

## Workflow

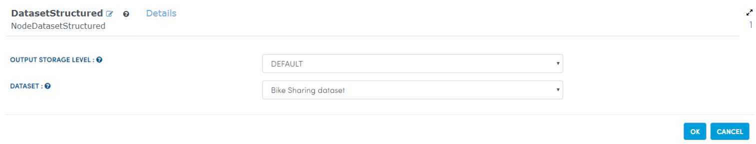
Below is the workflow. It does the following:

- Reads data from a sample dataset.
- Extracts hour from time using datatype timestamp.
- Calculates Count to datatype double.
- Assembles features for modelling.
- Calculates vectorindexer.
- Splits it.
- GBTRegression.
- Prediction.
- RegressionEvaluator.
- Correlation with columns.
- Summary analysis.
- Calculate count for rental per hour.
- Analyse using Graph.



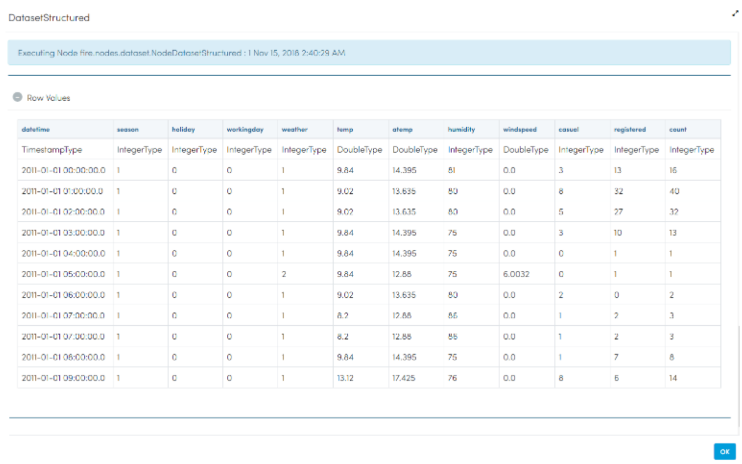

## Reading from Dataset

It reads sample Dataset file.




## Processor Configuration

## Processor Output

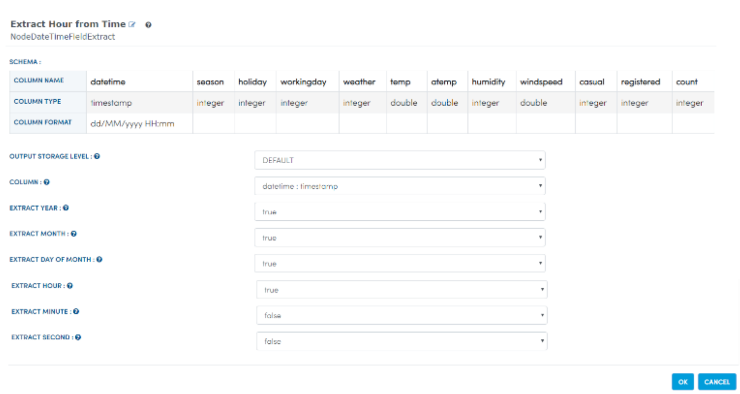



| datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | casual | registered | count |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TimestampType | IntegerType | IntegerType | IntegerType | IntegerType | DoubleType | DoubleType | IntegerType | DoubleType | IntegerType | IntegerType | IntegerType |
| 2011-01-01 00:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 81 | 0.0 | 3 | 13 | 16 |
| 2011-01-01 01:00:00.0 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 8 | 32 | 40 |
| 2011-01-01 02:00:00.0 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 5 | 27 | 32 |
| 2011-01-01 03:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 3 | 10 | 13 |
| 2011-01-01 04:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 0 | 1 | 1 |
| 2011-01-01 05:00:00.0 | 1 | 0 | 0 | 2 | 9.84 | 12.88 | 75 | 6.0032 | 0 | 1 | 1 |
| 2011-01-01 06:00:00.0 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 2 | 0 | 2 |
| 2011-01-01 07:00:00.0 | 1 | 0 | 0 | 1 | 8.2 | 12.88 | 86 | 0.0 | 1 | 2 | 3 |
| 2011-01-01 07:00:00.0 | 1 | 0 | 0 | 1 | 8.2 | 12.88 | 86 | 0.0 | 1 | 2 | 3 |
| 2011-01-01 08:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 1 | 7 | 8 |
| 2011-01-01 09:00:00.0 | 1 | 0 | 0 | 1 | 13.12 | 17.425 | 76 | 0.0 | 8 | 6 | 14 |

## Extract hour from time using datatype timestamp

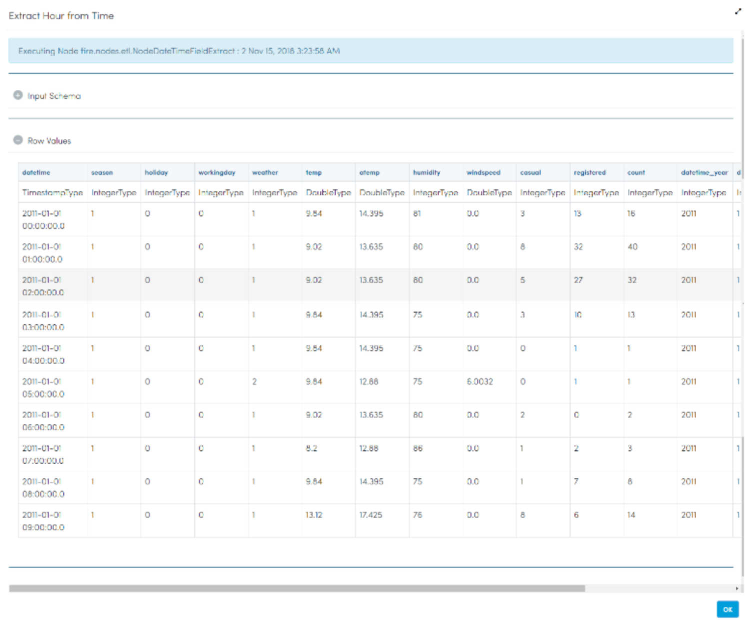
It Extracts hour from time using datatype timestamp using DateTimeFieldExtract Node.

## Processor Configuration




| COLUMN NAME | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | casual | registered | count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COLUMN TYPE | timestamp | integer | integer | integer | integer | double | double | integer | double | integer | integer | integer |
| COLUMN FORMAT | dd/MM/yyyy HH:mm | | | | | | | | | | | |

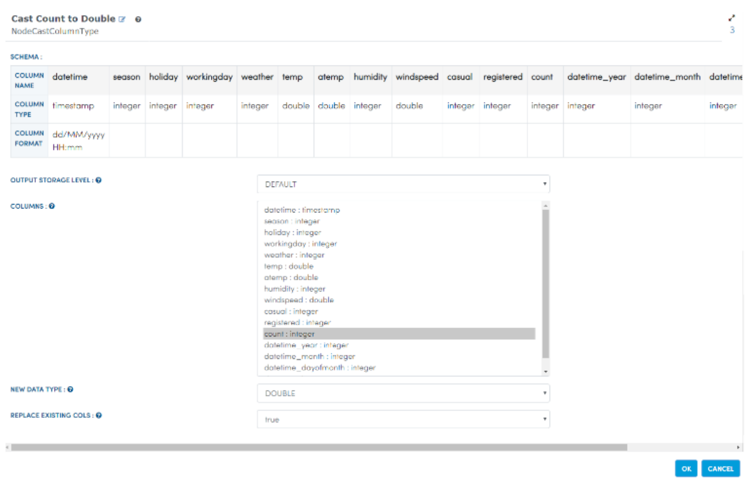
## Processor Output




### Calculate Count to datatype double

It Calculates cast the Count field to datatype double using CastColumnType Node.

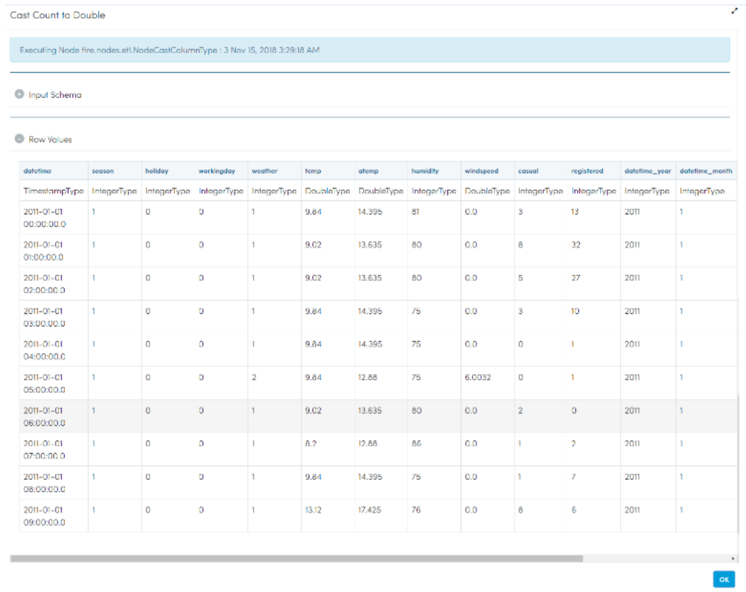
### Processor Configuration



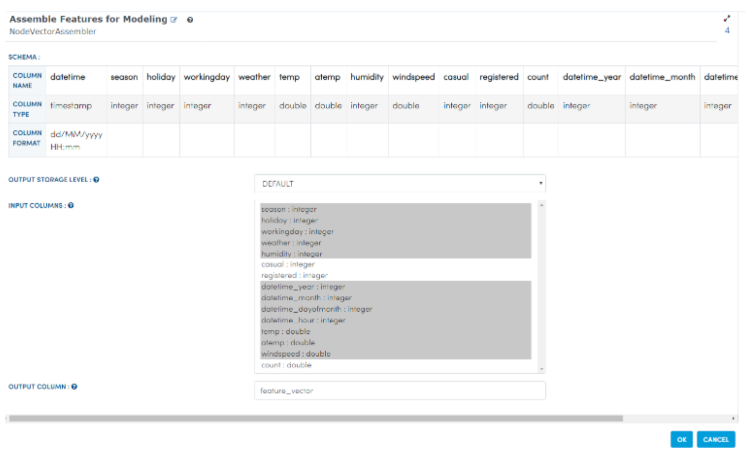

### Processor Output

### Assemble features for modelling

It Assembles features columns into a feature vector using VectorAssembler Node.




### Processor Configuration




### Processor Output

### Calculate vectorindexer

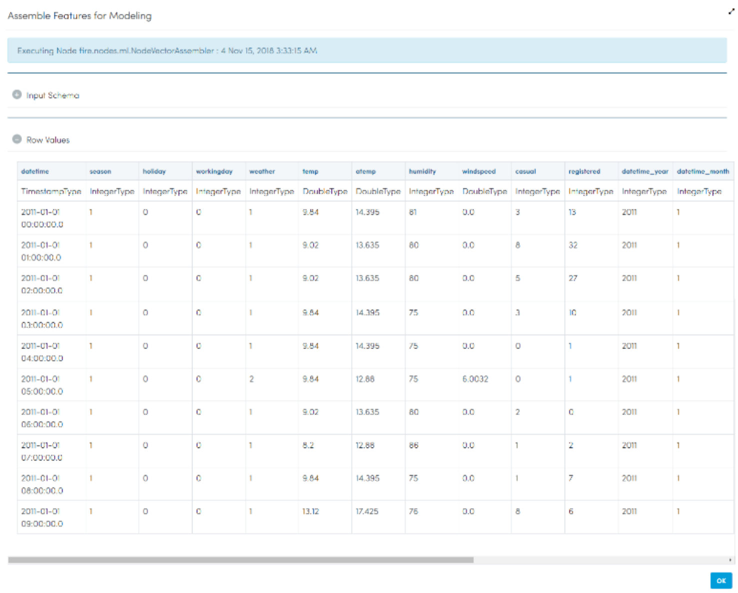
It identifies categorical features and index them using vectorindexer Node.

### Processor Configuration

### Processor Output

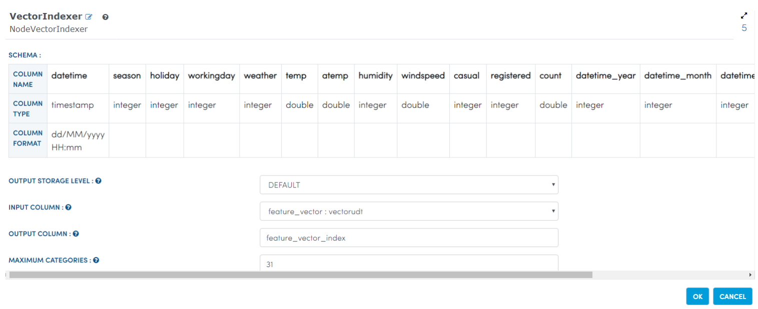
### Split it

It will split our dataset into seperate training and test sets using split Node.


Assemble Features for Modeling
Executing Node fire.nodes.ml.NodeVectorAssembler : 4 Nov 15, 2018 3:33:15 AM
Input Schema
Row Values

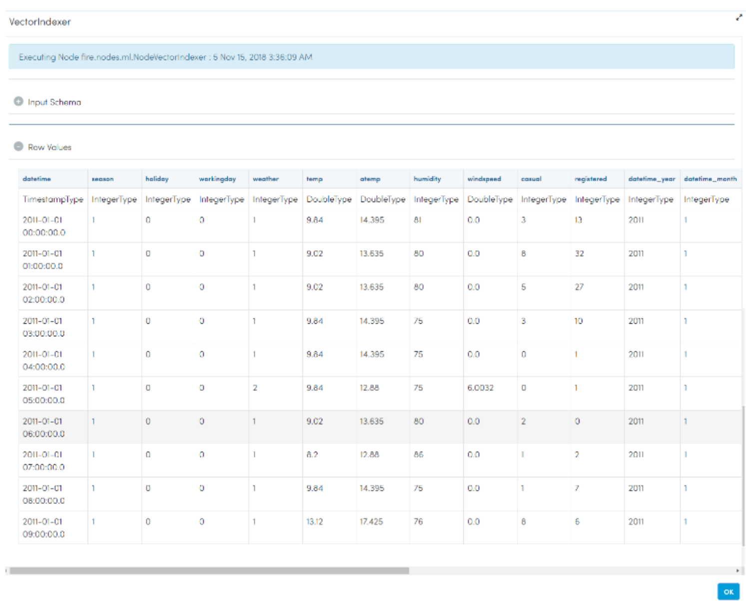
| datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | casual | registered | datetime_year | datetime_month |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TimestampType | IntegerType | IntegerType | IntegerType | IntegerType | DoubleType | DoubleType | IntegerType | DoubleType | IntegerType | IntegerType | IntegerType | IntegerType |
| 2011-01-01 00:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 81 | 0.0 | 3 | 13 | 2011 | 1 |
| 2011-01-01 01:00:00.0 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 8 | 32 | 2011 | 1 |
| 2011-01-01 02:00:00.0 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 5 | 27 | 2011 | 1 |
| 2011-01-01 03:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 3 | 10 | 2011 | 1 |
| 2011-01-01 04:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 0 | 1 | 2011 | 1 |
| 2011-01-01 05:00:00.0 | 1 | 0 | 0 | 2 | 9.84 | 12.88 | 75 | 6.0032 | 0 | 1 | 2011 | 1 |
| 2011-01-01 06:00:00.0 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 2 | 0 | 2011 | 1 |
| 2011-01-01 07:00:00.0 | 1 | 0 | 0 | 1 | 8.2 | 12.88 | 86 | 0.0 | 1 | 2 | 2011 | 1 |
| 2011-01-01 08:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 1 | 7 | 2011 | 1 |
| 2011-01-01 09:00:00.0 | 1 | 0 | 0 | 1 | 13.12 | 17.425 | 76 | 0.0 | 8 | 6 | 2011 | 1 |

OK


VectorIndexer
NodeVectorIndexer
5
SCHEMA :

| COLUMN NAME | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | casual | registered | count | datetime_year | datetime_month | datetime |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COLUMN TYPE | timestamp | integer | integer | integer | integer | double | double | integer | double | integer | integer | double | integer | integer | integer |
| COLUMN FORMAT | dd/MM/yyyy HH:mm | | | | | | | | | | | | | | |

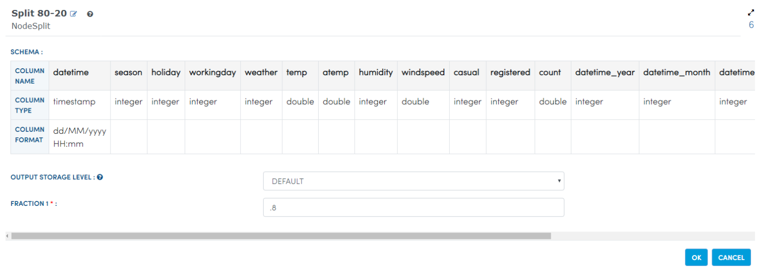
OUTPUT STORAGE LEVEL : DEFAULT
INPUT COLUMN : feature_vector : vectorudt
OUTPUT COLUMN : feature_vector_index
MAXIMUM CATEGORIES : 31
OK
CANCEL


VectorIndexer
Executing Node fire.nodes.ml.NodeVectorIndexer : 5 Nov 15, 2018 3:36:09 AM
Input Schema
Row Values

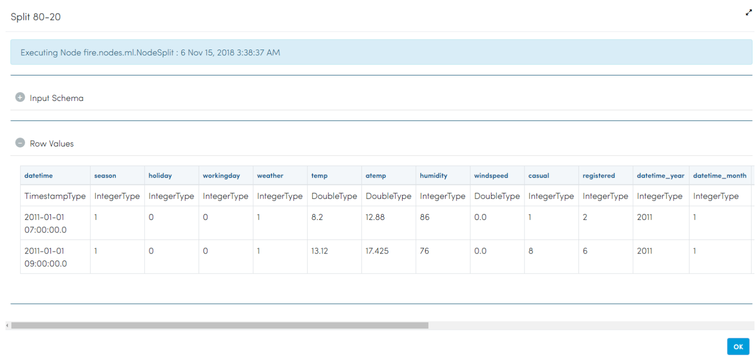
| datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | casual | registered | datetime_year | datetime_month |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TimestampType | IntegerType | IntegerType | IntegerType | IntegerType | DoubleType | DoubleType | IntegerType | DoubleType | IntegerType | IntegerType | IntegerType | IntegerType |
| 2011-01-01 00:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 81 | 0.0 | 3 | 13 | 2011 | 1 |
| 2011-01-01 01:00:00.0 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 8 | 32 | 2011 | 1 |
| 2011-01-01 02:00:00.0 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 5 | 27 | 2011 | 1 |
| 2011-01-01 03:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 3 | 10 | 2011 | 1 |
| 2011-01-01 04:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 0 | 1 | 2011 | 1 |
| 2011-01-01 05:00:00.0 | 1 | 0 | 0 | 2 | 9.84 | 12.88 | 75 | 6.0032 | 0 | 1 | 2011 | 1 |
| 2011-01-01 06:00:00.0 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 2 | 0 | 2011 | 1 |
| 2011-01-01 07:00:00.0 | 1 | 0 | 0 | 1 | 8.2 | 12.88 | 86 | 0.0 | 1 | 2 | 2011 | 1 |
| 2011-01-01 08:00:00.0 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 1 | 7 | 2011 | 1 |
| 2011-01-01 09:00:00.0 | 1 | 0 | 0 | 1 | 13.12 | 17.425 | 76 | 0.0 | 8 | 6 | 2011 | 1 |

OK

### Processor Configuration



### Processor Output



## GBTRegression

It validates held out test sets in order to know about high confidence using GBTRegression Node.

### Processor Configuration

### Processor Output

## Prediction

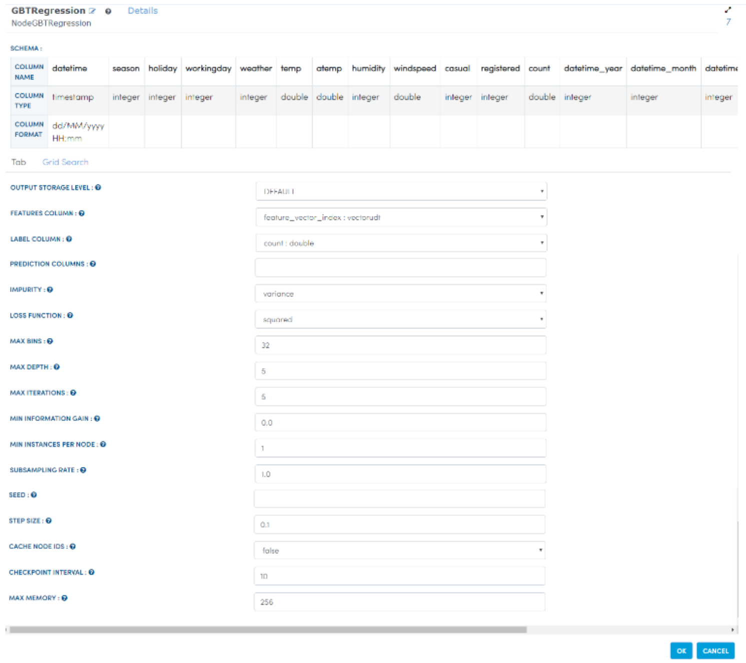
It will make prediction on future data using Prediction Node.

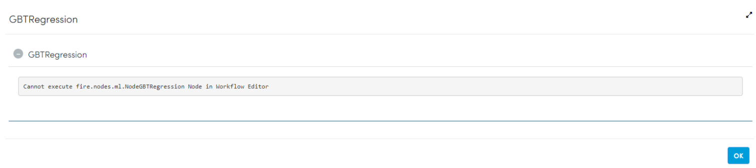
### Processor Configuration

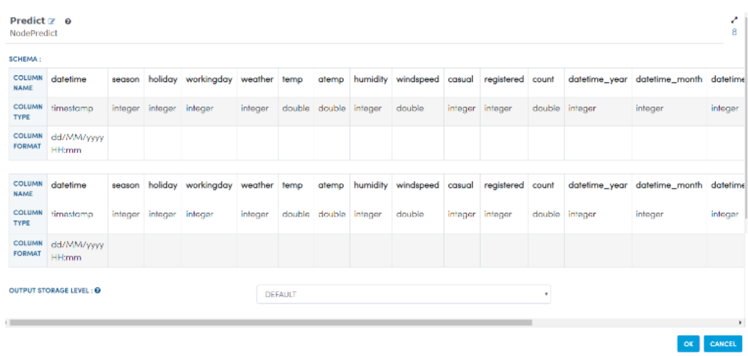
### Processor Output

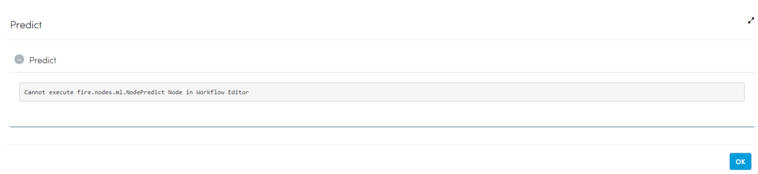
## RegressionEvaluator

It validates held out test sets in order to know about high confidence using RegressionEvaluator Node.

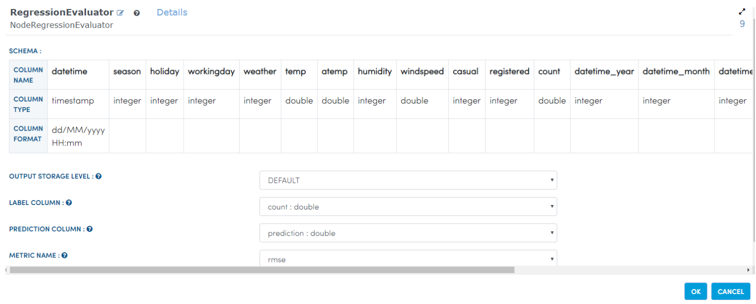

GBTRegression
Details
NodeGBTRegression
SCHEMA :
COLUMN NAME
COLUMN TYPE
COLUMN FORMAT
Tab
Grid Search
OUTPUT STORAGE LEVEL :
FEATURES COLUMN :
LABEL COLUMN :
PREDICTION COLUMNS :
IMPURITY :
LOSS FUNCTION :
MAX BINS :
MAX DEPTH :
MAX ITERATIONS :
MIN INFORMATION GAIN :
MIN INSTANCES PER NODE :
SUBSAMPLING RATE :
SEED :
STEP SIZE :
CACHE NODE IDS :
CHECKPOINT INTERVAL :
MAX MEMORY :
OK
CANCEL


GBTRegression
GBTRegression
Cannot execute fire.nodes.ml.NodeGBTRegression Node in Workflow Editor
OK

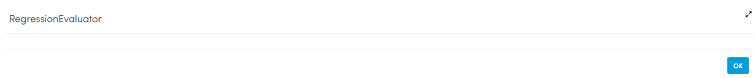

Predict
NodePredict
SCHEMA :
COLUMN NAME
COLUMN TYPE
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
OK
CANCEL


Predict
Predict
Cannot execute fire.nodes.ml.NodePredict Node in Workflow Editor
OK

### Processor Configuration



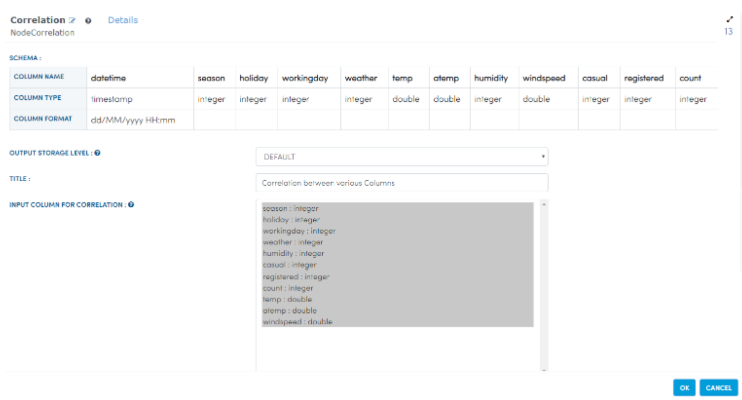

### Processor Output




### Correlation with columns

It will analyse correlation between various columns using Correlation Node.

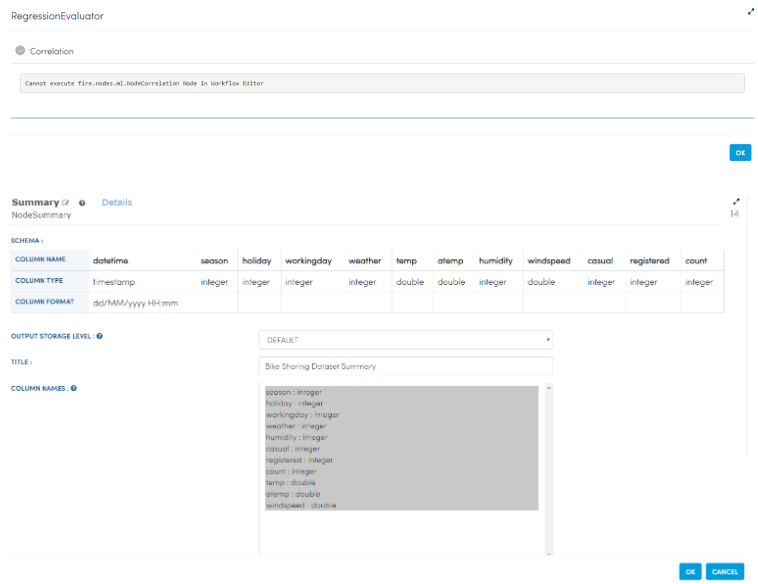
### Processor Configuration




### Processor Output

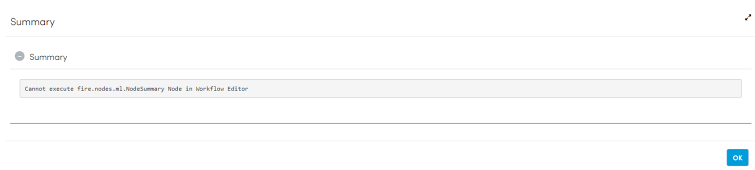
### Summary analysis

It visualizes our data to get sense of whether the features are meaningful using Summary Node.




### Processor Configuration

### Processor Output




### Calculate count for rental per hour

It calculates count for rental per hour using query with SQL Node.

### Processor Configuration

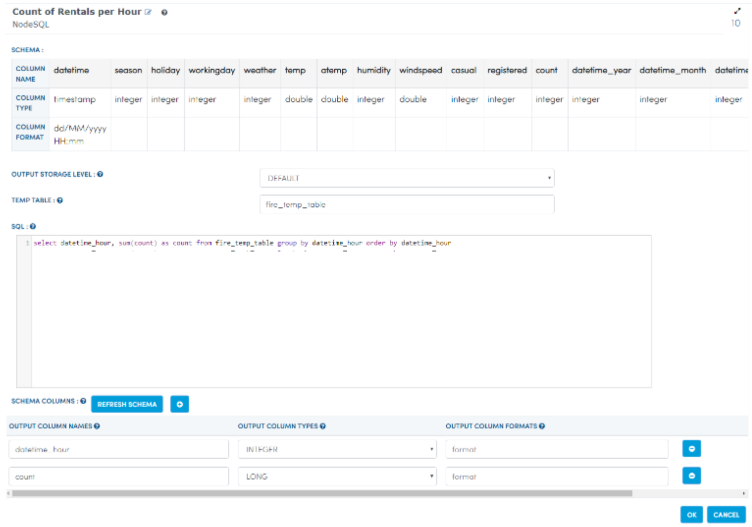
### Processor Output

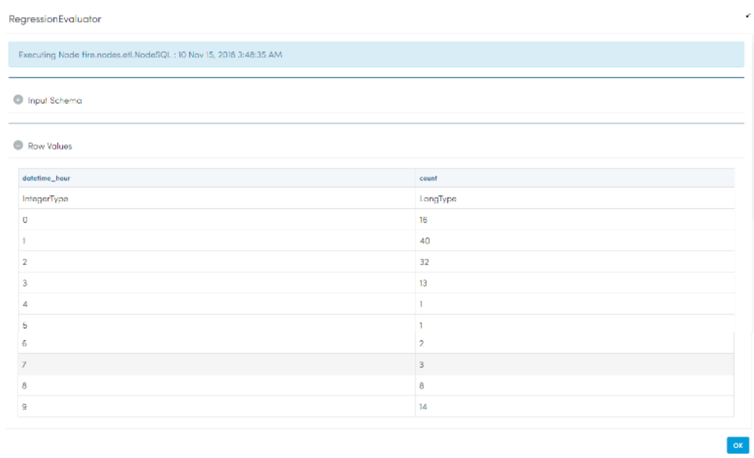
### Analyse using Graph

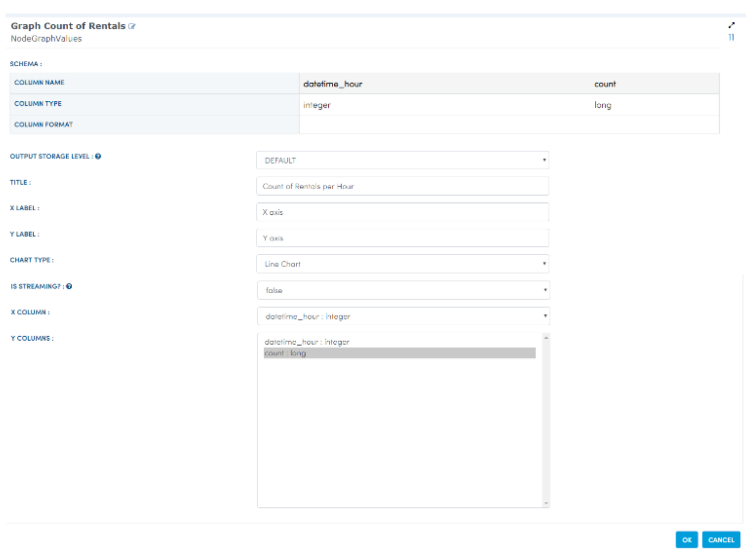
It will analyse graph with bike rental counts and hours of the day using GraphValue Node.

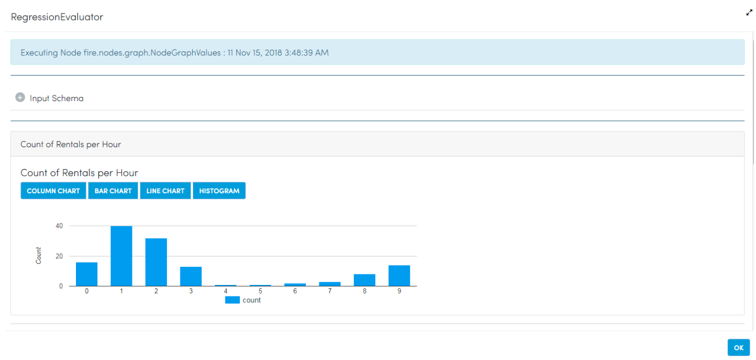
### Processor Configuration

### Processor Output


Count of Rentals per Hour
NodeSQL
SCHEMA :
COLUMN NAME datetime season holiday workingday weather temp atemp humidity windspeed casual registered count datetime_year datetime_month
COLUMN TYPE timestamp integer integer integer integer double double integer double integer integer integer integer integer integer
COLUMN FORMAT dd/MM/yyyy HH:mm
OUTPUT STORAGE LEVEL :
DEFAULT
TEMP TABLE :
fire_temp_table
SQL :
select datetime_hour, sum(count) as count from fire_temp_table group by datetime_hour order by datetime_hour
SCHEMA COLUMNS :
REFRESH SCHEMA
OUTPUT COLUMN NAMES
OUTPUT COLUMN TYPES
OUTPUT COLUMN FORMATS
datetime_hour
INTEGER
format
count
LONG
format
OK
CANCEL


RegressionEvaluator
Executing Node fire.nodes.etl.NodeSQL : 10 Nov 15, 2018 3:48:35 AM
Input Schema
Row Values
datetime_hour count
IntegerType LongType
0 16
1 40
2 32
3 13
4 1
5 1
6 2
7 3
8 8
9 14
OK


Graph Count of Rentals
NodeGraphValues
SCHEMA :
COLUMN NAME datetime_hour count
COLUMN TYPE integer long
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
TITLE :
Count of Rentals per Hour
X LABEL :
X axis
Y LABEL :
Y axis
CHART TYPE :
Line Chart
IS STREAMING? :
false
X COLUMN :
datetime_hour : integer
Y COLUMNS :
datetime_hour : integer
count : long
OK
CANCEL




### Farmers Market Prediction

It demonstrate to predict "the number of farmer's markets in a given zip code" based on the income and taxes paid in a given area using the past data.It seems plausible that areas with higher income have more farmer's markets simply because there is more of a market for those goods. Of course there are many potential holes in this idea, but that's part of the desire to test it.

DataBricks has published a clean approach to build this use case. It feature a Python notebook that demonstrates how to create ML Pipeline to preprocess a dataset, train a Machine Learning model, and make predictions.

Using Fire Insights visual designer, you can try to execute this approach visually and declaratively. This note speaks to that.

As the DataBricks link highlights:

- The first of the two datasets that you can work is the Farmers Markets Directory and Geographic Data. This dataset contains information on the longitude and latitude, state, address, name, and zip code of Farmers Markets in the United States. The raw data is published by the Department of Agriculture. The version on the data that is found in Databricks (and is used in this tutorial) was updated by the Department of Agriculture on

Dec 01, 2015.

- The second you can work is the SOI Tax Stats - Individual Income Tax Statistics - ZIP Code Data (SOI). This study provides detailed tabulations of individual income tax return data at the state and ZIP code level and is provided by the IRS. This repository only has a sample of the data: 2013 and includes "AGI". The ZIP Code data show selected income and tax items classified by State, ZIP Code, and size of adjusted gross income. Data are

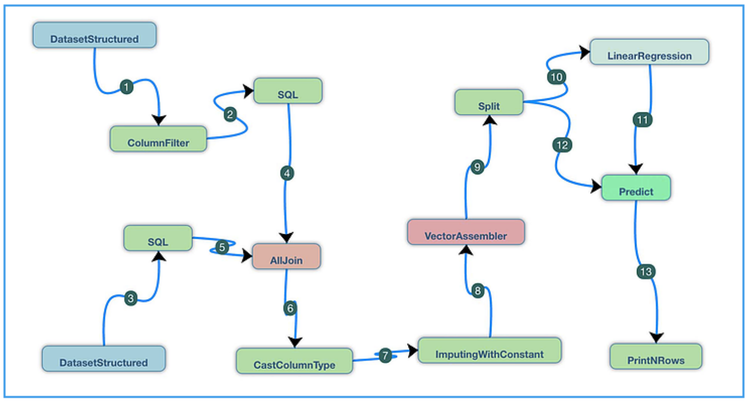
based on individual income tax returns filed with the IRS and are available for Tax Years 1998, 2001, 2004 through 2013. The data include items, such as:

  - Number of returns, which approximates the number of households
  - Number of personal exemptions, which approximates the population
  - Adjusted gross income
  - Wages and salaries
  - Dividends before exclusion
  - Interest received

Below is an overview of the workflow. You can create using the Fire Insights Visual Designer.

This workflow was simply created via the drag and drop capabilities of the Fire Insightss Designer UI. This ability to construct this data processing pipeline (or any DAG - Distributed Acyclic Graph, for that matter) in a WYSIWYG Plug-and-Play manner is a key innovation to continue our community's collective march to on-demand-instant-analytics. Benefits include:

- It opens up the power of ETL and ML (such pre-packaged functionality is available as a catalog of "Nodes") to a wider audience of analysts and semi-technical resources.
- The actual execution can either be local (testing) or can be submitted to a Apache Spark cluster.
- You can see during the adoption that a single workbench improves collaborative iteration across data engineers, data scientists and analysts, which in turn accelerates time-to-market.
- As one might observe, the visual approach doubles up as workflow documentation and hence contributes to solving the data-lineage problem.




This workflow consists of the following steps:

- Using the DatasetStructured Nodes: Read in the data from 2 different datasets - Farmers_Markets and Income Tax Return Data per Zip Code (both comma separated files:
- Instead of a CSV, one can easily read it from a data-lake or a Persistence Store (HDFS/RDBMS/NoSQL).
- Using the ColumnFilter node: Filter out the following columns from the Income Tax Return dataset and pass it to a SQL query node, so we can do further computation.

– State

– Zipcode

– MARS1 - Single Returns

– MARS2 - Joint Returns

– NUMDEP - Number of Dependents

– A02650 - Tota Income Amount

– A00300 - Taxable Interest Amount

– A00900

– A01000

- Using the SQL Node: Execute the following SQL to get the various aggregates from the filtered data from the Income Tax Return dataset
  - select zipcode, sum(MARS1) as single_returns, sum(MARS2) as joint_returns, sum(NUMDEP) as numdep, sum(A02650) as total_income_amount, sum(A00300) as taxable_interest_amount from fire_temp_table group by zipcode
- Using another SQL Node: Extract certain columns from the Farmers_Market dataset using the below SQL query:
  - select cast(zip as int) as zip, count(*) as count from fire_temp_table group by zip
- Using the AllJoin node - Join the two filtered datasets using the following query:
  - select a.zipcode , a.single_returns, a.joint_returns, a.numdep, a.total_income_amount, a.taxable_interest_amount, b.count, b.zip from fire_temp_table1 a LEFT OUTER JOIN fire_temp_table2 b ON(a.zipcode=b.zip)
- Using the CastColumnType Node - change the column type of the count column from Long to Double
- Using the ImputingWithConstant node, fill the blanks across all columns with constants.
- Using the VectorAssembler node, concatenate columns single_returns, joint_returns, numdep, total_income_amount, taxable_interest_amount into a feature vector feature_vector
- Using Split node: Split the dataset into (.7, .3)
  - 70% rows are used for training and 30% are used for prediction
- The model is evaluated based on how it predicts on the remaining 30%.
- Using the LinearRegression Node - Perform LinearRegression:
- This is a Spark MLLib provided algorithm that Sparkflows exposes to you as a plug-and-play "node". LinearRegression from SparkML.
- Using Predict Node: Perform prediction using the model generated on the remaining 30% dataset
- Finally evaluate the result using the PrintNRows node.

#### First Dataset

#### Column Filter

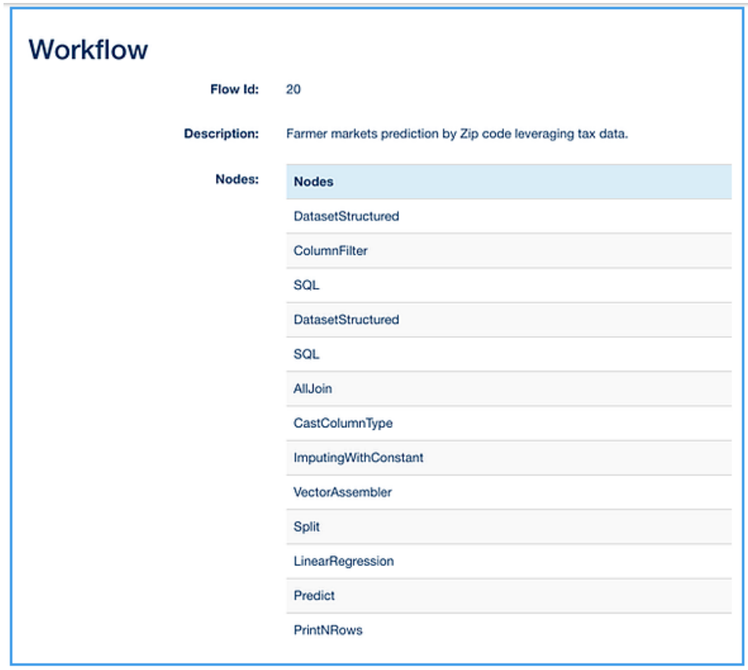
#### SQL

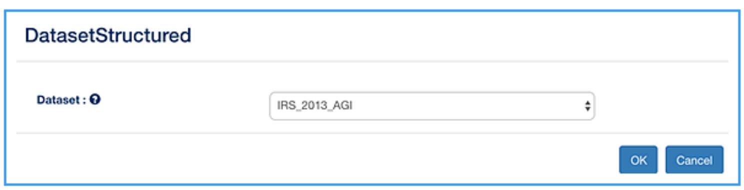
#### Second Dataset

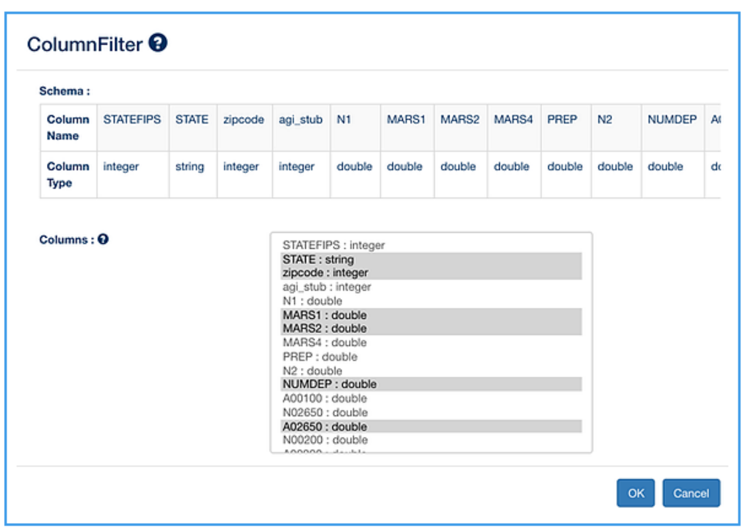
#### SQL

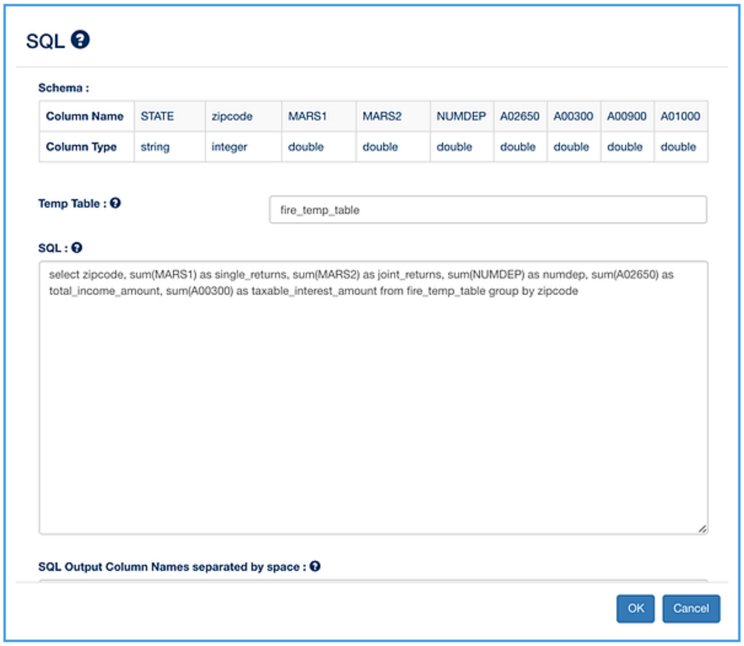
#### AllJoin - Join the two datasets

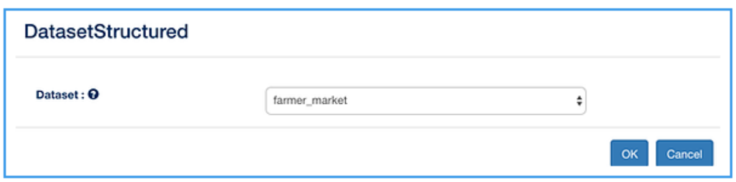
#### CastColumnType

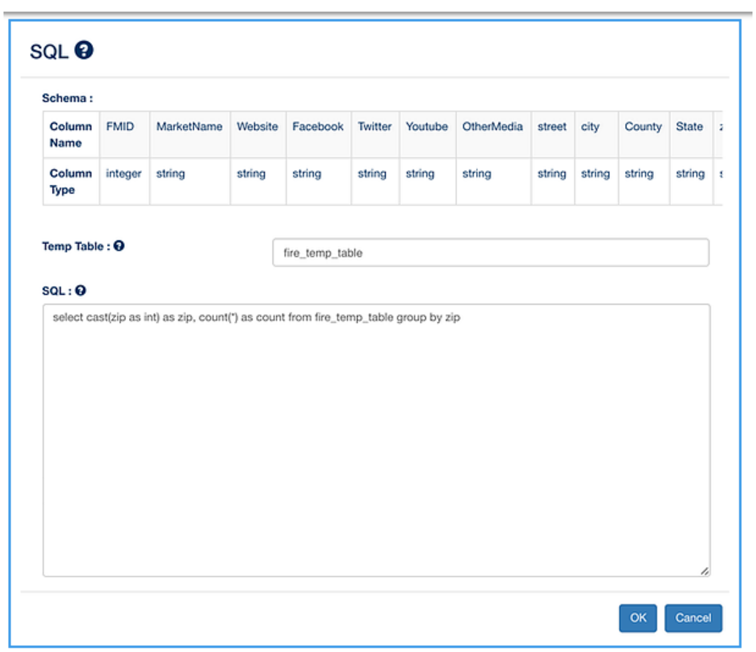

Workflow
Flow Id:
20
Description:
Farmer markets prediction by Zip code leveraging tax data.
Nodes:
Nodes
DatasetStructured
ColumnFilter
SQL
DatasetStructured
SQL
AllJoin
CastColumnType
ImputingWithConstant
VectorAssembler
Split
LinearRegression
Predict
PrintNRows

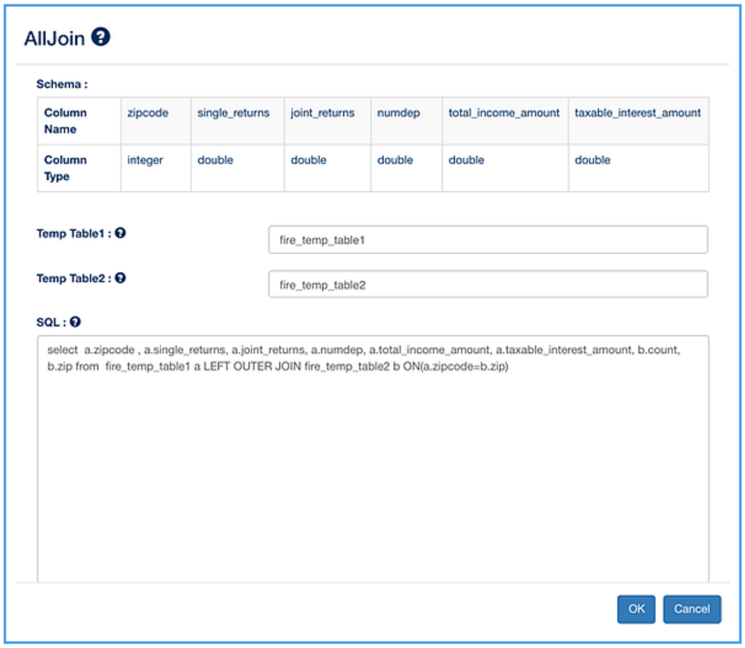

DatasetStructured
Dataset :
IRS_2013_AGI
OK
Cancel

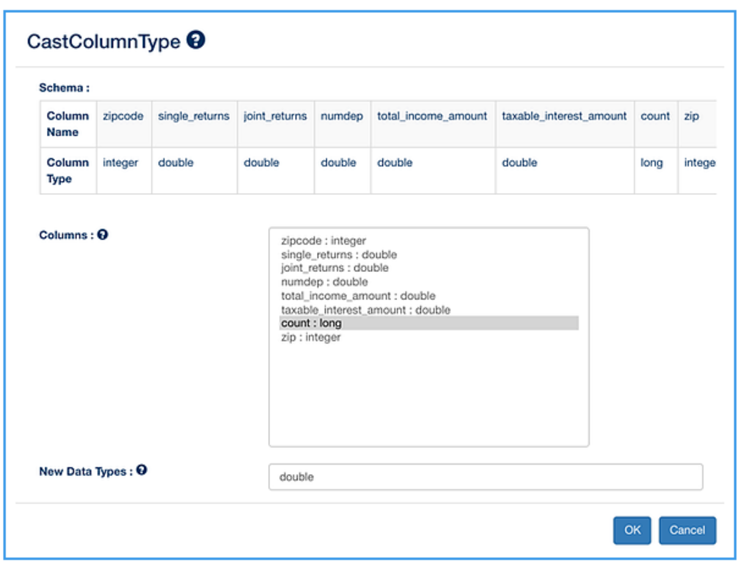

ColumnFilter
Schema :
Column Name
STATEFIPS
STATE
zipcode
agi_stub
N1
MARS1
MARS2
MARS4
PREP
N2
NUMDEP
Column Type
integer
string
integer
integer
double
double
double
double
double
double
double
Columns :
STATEFIPS : integer
STATE : string
zipcode : integer
agi_stub : integer
N1 : double
MARS1 : double
MARS2 : double
MARS4 : double
PREP : double
N2 : double
NUMDEP : double
A00100 : double
N02650 : double
A02650 : double
N00200 : double
OK
Cancel


SQL
Schema :
Column Name
STATE
zipcode
MARS1
MARS2
NUMDEP
A02650
A00300
A00900
A01000
Column Type
string
integer
double
double
double
double
double
double
double
Temp Table :
fire_temp_table
SQL :
select zipcode, sum(MARS1) as single_returns, sum(MARS2) as joint_returns, sum(NUMDEP) as numdep, sum(A02650) as total_income_amount, sum(A00300) as taxable_interest_amount from fire_temp_table group by zipcode
SQL Output Column Names separated by space :
OK
Cancel

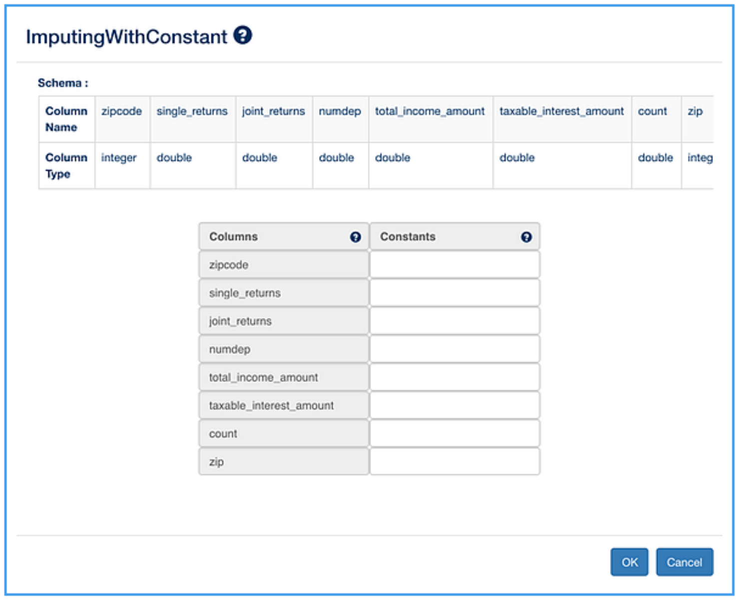

DatasetStructured
Dataset :
farmer_market
OK
Cancel


SQL
Schema :
Column Name
FMID
MarketName
Website
Facebook
Twitter
Youtube
OtherMedia
street
city
County
State
Column Type
integer
string
string
string
string
string
string
string
string
string
string
Temp Table :
fire_temp_table
SQL :
select cast(zip as int) as zip, count(*) as count from fire_temp_table group by zip
OK
Cancel

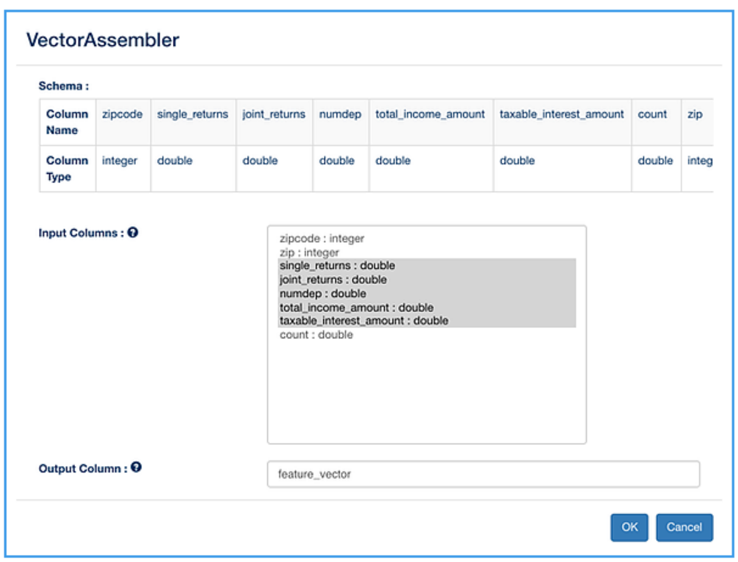

AllJoin
Schema :
Column Name
zipcode
single_returns
joint_returns
numdep
total_income_amount
taxable_interest_amount
Column Type
integer
double
double
double
double
double
Temp Table1 :
fire_temp_table1
Temp Table2 :
fire_temp_table2
SQL :
select a.zipcode , a.single_returns, a.joint_returns, a.numdep, a.total_income_amount, a.taxable_interest_amount, b.count, b.zip from fire_temp_table1 a LEFT OUTER JOIN fire_temp_table2 b ON(a.zipcode=b.zip)
OK
Cancel


CastColumnType
Schema :
Column Name
zipcode
single_returns
joint_returns
numdep
total_income_amount
taxable_interest_amount
count
zip
Column Type
integer
double
double
double
double
double
long
intege
Columns :
zipcode : integer
single_returns : double
joint_returns : double
numdep : double
total_income_amount : double
taxable_interest_amount : double
count : long
zip : integer
New Data Types :
double
OK
Cancel

### ImputingWithConstant



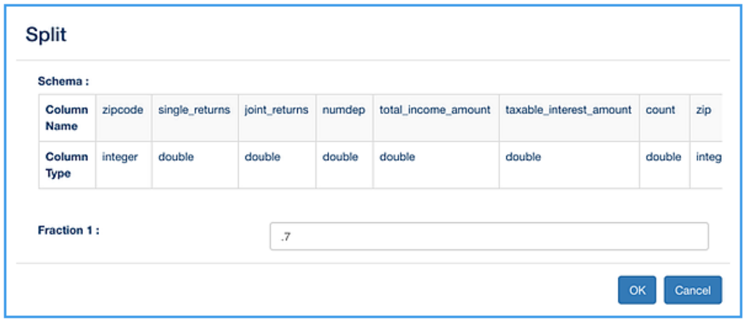

### VectorAssembler

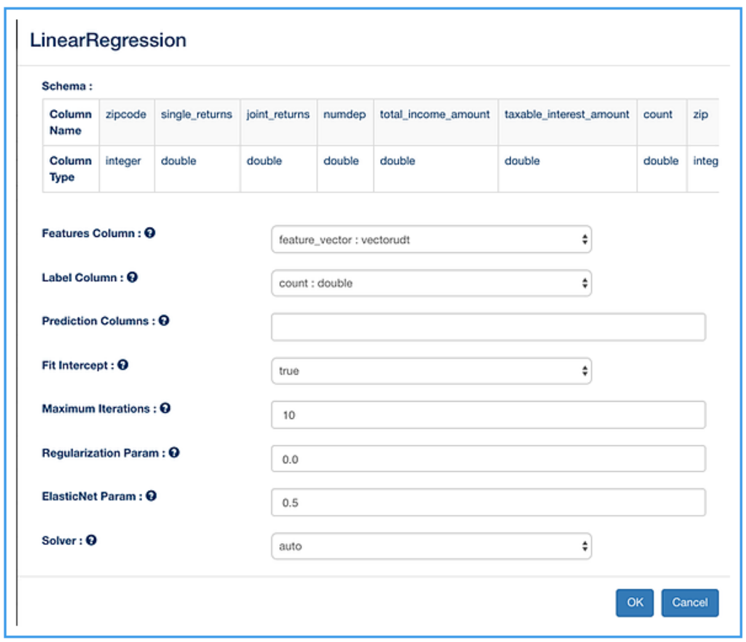


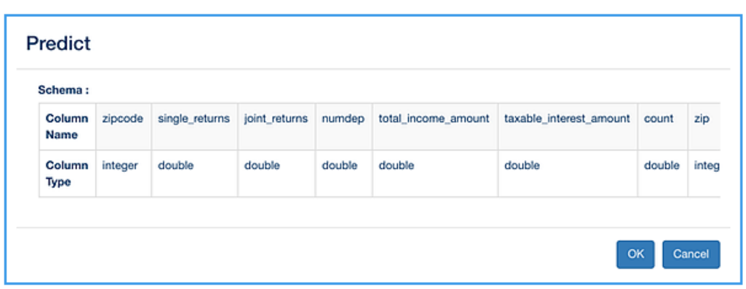

### Split

### LinearRegression

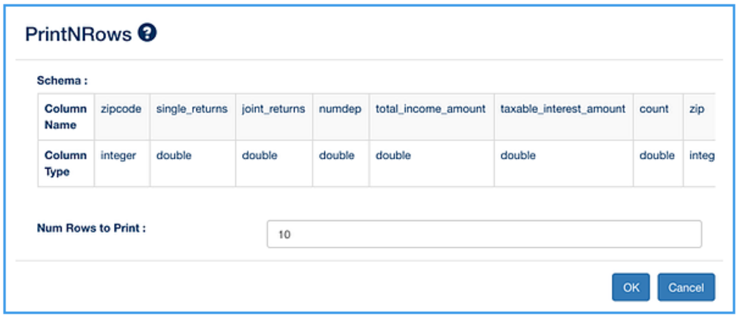
### Predict


Split
Schema :
Column Name
zipcode
single_returns
joint_returns
numdep
total_income_amount
taxable_interest_amount
count
zip
Column Type
integer
double
double
double
double
double
double
integ
Fraction 1 :
.7
OK
Cancel

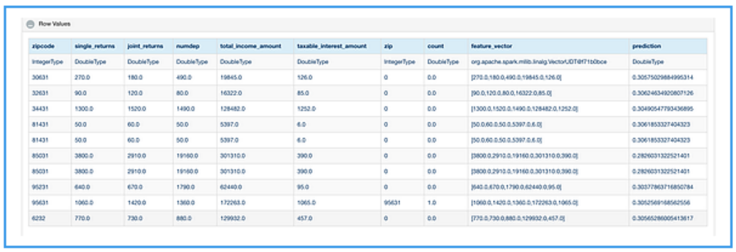

LinearRegression
Schema :
Column Name
zipcode
single_returns
joint_returns
numdep
total_income_amount
taxable_interest_amount
count
zip
Column Type
integer
double
double
double
double
double
double
integ
Features Column :
feature_vector : vectorudt
Label Column :
count : double
Prediction Columns :
Fit Intercept :
true
Maximum Iterations :
10
Regularization Param :
0.0
ElasticNet Param :
0.5
Solver :
auto
OK
Cancel


Predict
Schema :
Column Name
zipcode
single_returns
joint_returns
numdep
total_income_amount
taxable_interest_amount
count
zip
Column Type
integer
double
double
double
double
double
double
integ
OK
Cancel

### Print N Rows




Next you can execute the workflow and it come up with predictions for number of farmers markets in a zip code.




### Clustering Houses

This workflow reads in a dataset. It then performs KMeans Clustering on the Housing Dataset.

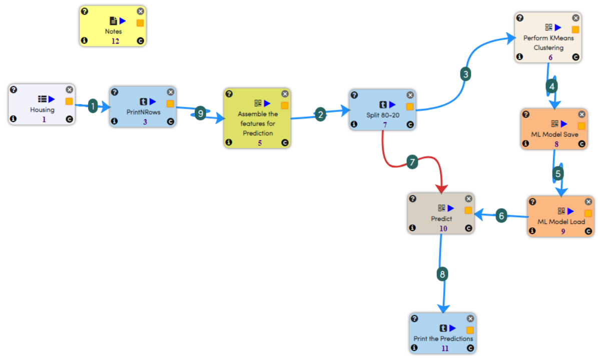
### Workflow

Below is the workflow. It does the following:

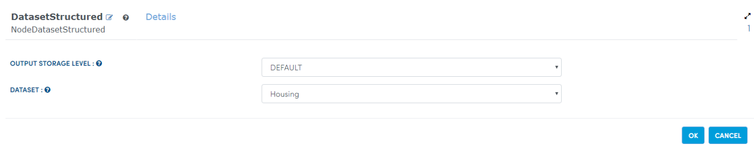
- Reads data from a sample dataset.
- Prints the results.
- Assembles the features for predictions.
- Splits it.
- Perform KMeans Clustering.
- ML Model save.
- ML Model Load.
- Prediction.
- Print the prediction results.

### Reading from Dataset

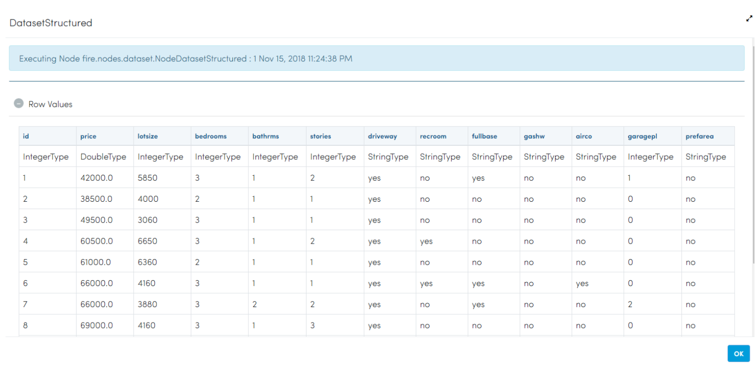
It reads sample Dataset file.




## Processor Configuration




## Processor Output


DatasetStructured

Executing Node fire.nodes.dataset.NodeDatasetStructured : 1 Nov 15, 2018 11:24:38 PM

Row Values

| id | price | lotsize | bedrooms | bathrms | stories | driveway | recroom | fullbase | gashw | airco | garagepl | prefarea |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | DoubleType | IntegerType | IntegerType | IntegerType | IntegerType | StringType | StringType | StringType | StringType | StringType | IntegerType | StringType |
| 1 | 42000.0 | 5850 | 3 | 1 | 2 | yes | no | yes | no | no | 1 | no |
| 2 | 38500.0 | 4000 | 2 | 1 | 1 | yes | no | no | no | no | 0 | no |
| 3 | 49500.0 | 3060 | 3 | 1 | 1 | yes | no | no | no | no | 0 | no |
| 4 | 60500.0 | 6650 | 3 | 1 | 2 | yes | yes | no | no | no | 0 | no |
| 5 | 61000.0 | 6360 | 2 | 1 | 1 | yes | no | no | no | no | 0 | no |
| 6 | 66000.0 | 4160 | 3 | 1 | 1 | yes | yes | yes | no | yes | 0 | no |
| 7 | 66000.0 | 3880 | 3 | 2 | 2 | yes | no | yes | no | no | 2 | no |
| 8 | 69000.0 | 4160 | 3 | 1 | 3 | yes | no | no | no | no | 0 | no |

OK

## Prints the results

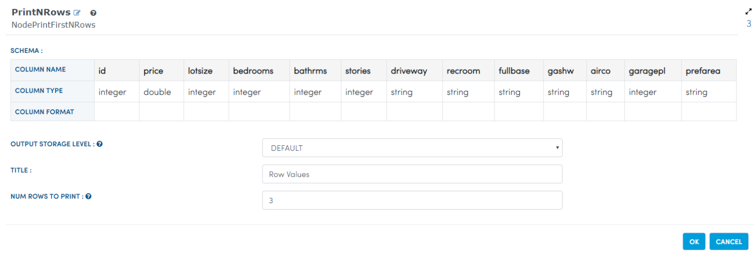
It prints the sample dataset file results.

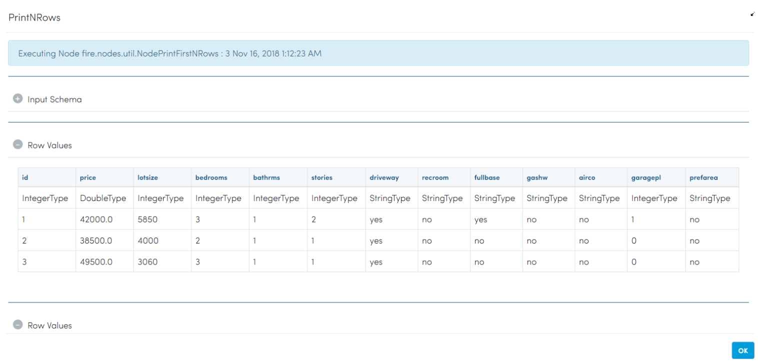
## Processor Configuration

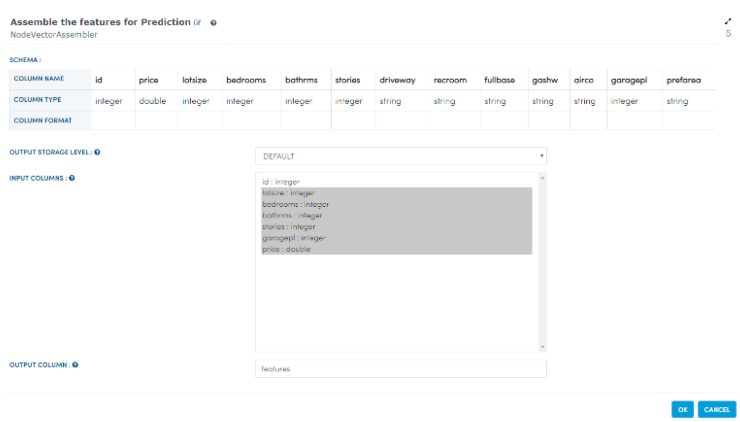
## Processor Output

## Assemble the features for predictions

It assembles the features for predictions using VectorAssembler Node.

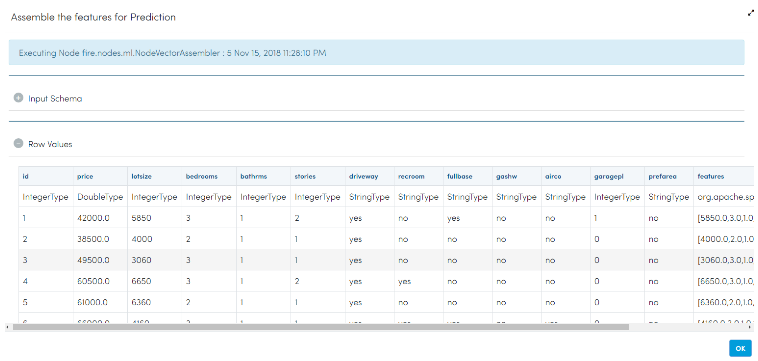

PrintNRows
NodePrintFirstNRows
SCHEMA :
COLUMN NAME id price lotsize bedrooms bathrms stories driveway recroom fullbase gashw airco garagepl prefarea
COLUMN TYPE integer double integer integer integer integer string string string string string integer string
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
TITLE :
Row Values
NUM ROWS TO PRINT :
3
OK
CANCEL


PrintNRows
Executing Node fire.nodes.util.NodePrintFirstNRows : 3 Nov 16, 2018 1:12:23 AM
Input Schema
Row Values
id price lotsize bedrooms bathrms stories driveway recroom fullbase gashw airco garagepl prefarea
IntegerType DoubleType IntegerType IntegerType IntegerType IntegerType StringType StringType StringType StringType StringType IntegerType StringType
1 42000.0 5850 3 1 2 yes no yes no no 1 no
2 38500.0 4000 2 1 1 yes no no no no 0 no
3 49500.0 3060 3 1 1 yes no no no no 0 no
Row Values
OK


Assemble the features for Prediction
NodeVectorAssembler
SCHEMA :
COLUMN NAME id price lotsize bedrooms bathrms stories driveway recroom fullbase gashw airco garagepl prefarea
COLUMN TYPE integer double integer integer integer integer string string string string string integer string
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
INPUT COLUMNS :
id : integer
lotsize : integer
bedrooms : integer
bathrms : integer
stories : integer
garagepl : integer
price : double
OUTPUT COLUMN :
features
OK
CANCEL

### Processor Configuration

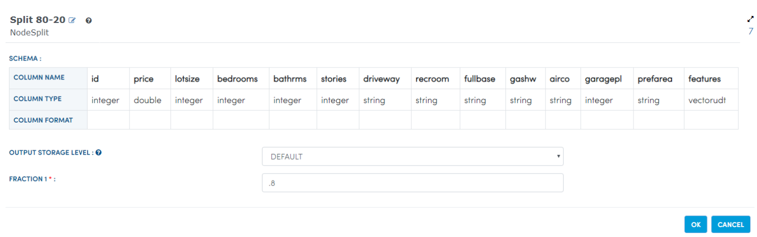
### Processor Output



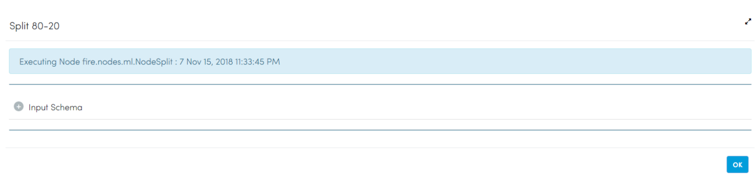

### Split it

It splits features of prediction using Split Node.

### Processor Configuration




### Processor Output

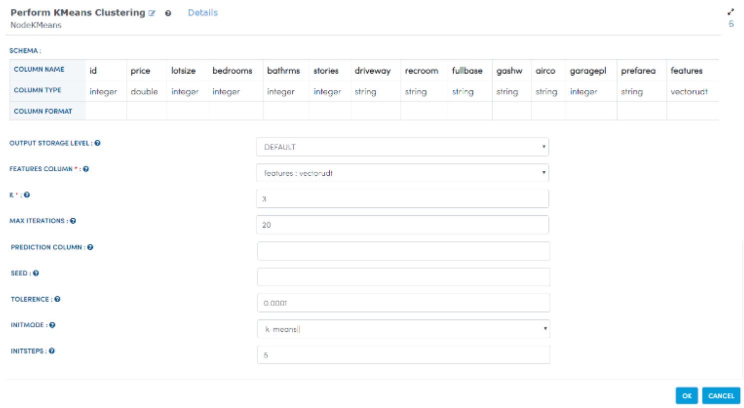


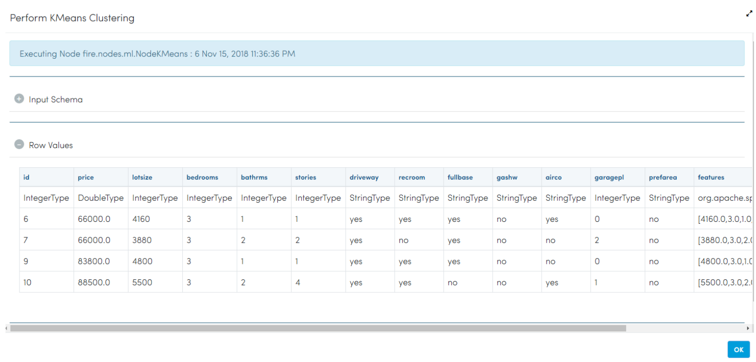

### Perform KMeans Clustering

It performs KMeans Clustering on the Housing Dataset using KMeans Node.

### Processor Configuration

### Processor Output

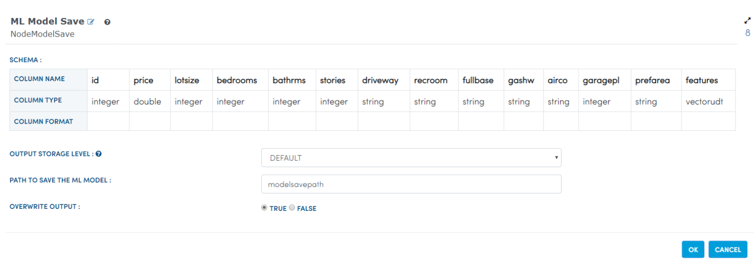



## ML Model save

It will save ML Model with given path using ModelSave Node.

## Processor Configuration

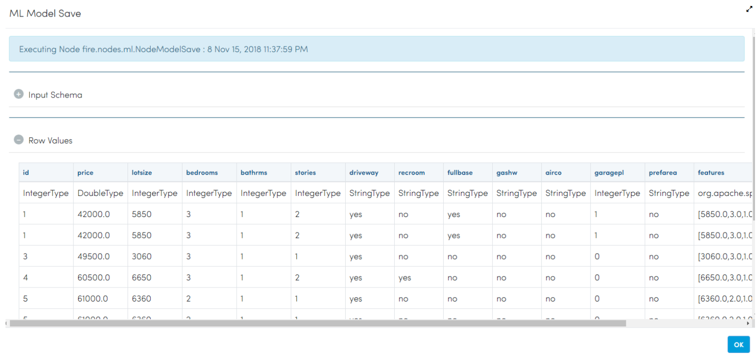


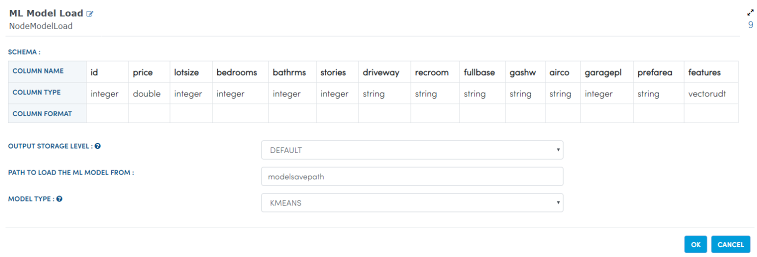

## Processor Output

## ML Model Load

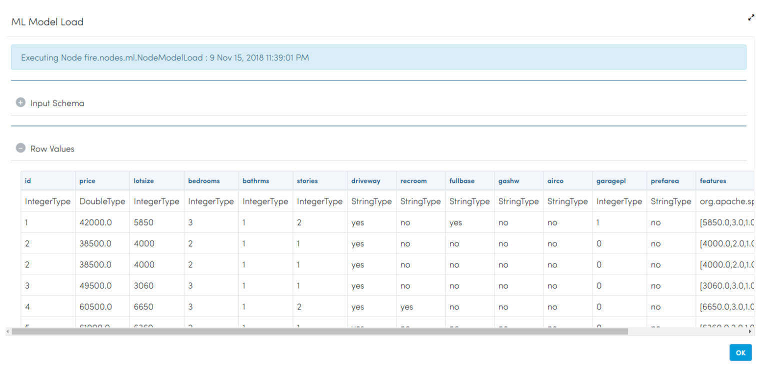
It will Load ML Model with given path using ModelSave Node.

## Processor Configuration




## Processor Output




## Prediction

It predicts features updated using Predict Node.

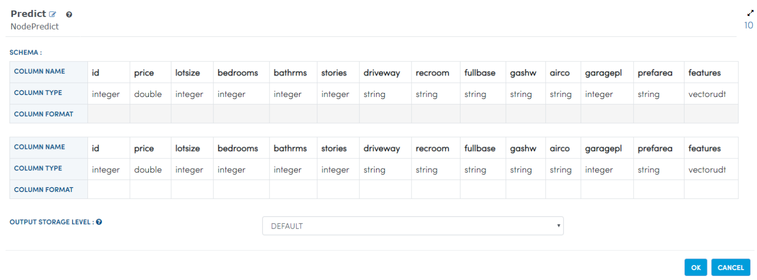
## Processor Configuration

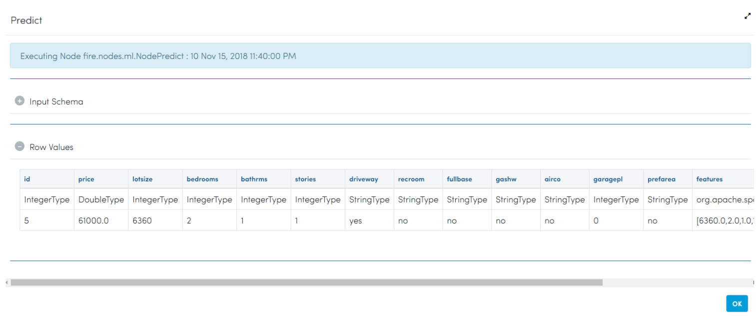
## Processor Output

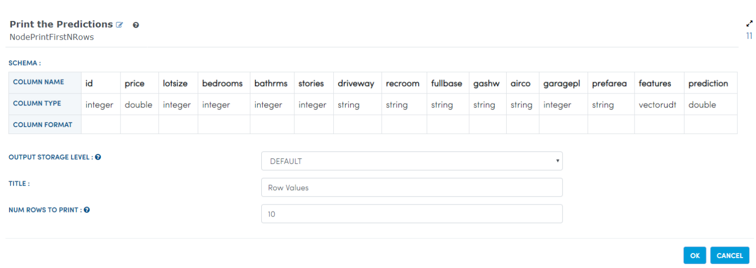
## Print the prediction results

It Print the prediction results.

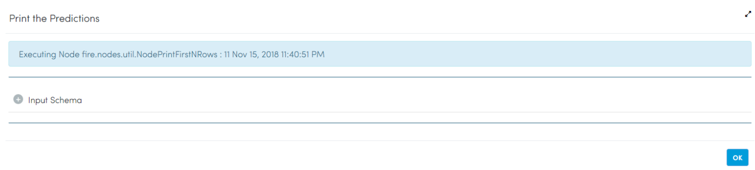
## Processor Configuration


Predict
NodePredict
10
SCHEMA :
COLUMN NAME
COLUMN TYPE
COLUMN FORMAT
id price lotsize bedrooms bathrms stories driveway recroom fullbase gashw airco garagepl prefarea features
integer double integer integer integer integer string string string string string integer string vectorudt
OUTPUT STORAGE LEVEL :
DEFAULT
OK
CANCEL


Predict
Executing Node fire.nodes.ml.NodePredict : 10 Nov 15, 2018 11:40:00 PM
Input Schema
Row Values
id price lotsize bedrooms bathrms stories driveway recroom fullbase gashw airco garagepl prefarea features
IntegerType DoubleType IntegerType IntegerType IntegerType IntegerType StringType StringType StringType StringType StringType IntegerType StringType
5 61000.0 6360 2 1 1 yes no no no no 0 no
OK


Print the Predictions
NodePrintFirstNRows
11
SCHEMA :
COLUMN NAME
COLUMN TYPE
COLUMN FORMAT
id price lotsize bedrooms bathrms stories driveway recroom fullbase gashw airco garagepl prefarea features prediction
integer double integer integer integer integer string string string string string integer string vectorudt double
OUTPUT STORAGE LEVEL :
DEFAULT
TITLE :
Row Values
NUM ROWS TO PRINT :
10
OK
CANCEL

### Processor Output



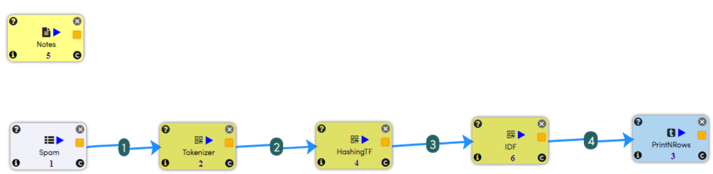

## TFIDF

This workflow reads in a dataset. It then Tokenizes and then performs TF/IDF on text content.

### Workflow

Below is the workflow. It does the following:

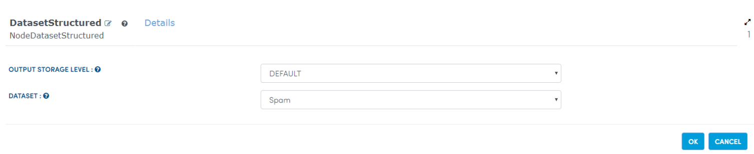
- Reads data from a sample dataset.
- Tokenizes message column.
- Performs TF.
- Performs IDF.
- Prints the results.

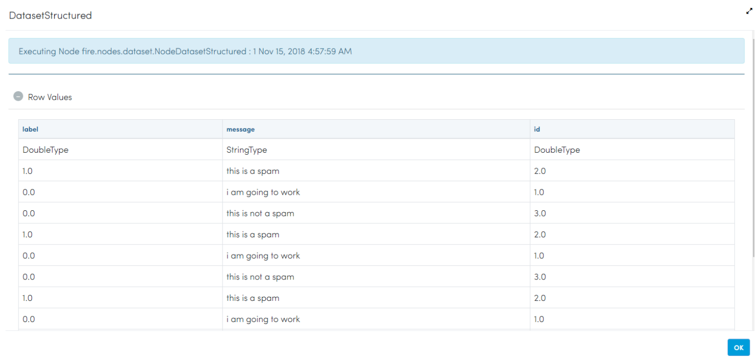



### Reading from Dataset

It reads sample Dataset file.

### Processor Configuration



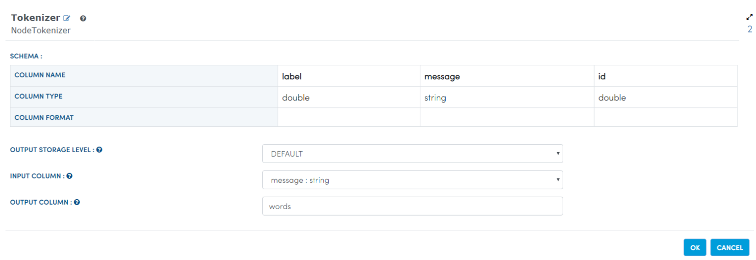



## Processor Output

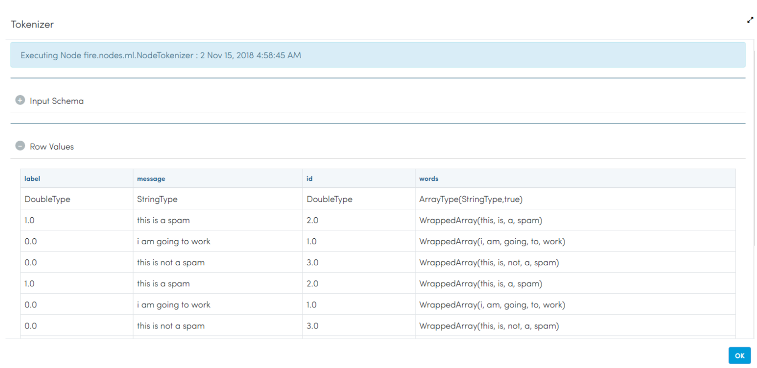
## Tokenizes message column

It Tokenizes message column generated by sample dataset file using Tokenizer Node.

## Processor Configuration



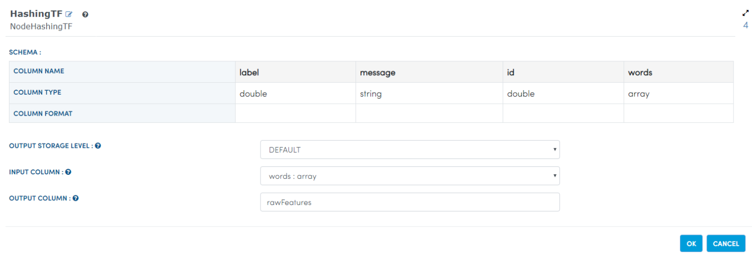

## Processor Output



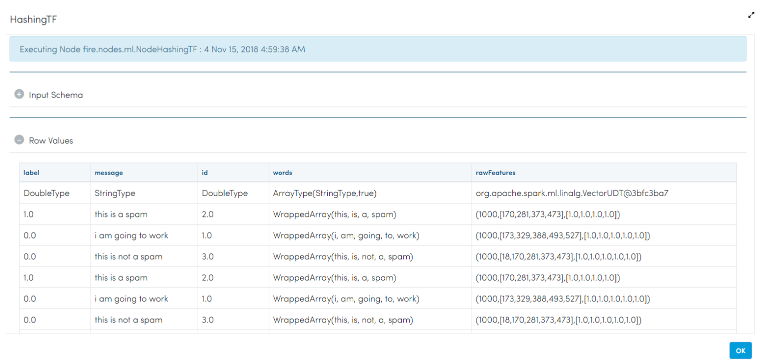

## Perform TF

It performs TF on text column using HashingTF Node.

## Processor Configuration



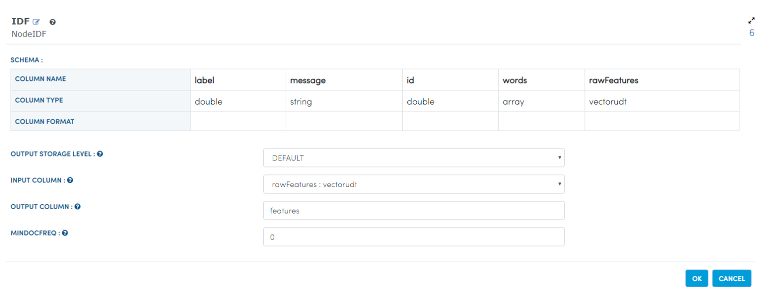

## Processor Output




## Perform IDF

It performs IDF on text column using IDF Node.

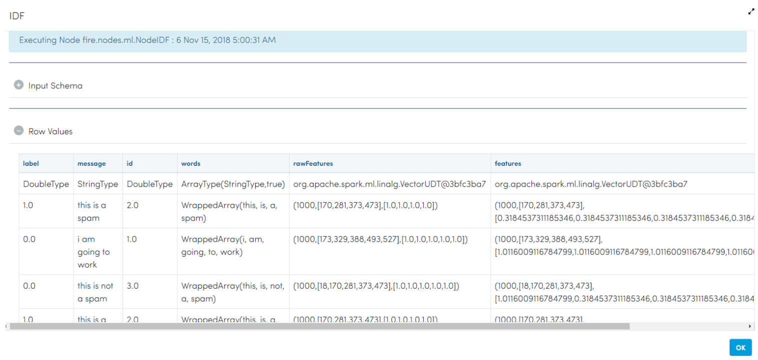
## Processor Configuration

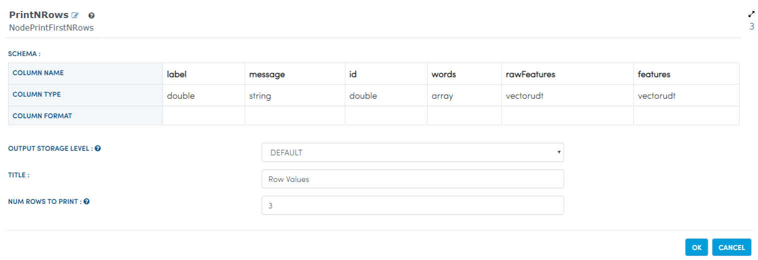



## Processor Output

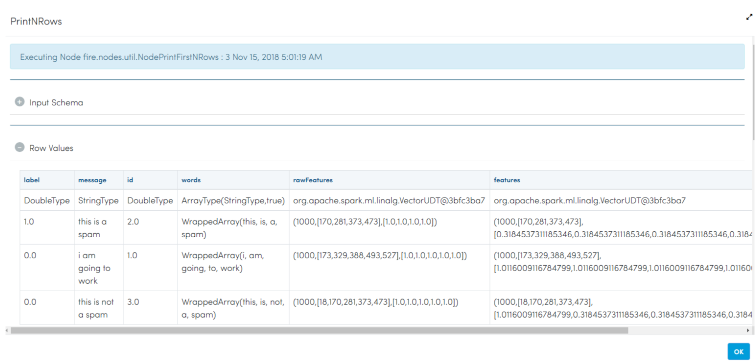
## Prints the results

It will print the result after performing TF/IDF on text content.




## Processor Output




## Earthquake Prediction

### Objective

As the motivation behind earthquake prediction is to empower crisis measures to decrease demise and devastation, inability to give notice of a significant earthquake that happens, or possibly a satisfactory assessment of the hazard, can bring about legitimate risk, or even political cleansing.

### Dataset

Dataset contains 2 columns as below:

- Acoustic_data - Acoustic wave reading
- Time_to_failure - Time remaining before the next earthquake

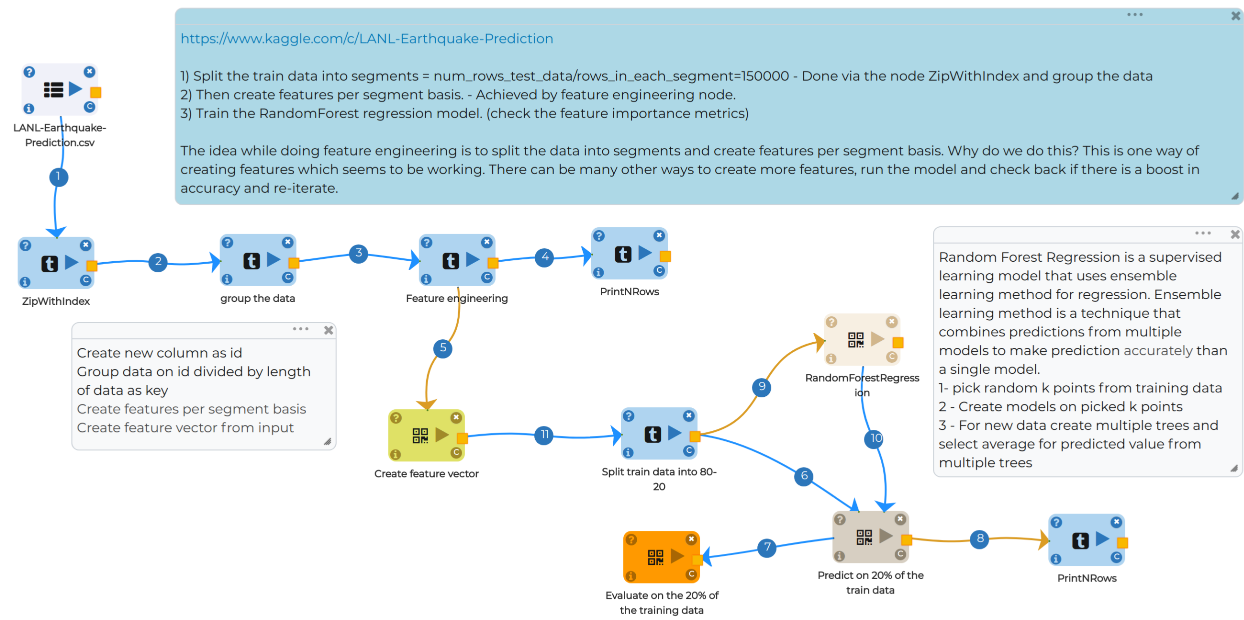
| acoustic_data | time_to_failure |
|---|---|
| IntegerType | DoubleType |
| 12 | 1.469 |
| 6 | 1.469 |
| 8 | 1.469 |
| 5 | 1.469 |
| 8 | 1.469 |
| 8 | 1.469 |
| 9 | 1.469 |
| 7 | 1.469 |
| -5 | 1.469 |
| 3 | 1.469 |
| 5 | 1.469 |
| 2 | 1.469 |
| 2 | 1.469 |
| 3 | 1.469 |

### Random Forest Regression Workflow for Earthquake Prediction

Random Forest Regression model belongs to family of bagging regression. It is a supervised learning model that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple models to make prediction more accurately than a single model.

Features of Random Forest -

- Aggregates many decision trees
- Prevents overfitting




### Prepare data for modeling

Follow workflow arrow

- **ZipWithIndex**- Creates new feature column from dataframe index as ID
- **Group data**- Creates new feature column as key obtained by ID divided by length of data
- **Feature Engineering**- Groups by data on key to create all statistical measures (min, max, mean, quartiles etc) as new feature
- **Feature Vector** - Merge multiple columns to form vector

### Data modeling

- Before we create Random Forest Regression model, split data (80:20) into train and test for performance evaluation.

Input Schema

| acoustic_data | time_to_failure | id |
|---|---|---|
| IntegerType | DoubleType | LongType |

Row Values

Row Values

| acoustic_data | time_to_failure | id | key |
|---|---|---|---|
| IntegerType | DoubleType | LongType | DoubleType |
| 12 | 1.469 | 0 | 0.0 |
| 6 | 1.469 | 1 | 0.0 |
| 8 | 1.469 | 2 | 0.0 |
| 5 | 1.469 | 3 | 0.0 |
| 8 | 1.469 | 4 | 0.0 |
| 8 | 1.469 | 5 | 0.0 |
| 9 | 1.469 | 6 | 0.0 |
| 7 | 1.469 | 7 | 0.0 |
| -5 | 1.469 | 8 | 0.0 |
| 3 | 1.469 | 9 | 0.0 |

Input Schema

| acoustic_data | time_to_failure | id | key |
|---|---|---|---|
| IntegerType | DoubleType | LongType | DoubleType |

Row Values

Row Values

| segment | max_a | min_a | avg_a | std_a | var_a | p_50 | p_25 | p_75 | time_to_failure_label |
|---|---|---|---|---|---|---|---|---|---|
| DoubleType | IntegerType | IntegerType | DoubleType | DoubleType | DoubleType | DoubleType | DoubleType | DoubleType | DoubleType |
| 0.0 | 14 | -5 | 5.28 | 3.344 | 11.185 | 5.0 | 3.0 | 7.75 | 1.469 |
| 1.0 | 13 | -4 | 5.88 | 3.612 | 13.047 | 6.0 | 4.0 | 8.75 | 1.469 |

INPUT COLUMNS :

Available

time_to_failure_label : double

Selected

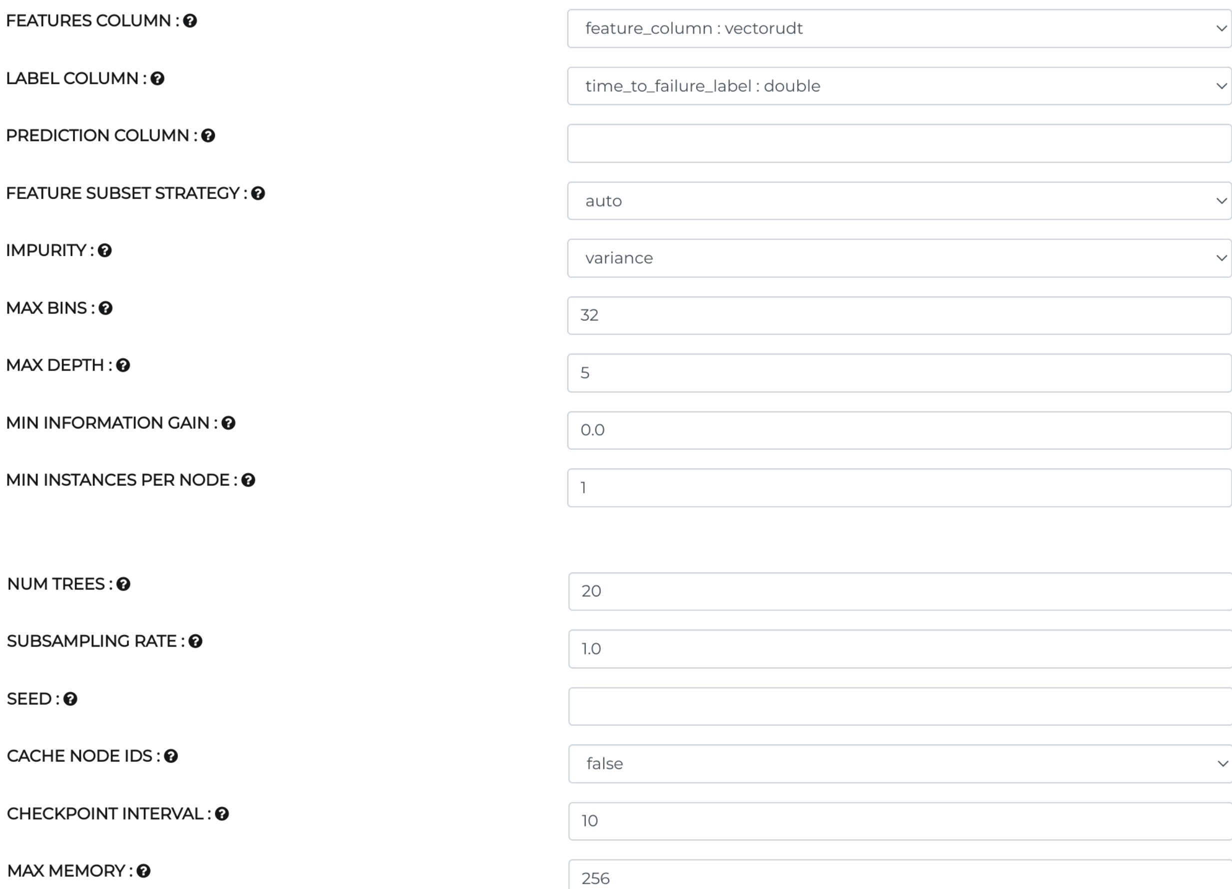
segment : double
max_a : integer
min_a : integer
avg_a : double
std_a : double
var_a : double
p_50 : double
p_25 : double
p_75 : double

OUTPUT COLUMN * :

feature_column

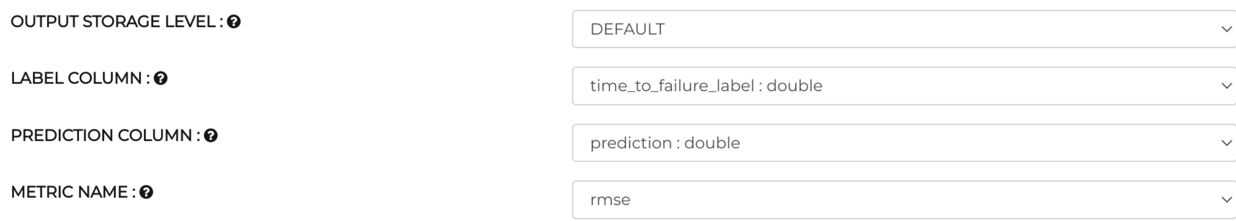
### Random Forest Regression

- Sets feature vector corresponding to label(time_to_failure_label).
- Sets number of features for each split node of tree.
- For regression the measure of impurity is variant.
- In random forest, the impurity decrease from each feature can be averaged across trees to determine the final importance of the variable.
- The maxBins signifies the maximum number of bins used for splitting the features, where the suggested value is 100 to get better results.
- The maxDepth is the maximum depth of the tree (for example, depth 0 means one leaf node, depth 1 means one internal node plus two leaf nodes).
- Information gain is calculated by comparing the entropy of the dataset before and after a transformation.




### Model evaluation

- Multiple ways to evaluate regression model such as R square, Root mean square error(rmse), mean square error(mse)




### 12.1.4 Analytics

#### Analyze Flights Delays

This workflow reads in a dataset. It then analyzes flights delay with sample datasets and prints the results.

#### Workflow

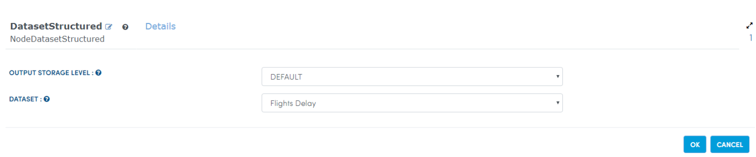
Below is the workflow. It does the following:

- Reads data from a sample dataset.
- Prints the sample datasets results.
- Column to be cast for new datatype double.
- Column to be cast for new datatype string.
- Updates the column name of datatype string.
- Prints the result of data updating after stringindexer Node.
- Executes the SQL queries with the given conditions.
- Prints the results.

#### Reading from Dataset

It reads Dataset files.

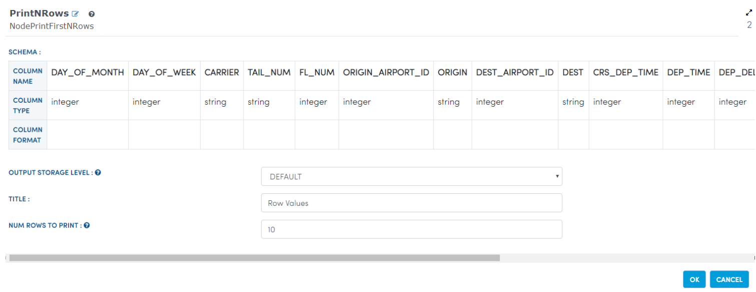
#### Processor Configuration



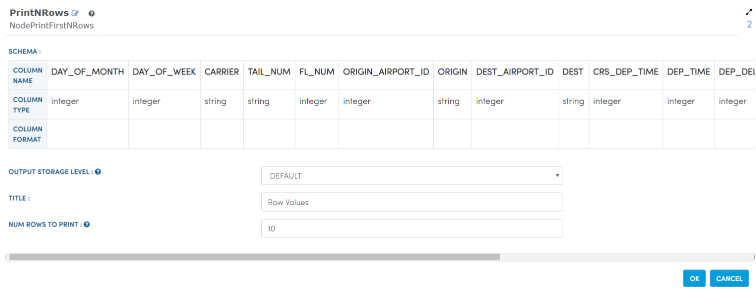

#### Processor Output

#### Print the sample datasets results

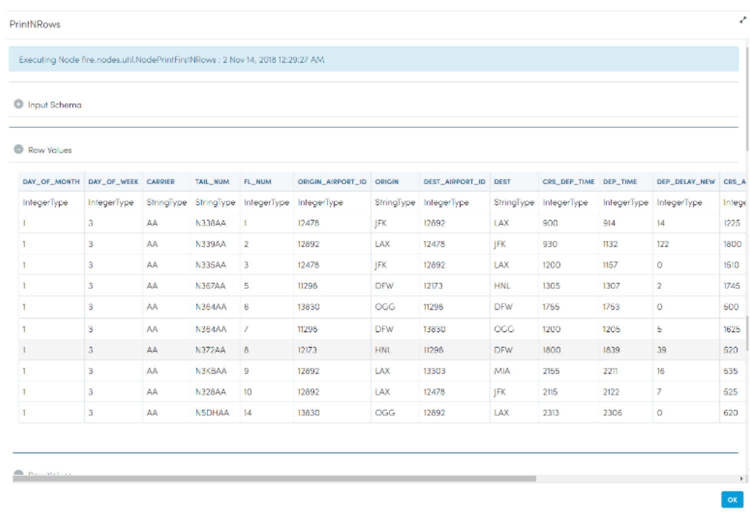
It prints the sample datasets results.




## Processor Configuration



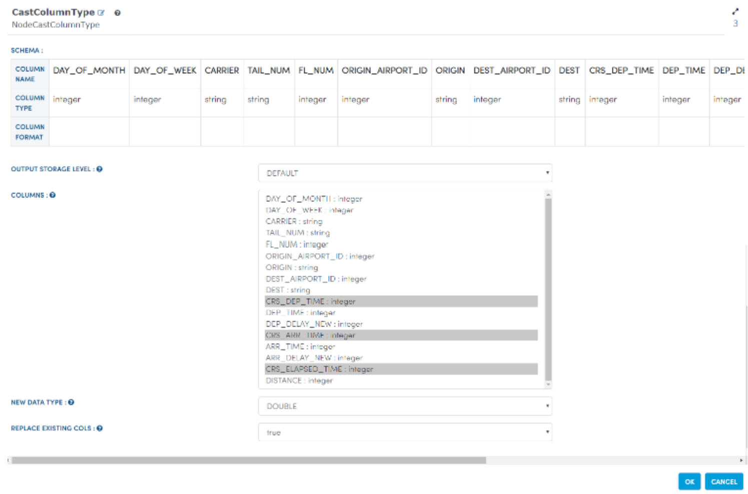

## Processor Output



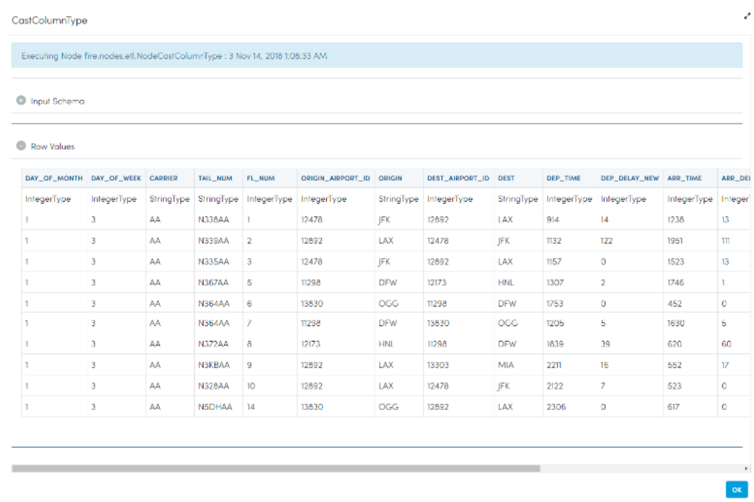

## Column to be cast for new datatype double

It casts for new datatype double using castcolumn type Node.

## Processor Configuration




### Processor Output




## Column to be cast for new datatype string

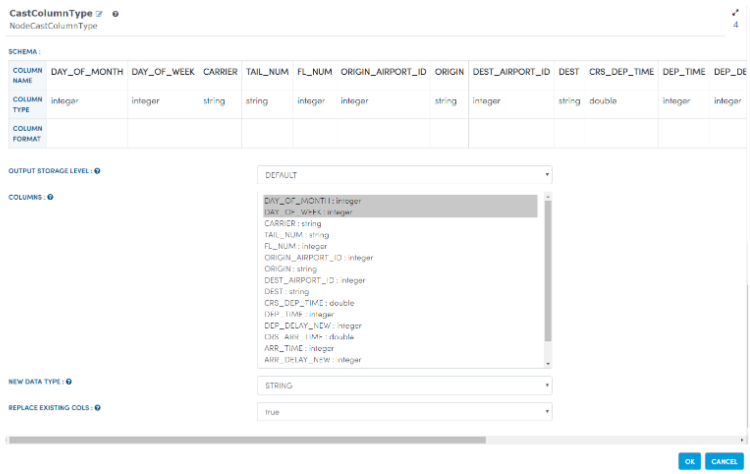
It casts for new datatype string using castcolumn type Node.

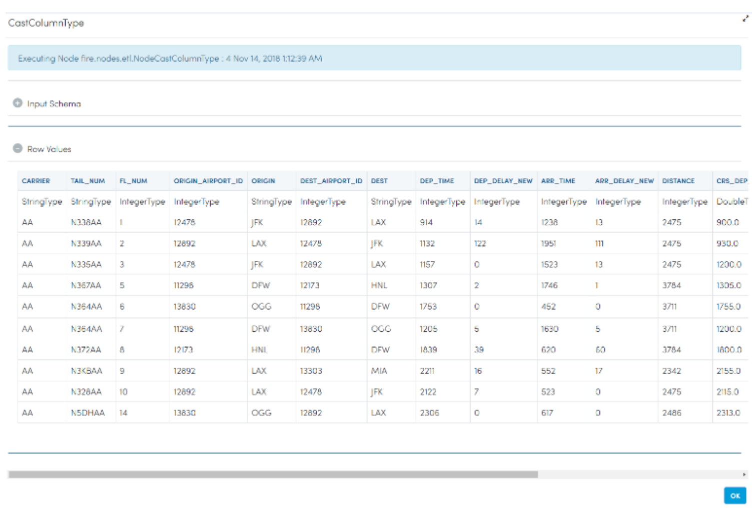
### Processor Configuration

### Processor Output

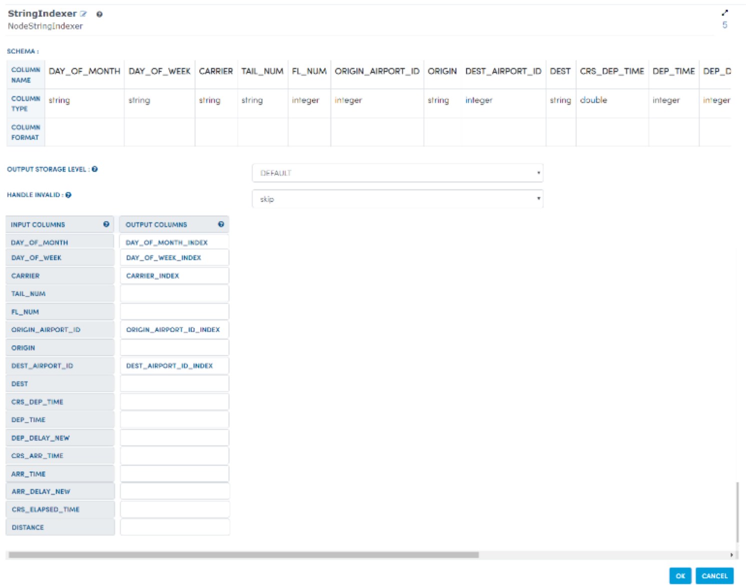
## Updates the column name of datatype string

It updates the column name of datatype string using stringindexer type Node.

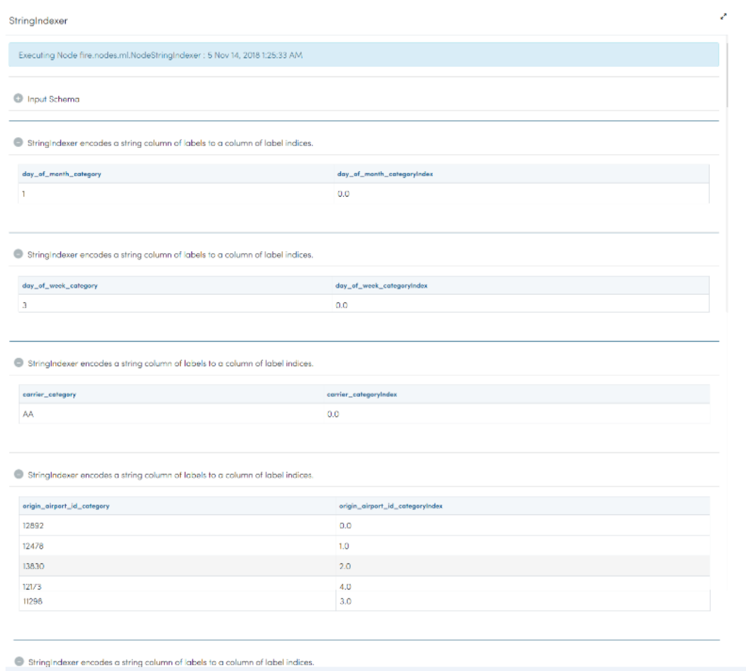

CastColumnType
NodeCastColumnType
SCHEMA :
COLUMN NAME
COLUMN TYPE
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
COLUMNS :
NEW DATA TYPE :
STRING
REPLACE EXISTING COLS :
true
OK
CANCEL


CastColumnType
Executing Node fire.nodes.etl.NodeCastColumnType : 4 Nov 14, 2018 1:12:39 AM
Input Schema
Row Values
OK

### Processor Configuration



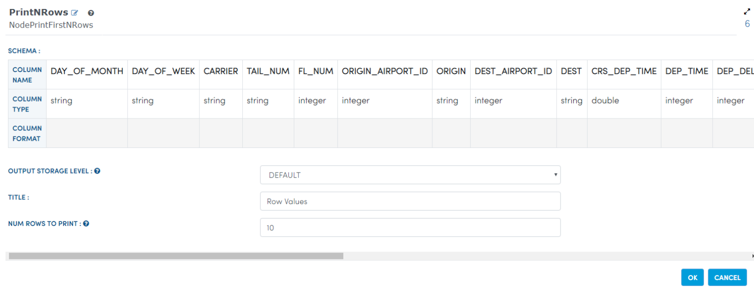
### Processor Output



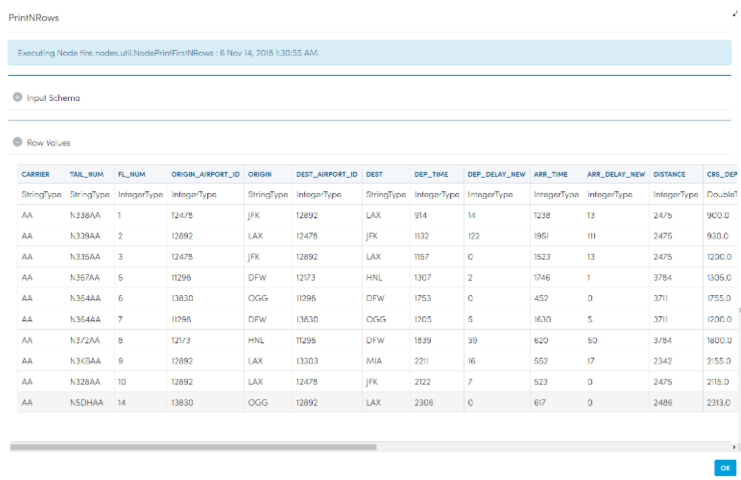
### Prints the Results

It prints the result of data updating after stringindexer Node.

### Processor Configuration




### Processor Output




## Executes the SQL queries

It executes the SQL queries with the given conditions.

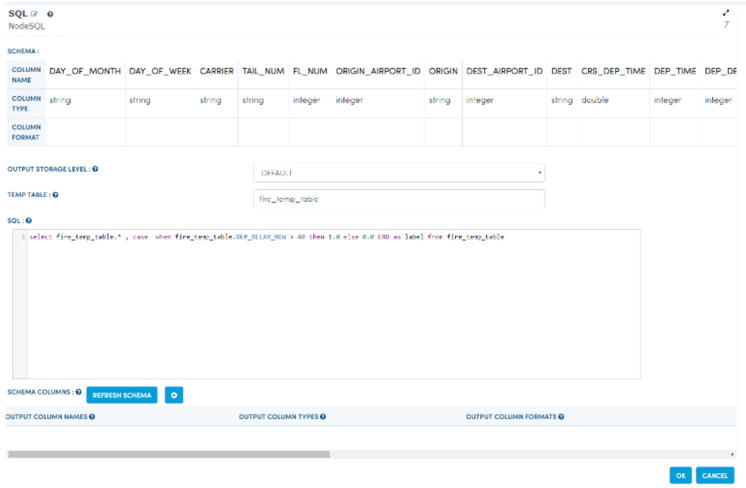
### Processor Configuration

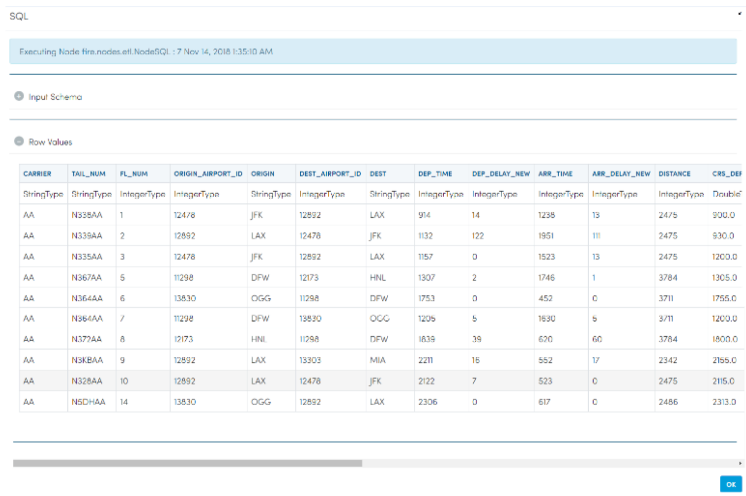
### Processor Output

## Prints the Results

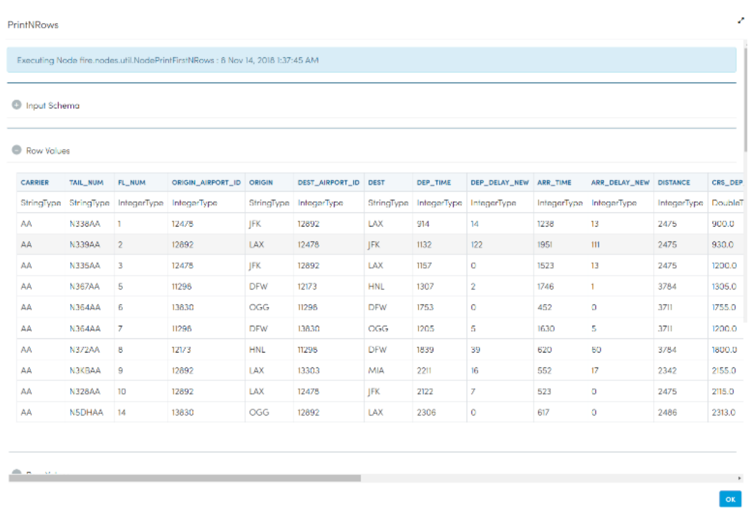
It prints the results after satisfied condition by sql queries.

### Processor Configuration


SQL
NodeSQL
OUTPUT STORAGE LEVEL :
TEMP TABLE :
SQL :
REFRESH SCHEMA
OK
CANCEL


SQL
Input Schema
Row Values
OK


PrintNRows
NodePrintFirstNRows
OUTPUT STORAGE LEVEL :
TITLE :
NUM ROWS TO PRINT :
OK
CANCEL


PrintNRows
Input Schema
Row Values
OK

#### Processor Output

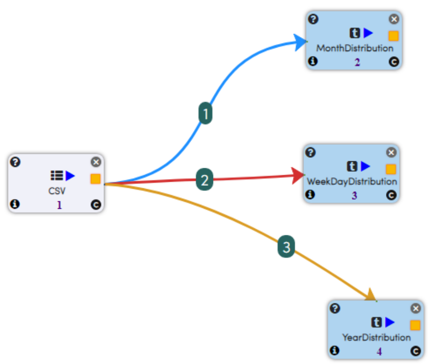
### Distribution Graphs

This workflow reads a CSV file. It then plots graphs of distribution of data by Week, Month & Year.

#### Workflow

Below is the workflow. It does the following:

- Reads a CSV file.
- Distribution of data by Week.
- Distribution of data by Month.
- Distribution of data by Year.




#### Reading CSV file

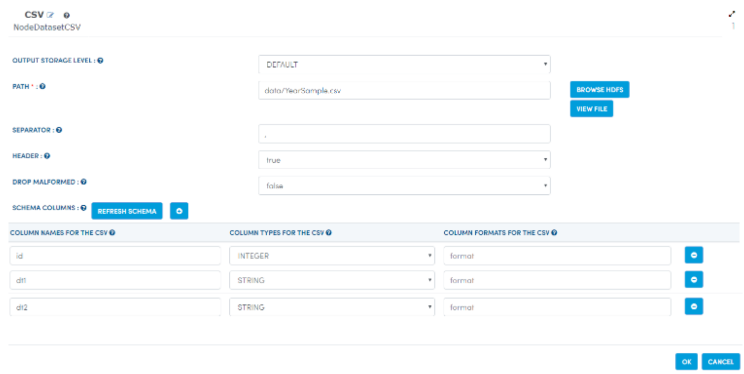
It reads CSV files.

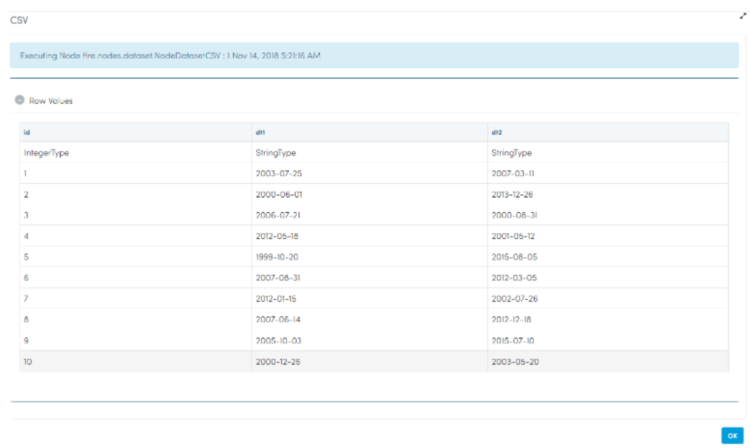
#### Processor Configuration

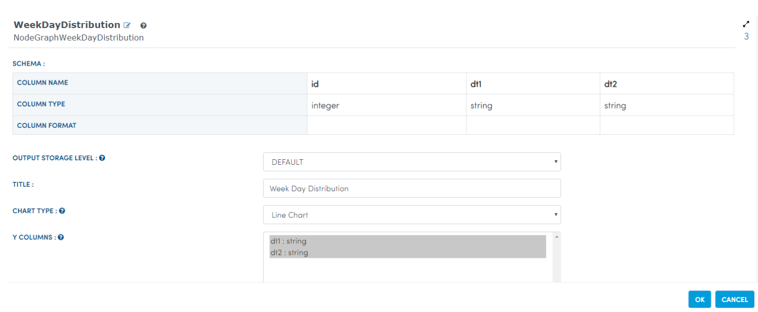
#### Processor Output

#### Distribution of data by Week

It plots graphs of distribution of data by Week using GraphWeekDistribution Node.

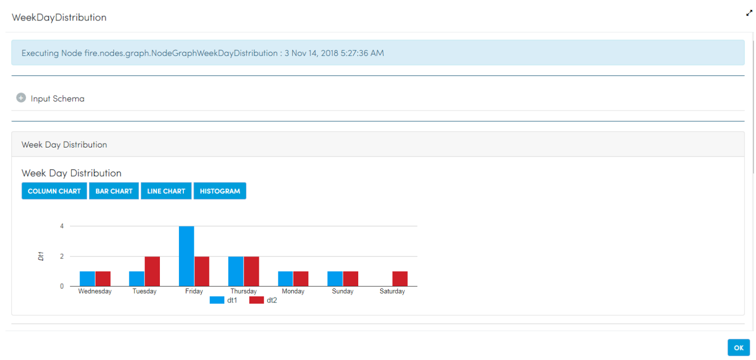

CSV
NodeDatasetCSV
OUTPUT STORAGE LEVEL :
DEFAULT
PATH * :
data/YearSample.csv
BROWSE HDFS
VIEW FILE
SEPARATOR :
HEADER :
true
DROP MALFORMED :
false
SCHEMA COLUMNS :
REFRESH SCHEMA
COLUMN NAMES FOR THE CSV
COLUMN TYPES FOR THE CSV
COLUMN FORMATS FOR THE CSV
id
INTEGER
format
dt1
STRING
format
dt2
STRING
format
OK
CANCEL


CSV
Executing Node fire.nodes.dataset.NodeDatasetCSV : 1 Nov 14, 2018 5:21:16 AM
Row Values
id
dt1
dt2
IntegerType
StringType
StringType
1
2003-07-25
2007-03-11
2
2000-06-01
2013-12-26
3
2006-07-21
2000-08-31
4
2012-05-18
2001-05-12
5
1999-10-20
2015-08-05
6
2007-08-31
2012-03-05
7
2012-01-15
2002-07-26
8
2007-06-14
2012-12-18
9
2005-10-03
2015-07-10
10
2000-12-26
2003-05-20
OK


WeekDayDistribution
NodeGraphWeekDayDistribution
SCHEMA :
COLUMN NAME
id
dt1
dt2
COLUMN TYPE
integer
string
string
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
TITLE :
Week Day Distribution
CHART TYPE :
Line Chart
Y COLUMNS :
dt1 : string
dt2 : string
OK
CANCEL

### Processor Configuration

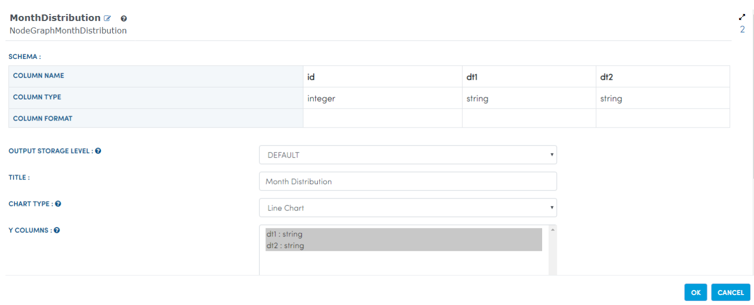
### Processor Output



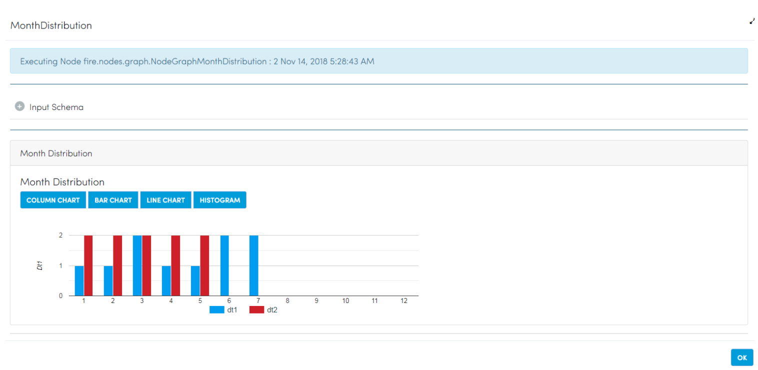

### Distribution of data by Month

It plots graphs of distribution of data by month using GraphMonthDistribution Node.

### Processor Configuration



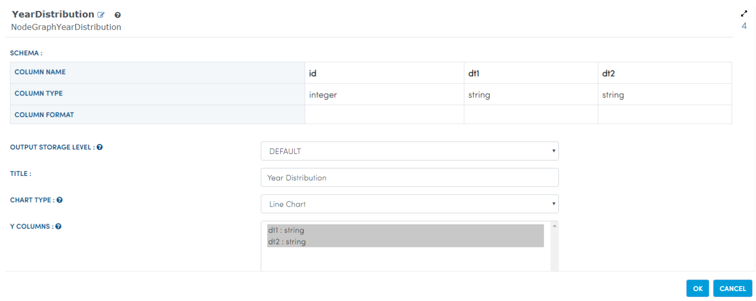

### Processor Output



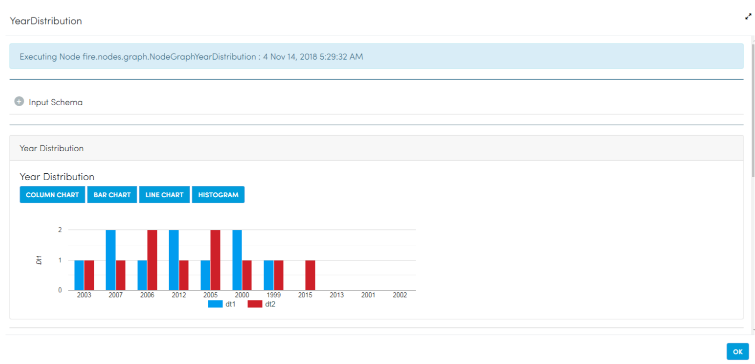
### Distribution of data by Year

It plots graphs of distribution of data by year using GraphYearDistribution Node.

### Processor Configuration




### Processor Output




### Farmers Markets On Geo Maps

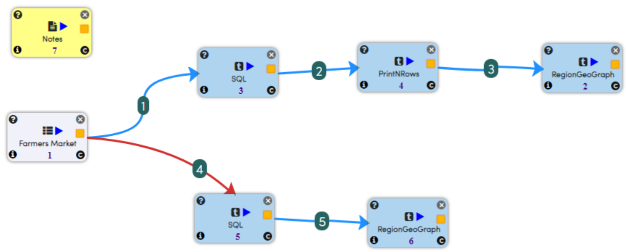
This workflow reads in a dataset. It then plots number of Farmers Market by City and by State on a Graph.

### Workflow

Below is the workflow. It does the following:

- Reads data from a sample dataset.
- Executes SQL Query for state count.
- Prints the result after executing query for state counts.
- Plots Graph for farmers with state counts.

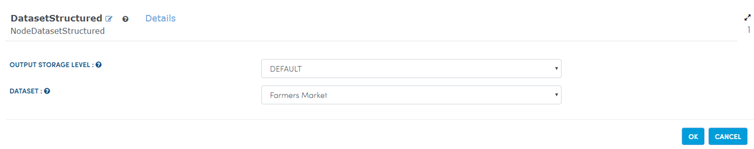
- Executes SQL Query for city counts.
- Plots Graph for farmers with city counts.



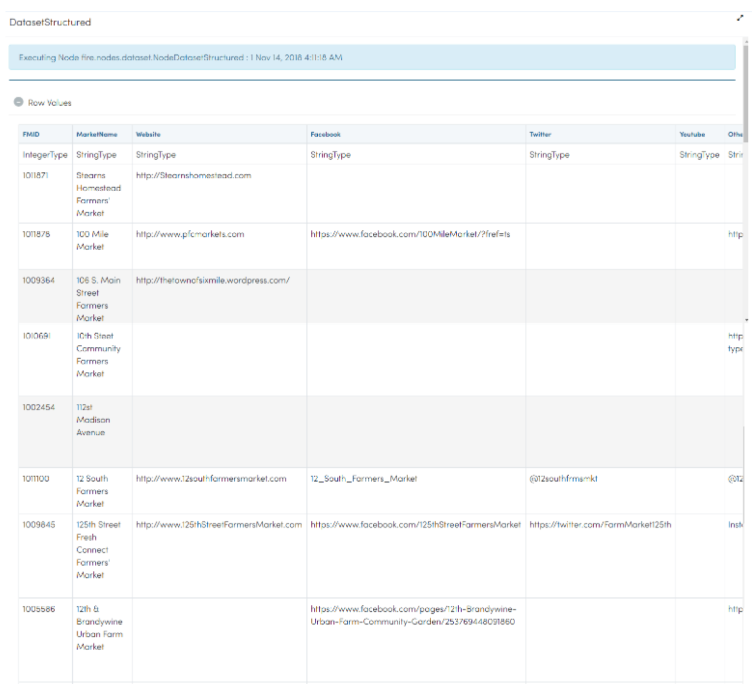

## Reading from Dataset

It reads sample Dataset files.

## Processor Configuration



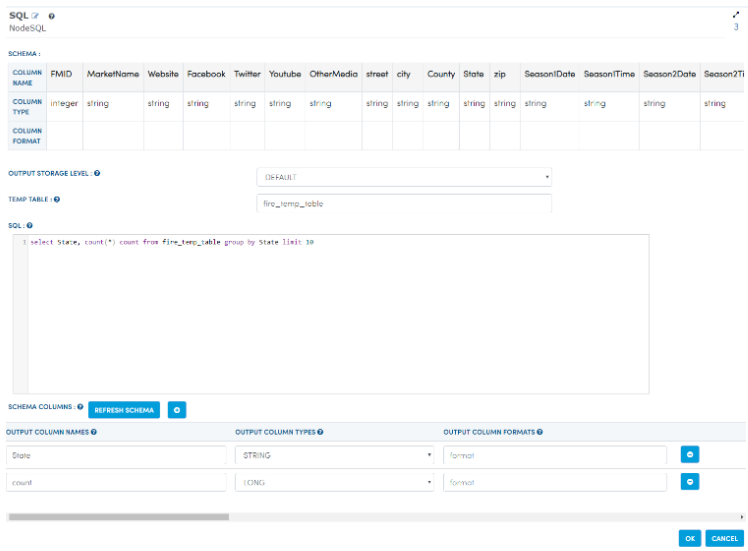

## Processor Output



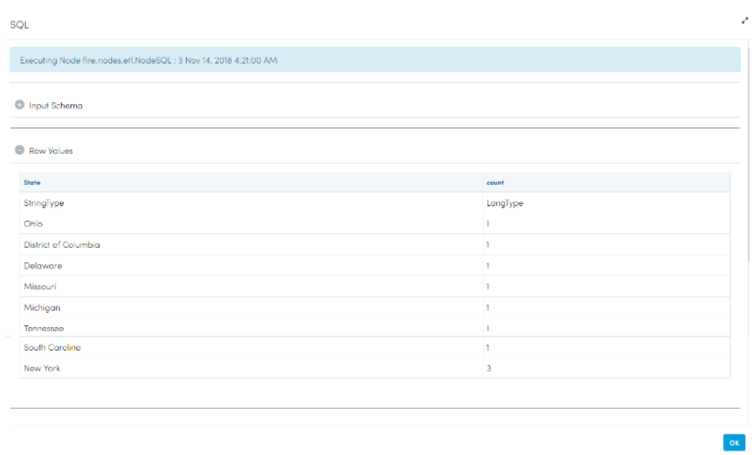
### Execute SQL Query

It Executes SQL Query for state count from the SQL node.

### Processor Configuration

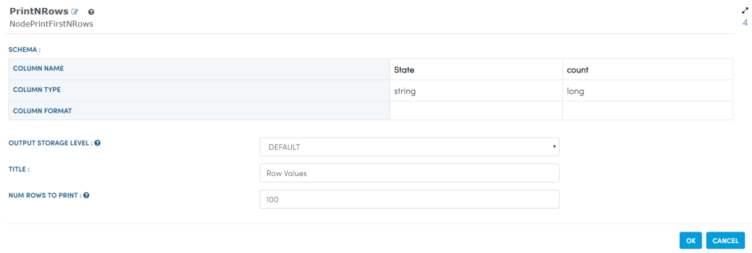


### Processor Output



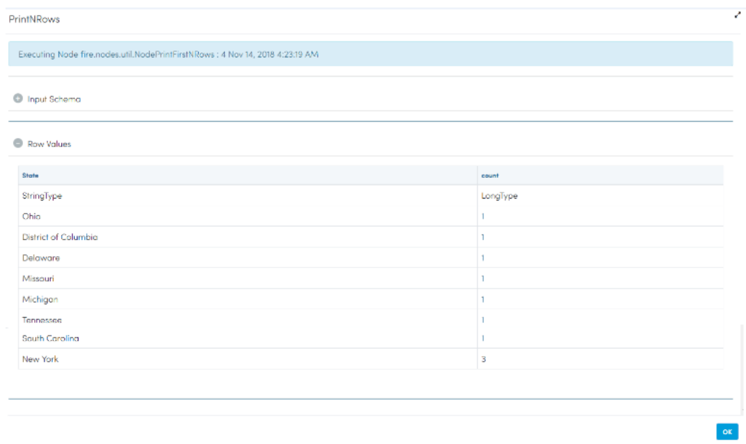
### Prints the Results

It prints the results after executing query for state counts by SQL Node.




### Processor Configuration

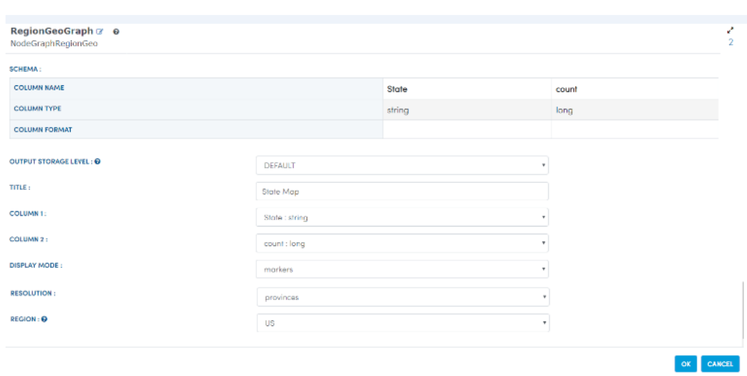
### Processor Output



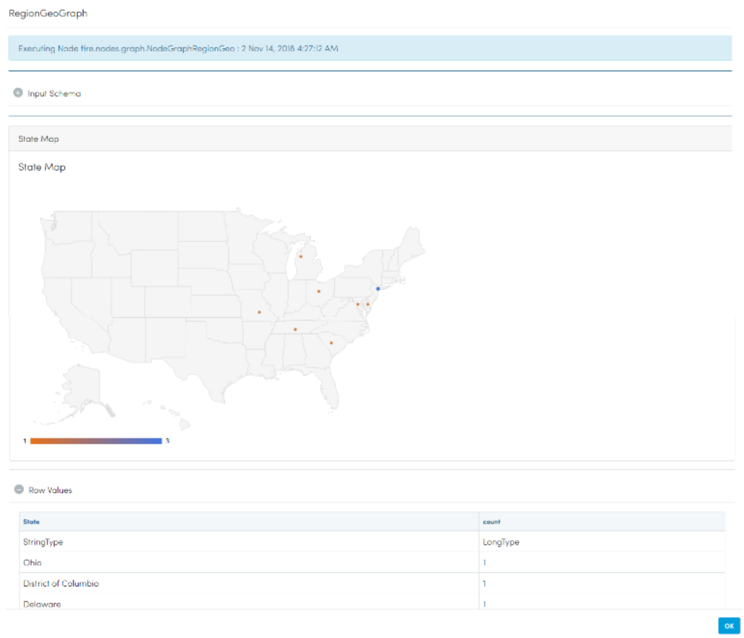

### Analyze using Graph

It plots Graph for farmers with state counts using RegionGeoGraph Processor.

### Processor Configuration



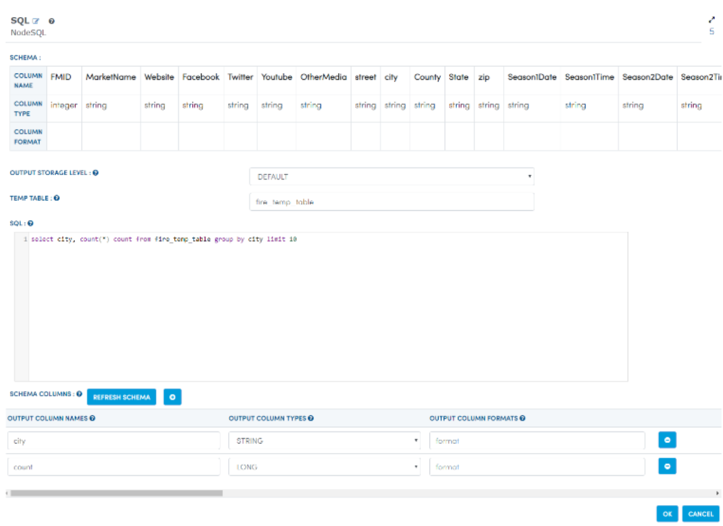
### Processor Output



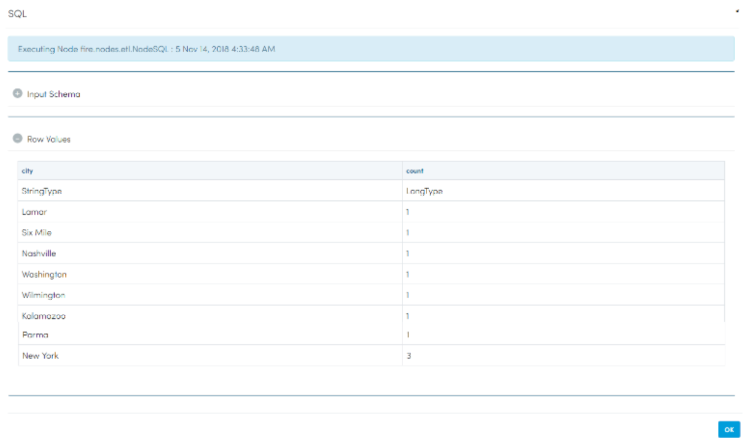

### Execute SQL Query

It executes SQL Query for City count from the SQL node.

### Processor Configuration



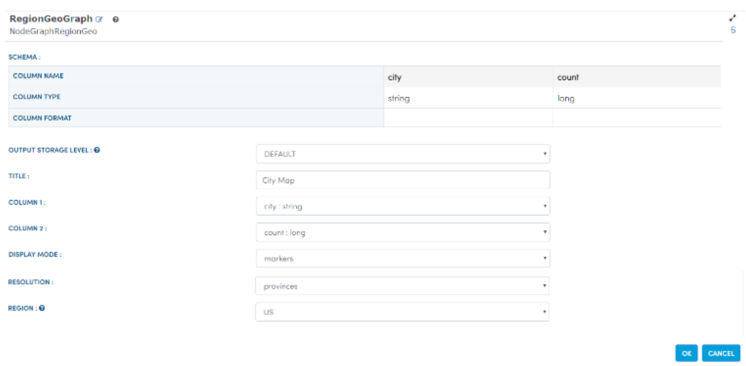

### Processor Output




### Analyze using Graph

It plots Graph for farmers with City counts using RegionGeoGRaph Node.

### Processor Configuration




### Processor Output

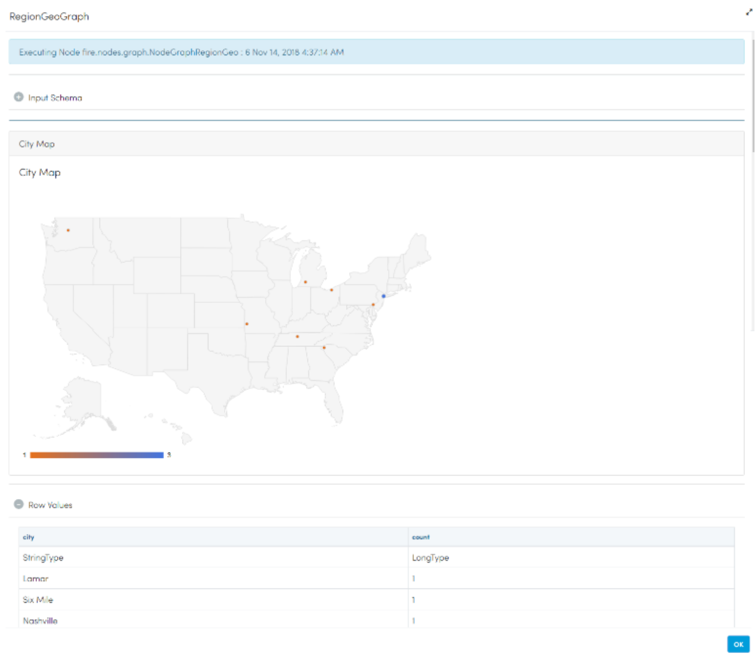
### General Payment Data Analysis

This workflow reads in a dataset. It then performs detailed analytics on general payment dataset.

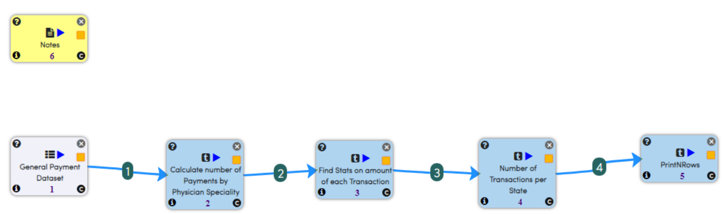
### Workflow

Below is the workflow. It does the following:

- Reads data from a sample dataset.
- Calculates count transactions by speciality.



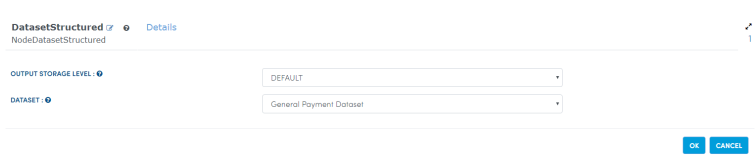

- Summary of transactions.
- Number of transactions per state.
- Prints the results.



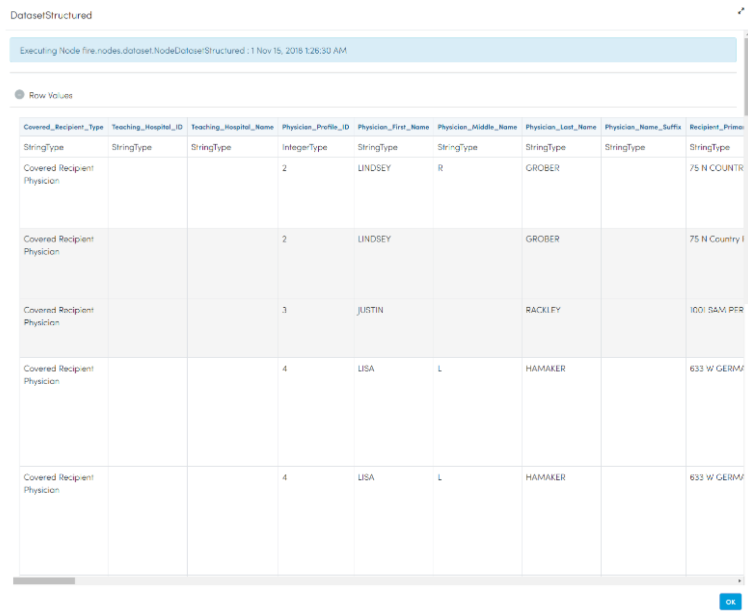

### Reading from Dataset

It reads from sample Dataset file.

### Processor Configuration



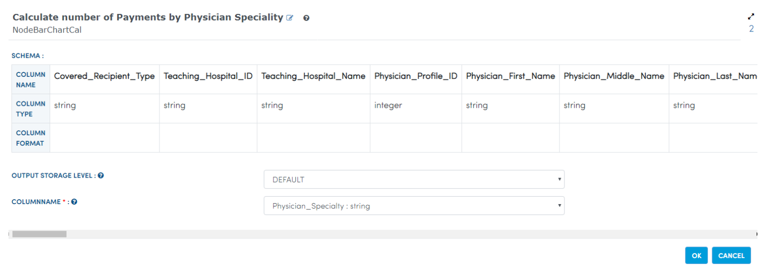

### Processor Output



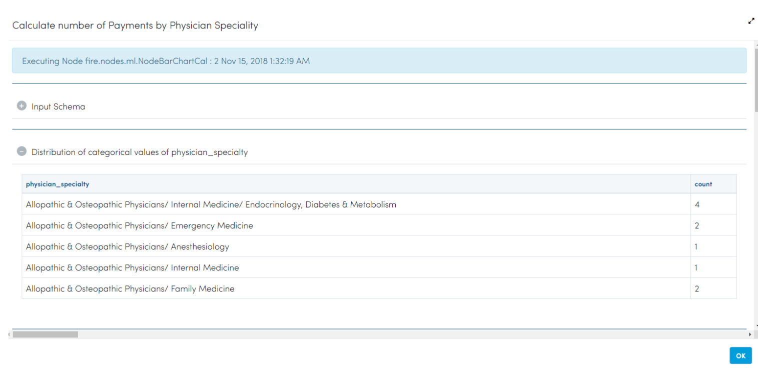
### Calculate count transactions by speciality

It will calculate count transactions by speciality using BarChartCal Node.

### Processor Configuration



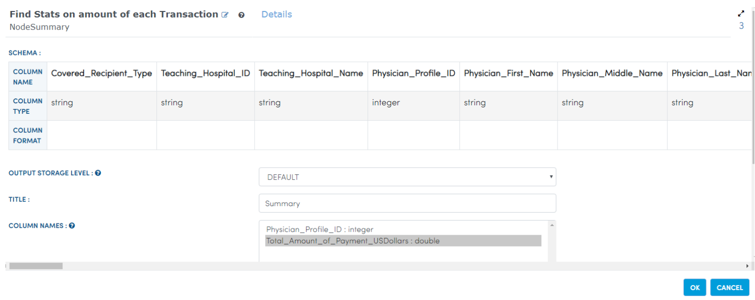
### Processor Output



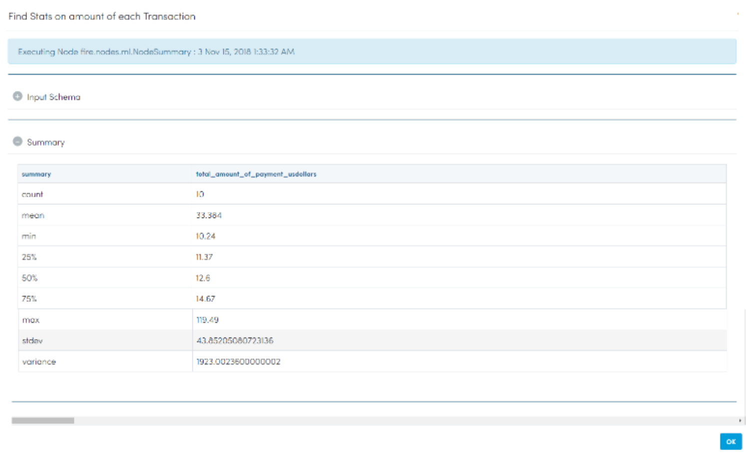
### Summary of transactions

It finds stats on amount of each transaction using Summary Node.

#### Processor Configuration




#### Processor Output




### Number of transaction per state

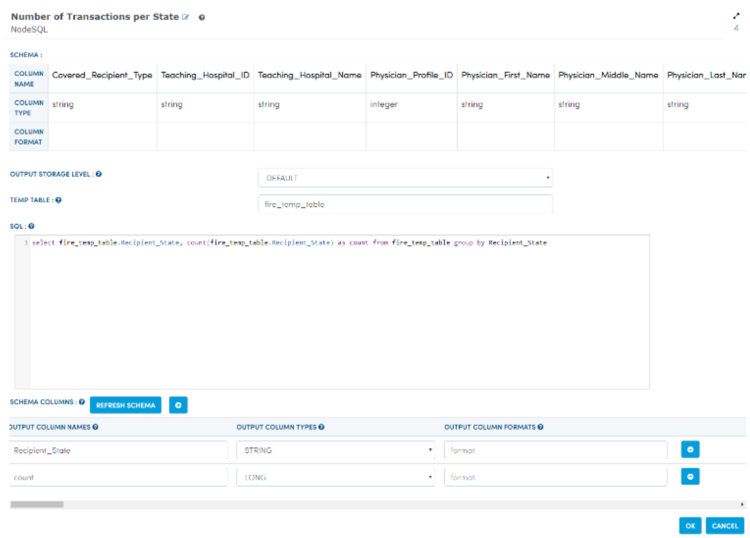
It finds number of transactions per state using SQL Node.

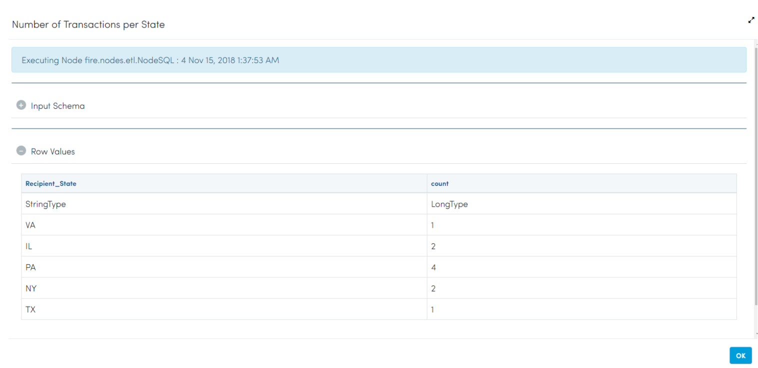
#### Processor Configuration

#### Processor Output

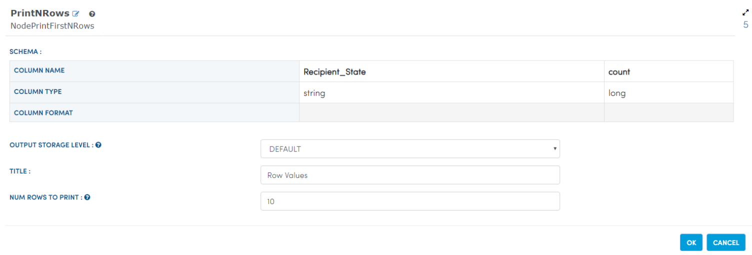
### Prints the results

It will print the result of output getting from SQL Node.

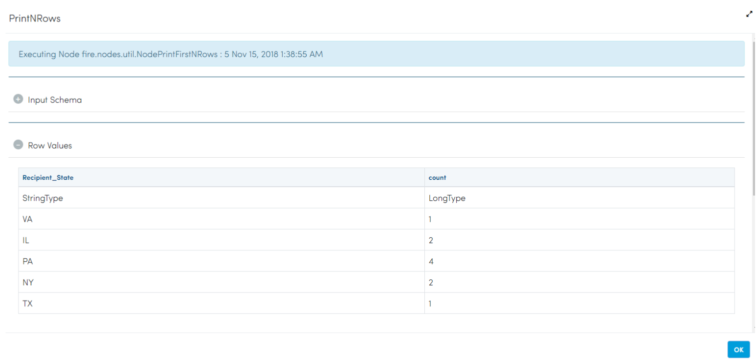




### Processor Configuration




### Processor Output




## Jetrail Data Analysis

This workflow reads in a dataset. It then calculates the monthly trend in JetRail Dataset and annalyses using graph.

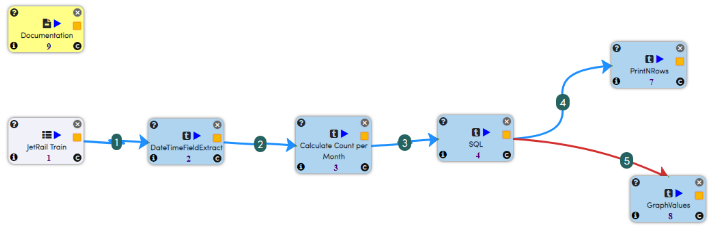
### Workflow

Below is the workflow. It does the following:

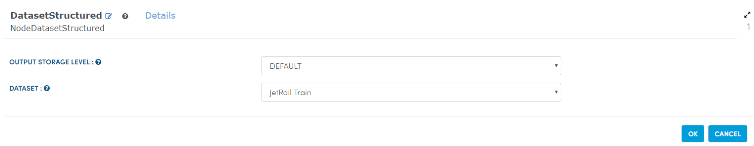
- Reads data from a sample dataset.
- Extracts date time field.
- Calculates count per month.
- Executes query for months.
- Print the results.
- Graphical analysis.

### Reading from Dataset

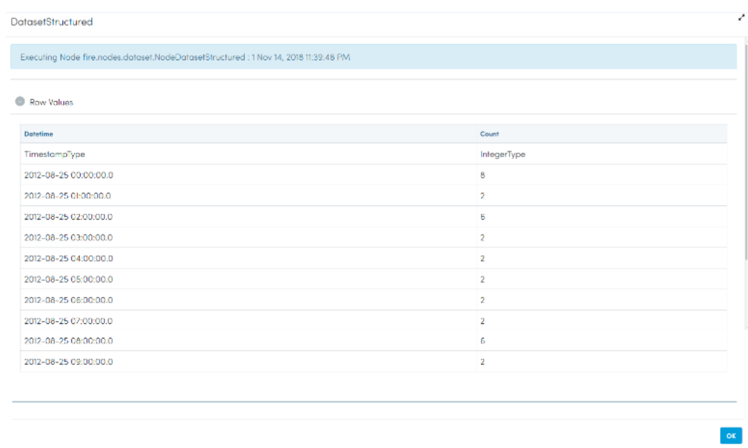
It reads from sample Dataset file.




### Processor Configuration




### Processor Output



DatasetStructured

Executing Node fire.nodes.dataset.NodeDatasetStructured : 1 Nov 14, 2018 11:39:48 PM

Row Values

| Datetime | Count |
|---|---|
| TimestampType | IntegerType |
| 2012-08-25 00:00:00.0 | 8 |
| 2012-08-25 01:00:00.0 | 2 |
| 2012-08-25 02:00:00.0 | 6 |
| 2012-08-25 03:00:00.0 | 2 |
| 2012-08-25 04:00:00.0 | 2 |
| 2012-08-25 05:00:00.0 | 2 |
| 2012-08-25 06:00:00.0 | 2 |
| 2012-08-25 07:00:00.0 | 2 |
| 2012-08-25 08:00:00.0 | 6 |
| 2012-08-25 09:00:00.0 | 2 |

OK

### Extract date time field

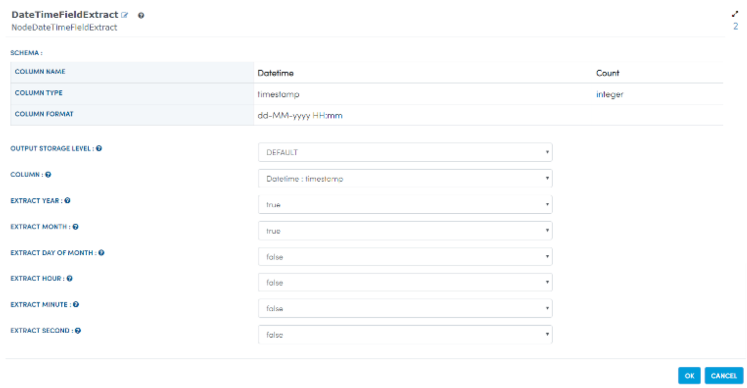
It extracts year and month field from date time field of timestamp using date time field extract Node.

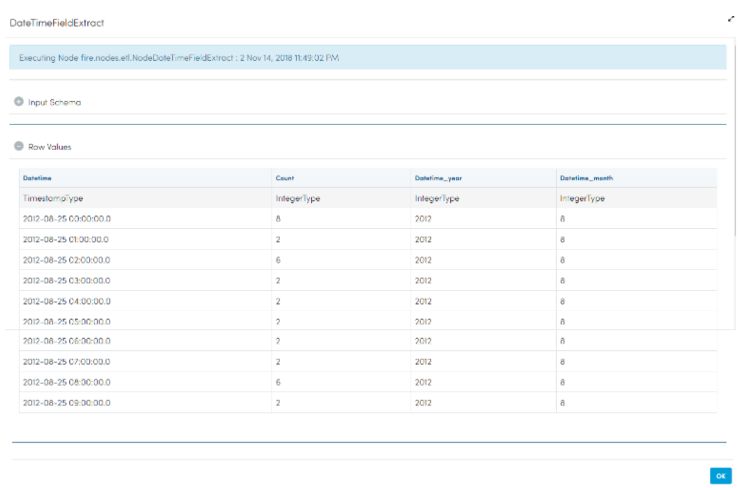
### Processor Configuration

### Processor Output

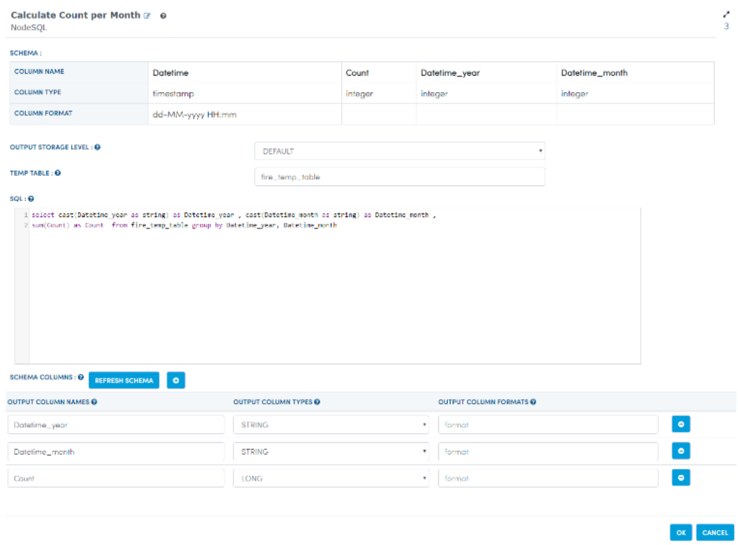
### Calculate count per month

It calculates count per month using query by SQL Node.

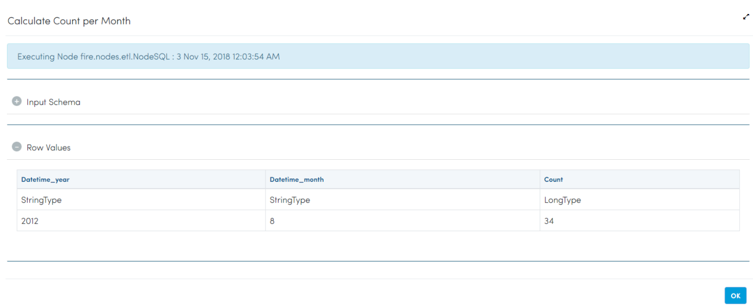

DateTimeFieldExtract
NodeDateTimeFieldExtract
SCHEMA :
COLUMN NAME
Datetime
Count
COLUMN TYPE
timestamp
integer
COLUMN FORMAT
dd-MM-yyyy HH:mm
OUTPUT STORAGE LEVEL :
DEFAULT
COLUMN :
Datetime : timestamp
EXTRACT YEAR :
true
EXTRACT MONTH :
true
EXTRACT DAY OF MONTH :
false
EXTRACT HOUR :
false
EXTRACT MINUTE :
false
EXTRACT SECOND :
false
OK
CANCEL


DateTimeFieldExtract
Input Schema
Row Values
Datetime
Count
Datetime_year
Datetime_month
TimestampType
IntegerType
IntegerType
IntegerType
OK

### Processor Configuration



### Processor Output



### Execute query for months

It executes query for grouping and selecting required fields, calculates sum of counts by SQL Node.

### Processor Configuration

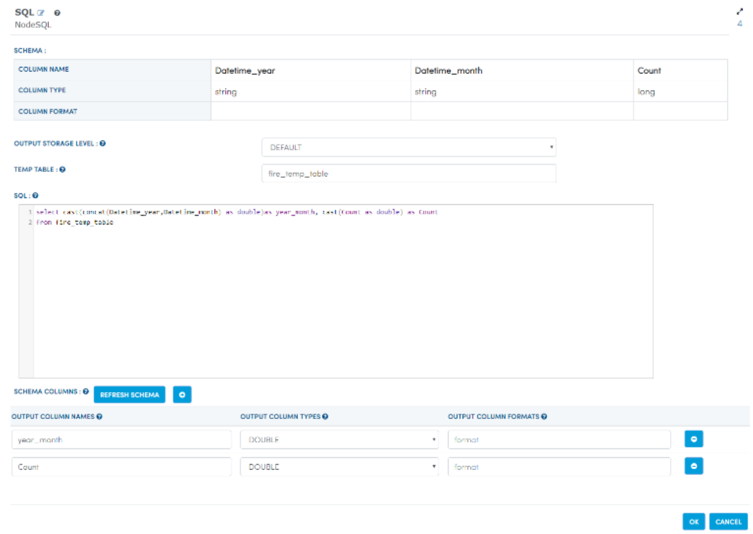
### Processor Output

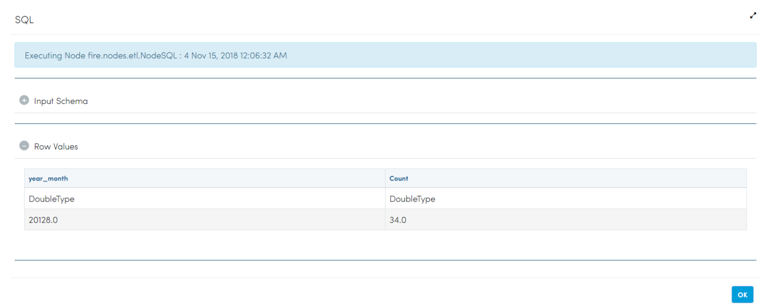
### Prints the Results

**It prints the results after executing SQL Query**

**align** center

**width** 60%




## Graphical analysis

It will graphically represent month with count using GraphValue Node.

## Processor Configuration

## Processor Output

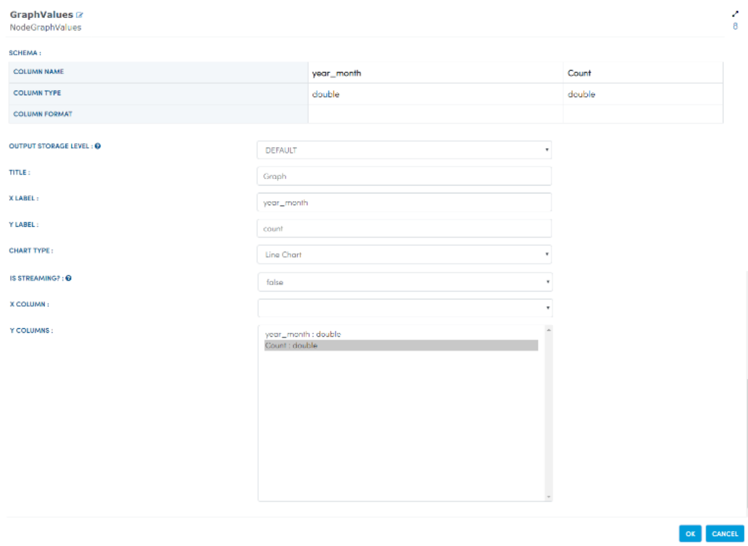
## NYC Taxidata Analysis

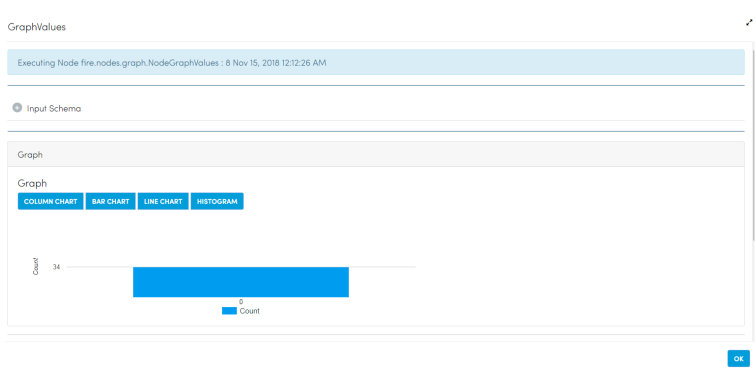
This workflow reads in a sample dataset. It then analyses average speed of taxis at each hour with sample data and prints the results.

## Workflow

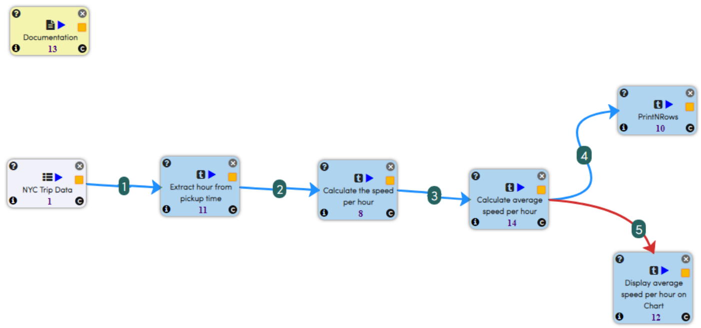
Below is the workflow. It does the following:

- Reads data from a dataset.
- Extracts hour from pickup time.
- Calculates the speed per hour.


GraphValues
NodeGraphValues
SCHEMA :
COLUMN NAME
year_month
Count
COLUMN TYPE
double
double
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
TITLE :
Graph
X LABEL :
year_month
Y LABEL :
count
CHART TYPE :
Line Chart
IS STREAMING? :
false
X COLUMN :
Y COLUMNS :
year_month : double
Count : double
OK
CANCEL


GraphValues
Executing Node fire.nodes.graph.NodeGraphValues : 8 Nov 15, 2018 12:12:26 AM
Input Schema
Graph
Graph
COLUMN CHART
BAR CHART
LINE CHART
HISTOGRAM
Count
34
0
Count
OK

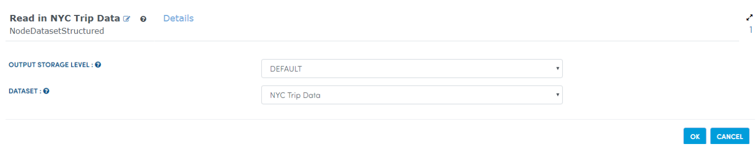
- Calculates the average speed per hour.
- Prints the results.
- Displays average speed per hour on chart.



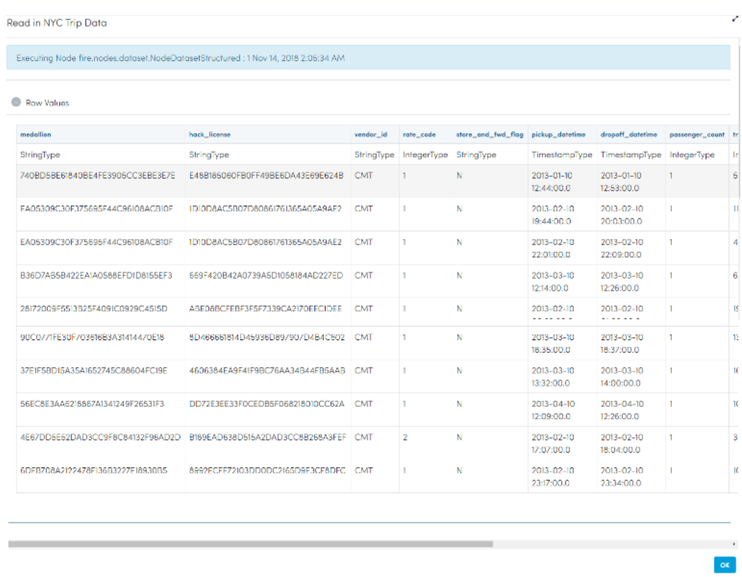

## Reading from Dataset

It reads sample Dataset files.

## Processor Configuration




## Processor Output


Read in NYC Trip Data

Executing Node fire.nodes.dataset.NodeDatasetStructured : 1 Nov 14, 2018 2:05:34 AM

Row Values

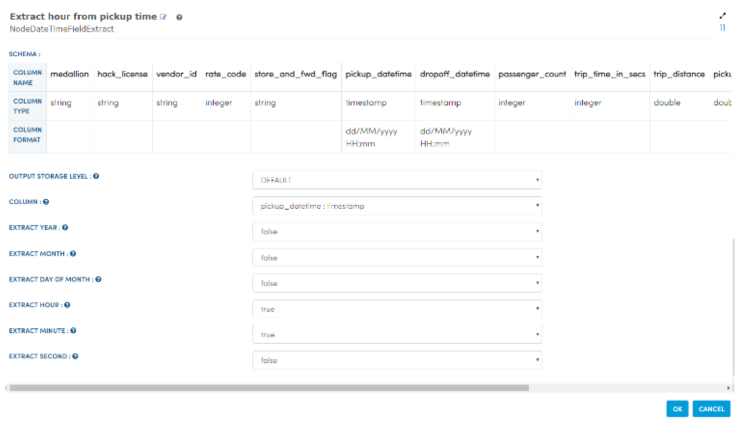
| medallion | hack_license | vendor_id | rate_code | store_and_fwd_flag | pickup_datetime | dropoff_datetime | passenger_count |
|---|---|---|---|---|---|---|---|
| StringType | StringType | StringType | IntegerType | StringType | TimestampType | TimestampType | IntegerType |
| 740BD5BE61840BE4FE3905CC3EBE3E7E | E48B185060FB0FF49BE6DA43E69E624B | CMT | 1 | N | 2013-01-10 12:44:00.0 | 2013-01-10 12:53:00.0 | 1 |
| EA05309C30F375695F44C96108ACB10F | 1D10D8AC5B07D80861761365A05A9AE2 | CMT | 1 | N | 2013-02-10 19:44:00.0 | 2013-02-10 20:03:00.0 | 1 |
| EA05309C30F375695F44C96108ACB10F | 1D10D8AC5B07D80861761365A05A9AE2 | CMT | 1 | N | 2013-02-10 22:01:00.0 | 2013-02-10 22:09:00.0 | 1 |
| B36D7AB5B422EA1A0588EFD1D8155EF3 | 669F420B42A0739A5D1058184AD227ED | CMT | 1 | N | 2013-03-10 12:14:00.0 | 2013-03-10 12:26:00.0 | 1 |
| 28172009F5513B25F4091C0929C4515D | ABE08BCFEBF3F5F7339CA2170EEC1DEE | CMT | 1 | N | 2013-02-10 | 2013-02-10 | 1 |
| 90C0771FE30F703616B3A31414470E18 | 8D466661814D45936D89790 7D4B4C602 | CMT | 1 | N | 2013-03-10 18:35:00.0 | 2013-03-10 18:37:00.0 | 1 |
| 37E1F5BD15A35A1652745C88604FC19E | 4606384EA9F41F9BC76AA34B44FB5AAB | CMT | 1 | N | 2013-03-10 13:32:00.0 | 2013-03-10 14:00:00.0 | 1 |
| 56EC8E3AA6218867A1341249F26531F3 | DD72E3EE33F0CEDB5F068218010CC62A | CMT | 1 | N | 2013-04-10 12:09:00.0 | 2013-04-10 12:26:00.0 | 1 |
| 4E67DD5E52DAD3CC9F8C84132F96AD2D | B189EAD638D515A2DAD3CC8B268A3FEF | CMT | 2 | N | 2013-02-10 17:07:00.0 | 2013-02-10 18:04:00.0 | 1 |
| 6DFB708A2122478F136B3227E18930B5 | 8992FCFF72103DD0DC2165D9E3CF8DFC | CMT | 1 | N | 2013-02-10 23:17:00.0 | 2013-02-10 23:34:00.0 | 1 |

OK

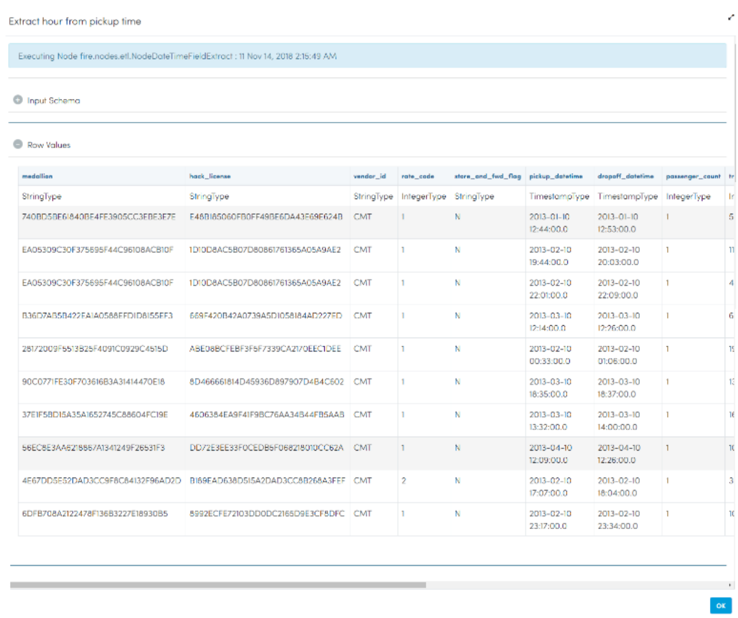
### Extract hour from pickup time

It extracts hour from pickup time using datetimefieldextract Node.

### Processor Configuration



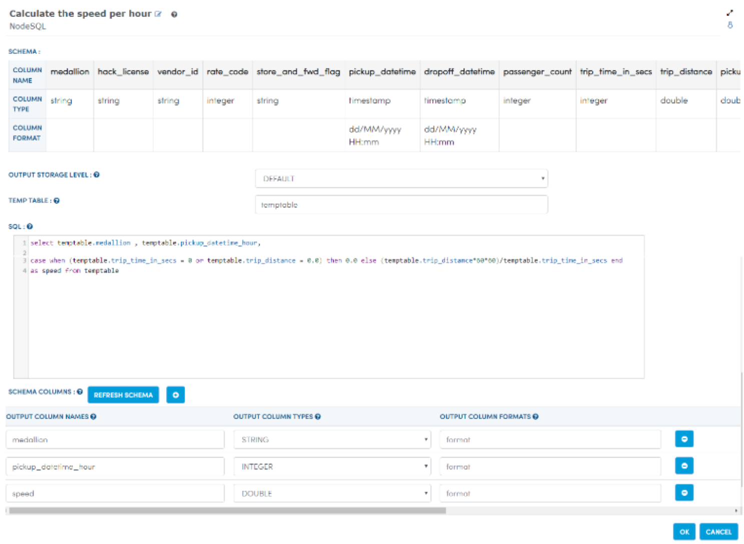

### Processor Output



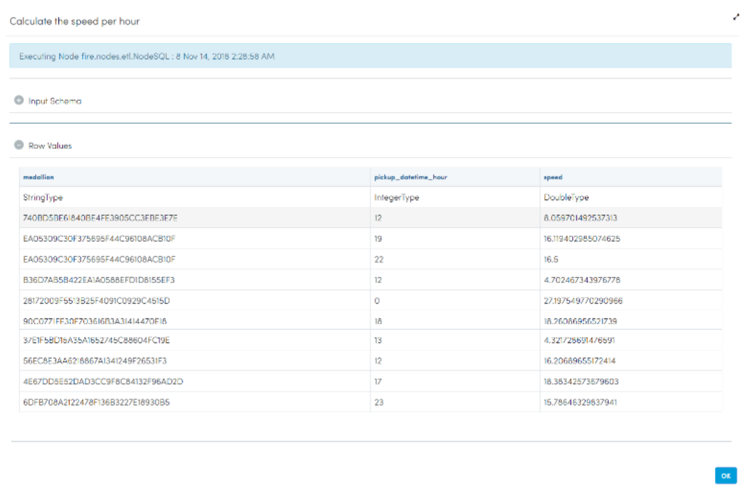

### Calculate the speed per hour

It calculates the speed per hour using SQL Node.

### Processor Configuration



### Processor Output

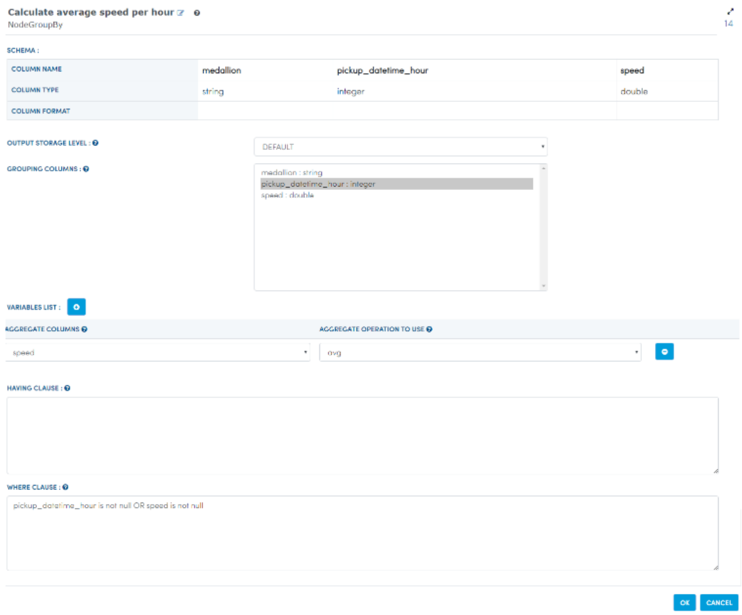


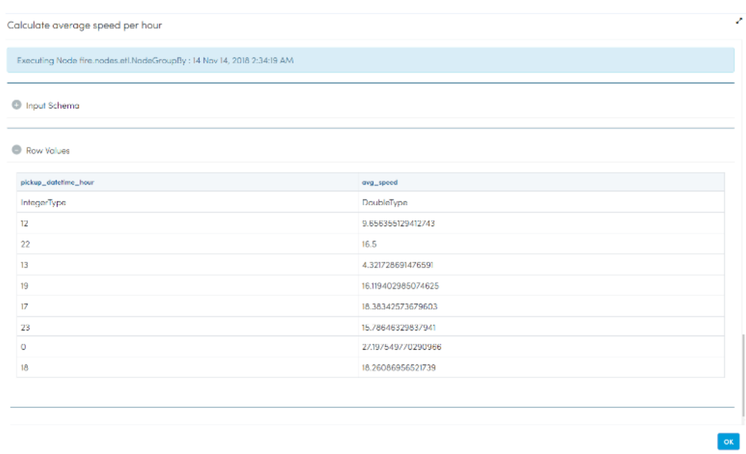
## Calculate the average speed per hour

It calculates the average speed per hour using GroupBy Node.

### Processor Configuration

### Processor Output

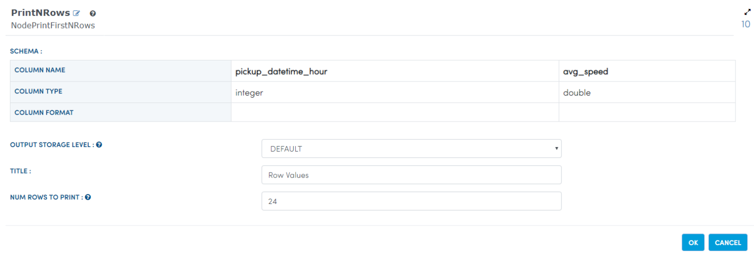

Calculate average speed per hour
NodeGroupBy
14
SCHEMA :
COLUMN NAME
medallion
pickup_datetime_hour
speed
COLUMN TYPE
string
integer
double
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
GROUPING COLUMNS :
medallion : string
pickup_datetime_hour : integer
speed : double
VARIABLES LIST :
AGGREGATE COLUMNS
AGGREGATE OPERATION TO USE
speed
avg
HAVING CLAUSE :
WHERE CLAUSE :
pickup_datetime_hour is not null OR speed is not null
OK
CANCEL


Calculate average speed per hour
Executing Node fire.nodes.etl.NodeGroupBy : 14 Nov 14, 2018 2:34:19 AM
Input Schema
Row Values
pickup_datetime_hour
avg_speed
IntegerType
DoubleType
12
9.656355129412743
22
16.5
13
4.321728691476591
19
16.119402985074625
17
18.38342573679603
23
15.78646329837941
0
27.197549770290966
18
18.26086956521739
OK

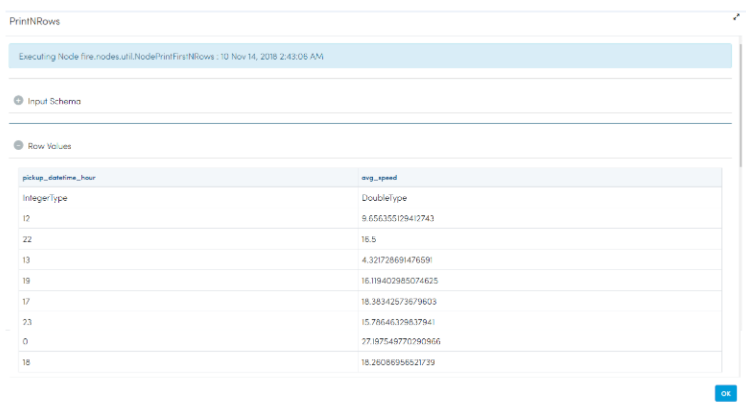
### Prints the results

It will print the result with the output of GroupBy Node.

#### Processor Configuration




#### Processor Output




### Analyze using Chart Graph

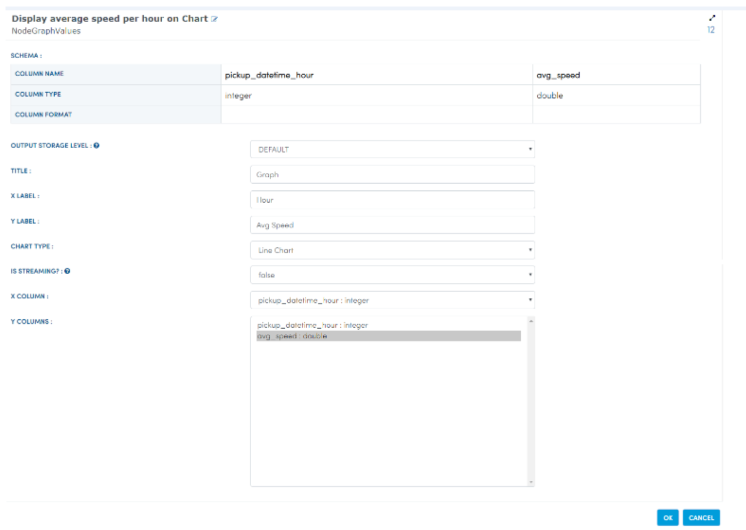
It displays average speed per hour on chart using Graphvalue Node.

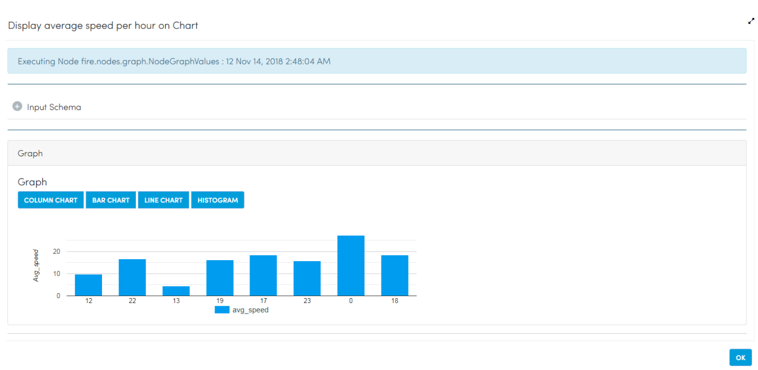
#### Processor Configuration

#### Processor Output

### Transaction Data Analytics

This workflow reads in a dataset. It then prints the results from the sample dataset and analyses using graphs.

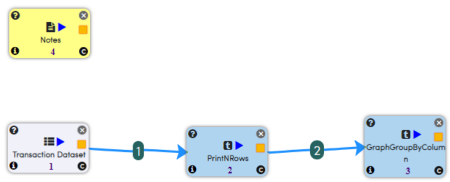

Display average speed per hour on Chart
NodeGraphValues
SCHEMA :
COLUMN NAME
pickup_datetime_hour
avg_speed
COLUMN TYPE
integer
double
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
TITLE :
Graph
X LABEL :
Hour
Y LABEL :
Avg Speed
CHART TYPE :
Line Chart
IS STREAMING? :
false
X COLUMN :
pickup_datetime_hour : integer
Y COLUMNS :
pickup_datetime_hour : integer
avg_speed : double
OK
CANCEL


Display average speed per hour on Chart
Executing Node fire.nodes.graph.NodeGraphValues : 12 Nov 14, 2018 2:48:04 AM
Input Schema
Graph
Graph
COLUMN CHART
BAR CHART
LINE CHART
HISTOGRAM
avg_speed
OK

## Workflow

Below is the workflow. It does the following:

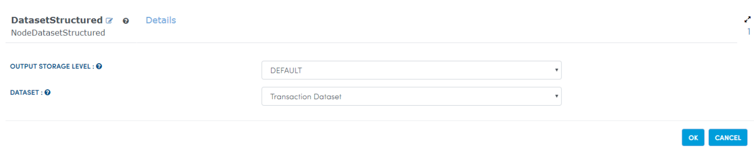
- Reads data from a sample dataset.
- It then prints the results from the sample dataset.
- Analysing using graphs.



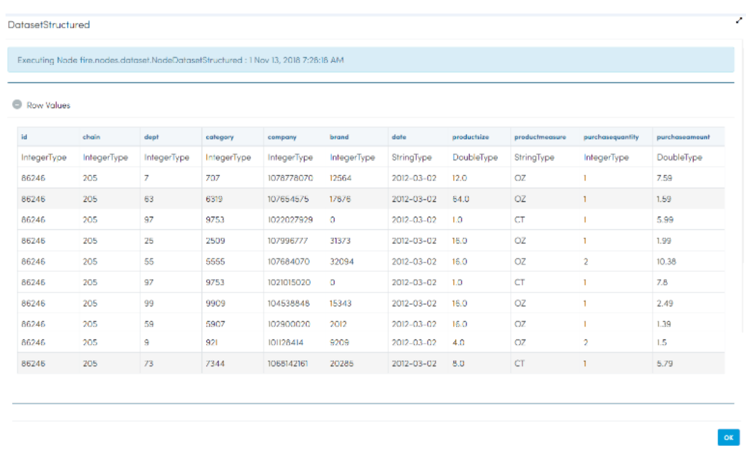

## Reading from Dataset

It reads Dataset File.

## Processor Configuration




## Processor Output

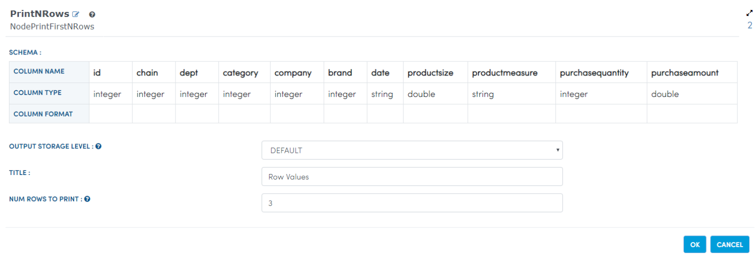



| id | chain | dept | category | company | brand | date | productsize | productmeasure | purchasequantity | purchaseamount |
|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | StringType | DoubleType | StringType | IntegerType | DoubleType |
| 86246 | 205 | 7 | 707 | 1078778070 | 12564 | 2012-03-02 | 12.0 | OZ | 1 | 7.59 |
| 86246 | 205 | 63 | 6319 | 107654575 | 17876 | 2012-03-02 | 64.0 | OZ | 1 | 1.59 |
| 86246 | 205 | 97 | 9753 | 1022027929 | 0 | 2012-03-02 | 1.0 | CT | 1 | 5.99 |
| 86246 | 205 | 25 | 2509 | 107996777 | 31373 | 2012-03-02 | 16.0 | OZ | 1 | 1.99 |
| 86246 | 205 | 55 | 5555 | 107684070 | 32094 | 2012-03-02 | 16.0 | OZ | 2 | 10.38 |
| 86246 | 205 | 97 | 9753 | 1021015020 | 0 | 2012-03-02 | 1.0 | CT | 1 | 7.8 |
| 86246 | 205 | 99 | 9909 | 104538848 | 15343 | 2012-03-02 | 16.0 | OZ | 1 | 2.49 |
| 86246 | 205 | 59 | 5907 | 102900020 | 2012 | 2012-03-02 | 16.0 | OZ | 1 | 1.39 |
| 86246 | 205 | 9 | 921 | 101128414 | 9209 | 2012-03-02 | 4.0 | OZ | 2 | 1.5 |
| 86246 | 205 | 73 | 7344 | 1068142161 | 20285 | 2012-03-02 | 8.0 | CT | 1 | 5.79 |

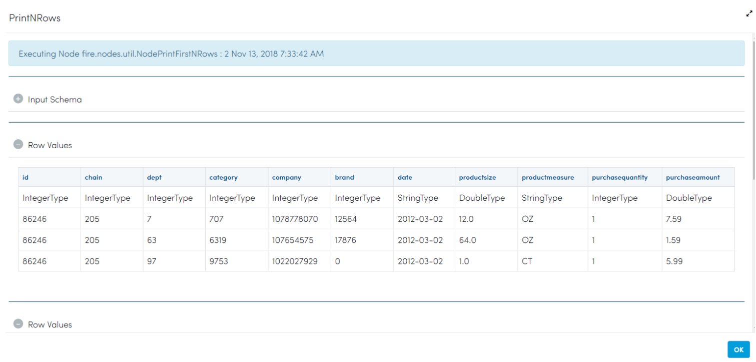
## Prints the sample Dataset Results

It prints sample Dataset Results.

#### Processor Configuration



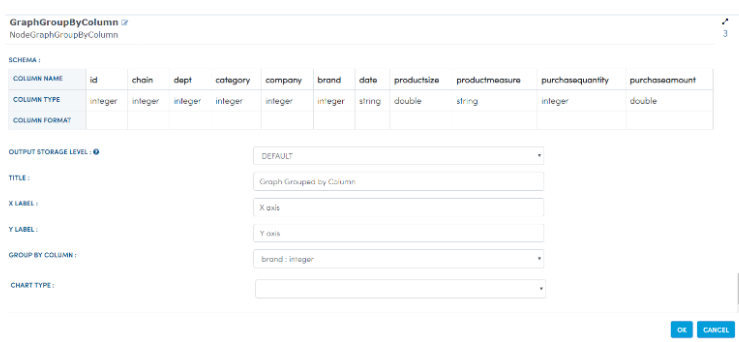

#### Processor Output




#### Analysing using Graph

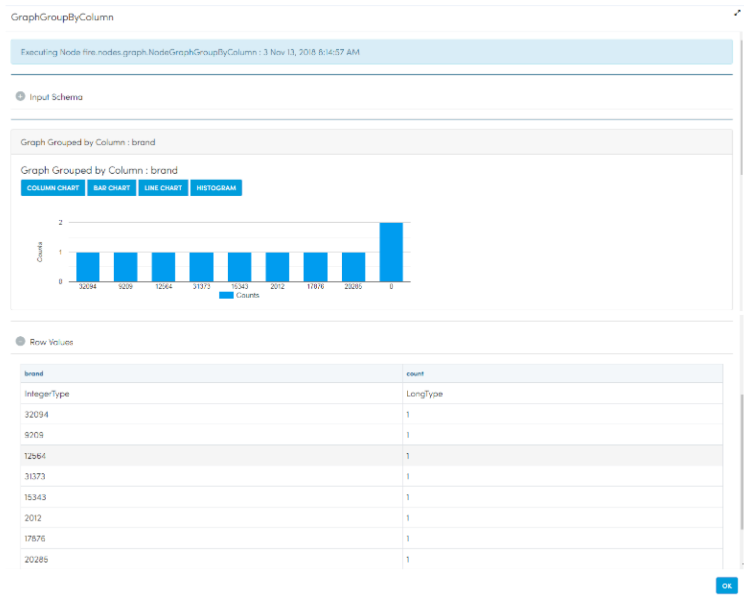
It helps to analyse using graph with Graph grouped by column brand and count.

#### Processor Configuration




#### Processor Output

### 12.1.5 Data Preparation



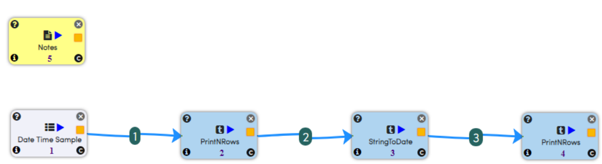

### Convert To Timestamps

This example converts to timestamp from the input sample dataset using string to date Node.

### Workflow

Below is the workflow. It does the following:

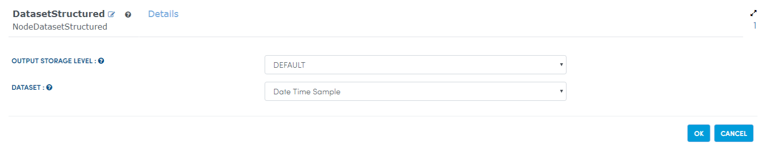
- Reads data from a sample dataset file.
- Prints sample dataset result.
- Converts sample string to timestamp.
- Prints the expected result.



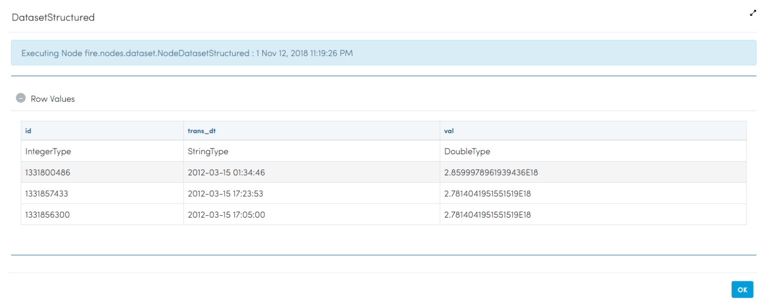

### Reading from Dataset

It reads sample Dataset File.

### Processor Configuration



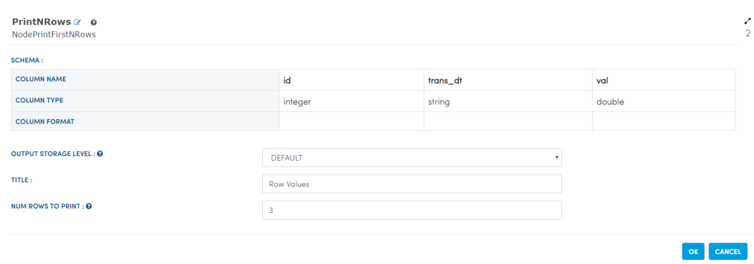

### Processor Output




### Prints the sample Dataset Results

It prints the results of the sample dataset available.

### Processor Configuration



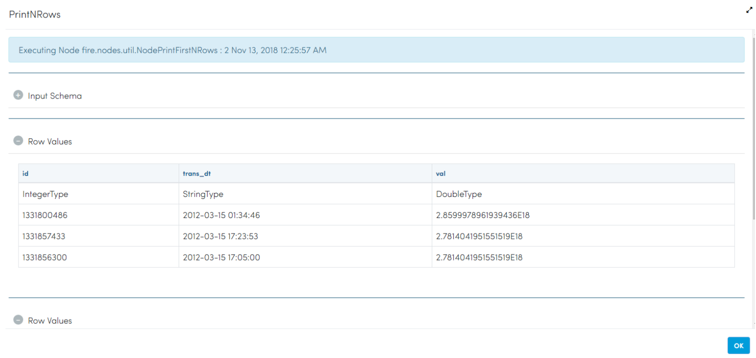

### Processor Output

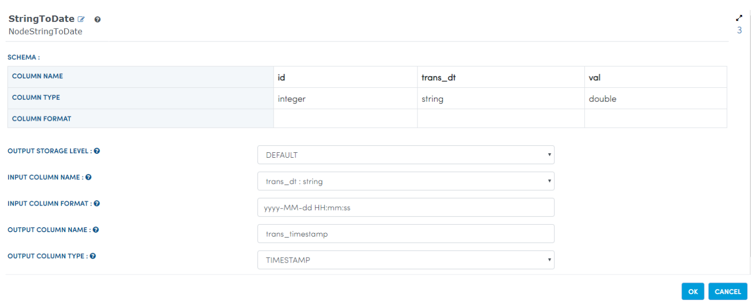
### Convert To Timestamps

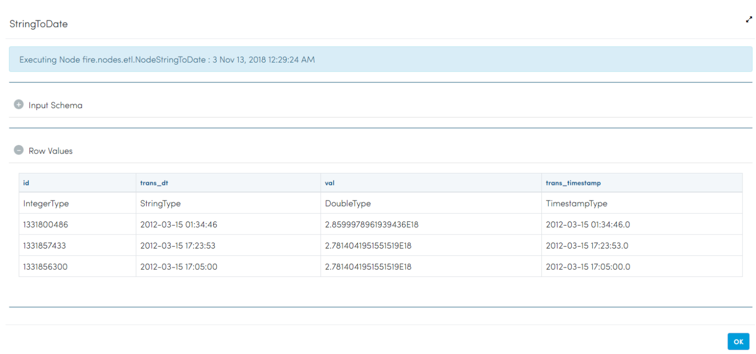
It converts To Timestamps using stringtodate Node.

### Processor Configuration

### Processor Output


PrintNRows
Executing Node fire.nodes.util.NodePrintFirstNRows : 2 Nov 13, 2018 12:25:57 AM
Input Schema
Row Values
id
trans_dt
val
IntegerType
StringType
DoubleType
1331800486
2012-03-15 01:34:46
2.8599978961939436E18
1331857433
2012-03-15 17:23:53
2.7814041951551519E18
1331856300
2012-03-15 17:05:00
2.7814041951551519E18
Row Values
OK

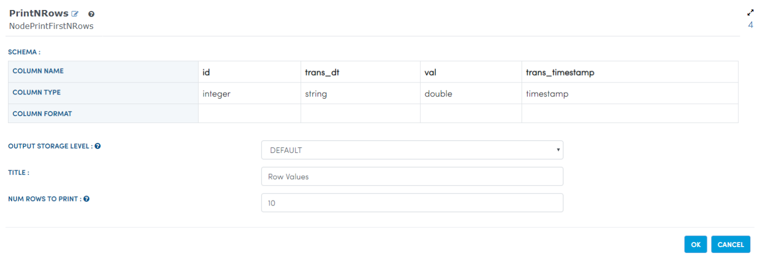

StringToDate
NodeStringToDate
SCHEMA :
COLUMN NAME
id
trans_dt
val
COLUMN TYPE
integer
string
double
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
INPUT COLUMN NAME :
trans_dt : string
INPUT COLUMN FORMAT :
yyyy-MM-dd HH:mm:ss
OUTPUT COLUMN NAME :
trans_timestamp
OUTPUT COLUMN TYPE :
TIMESTAMP
OK
CANCEL


StringToDate
Executing Node fire.nodes.etl.NodeStringToDate : 3 Nov 13, 2018 12:29:24 AM
Input Schema
Row Values
id
trans_dt
val
trans_timestamp
IntegerType
StringType
DoubleType
TimestampType
1331800486
2012-03-15 01:34:46
2.8599978961939436E18
2012-03-15 01:34:46.0
1331857433
2012-03-15 17:23:53
2.7814041951551519E18
2012-03-15 17:23:53.0
1331856300
2012-03-15 17:05:00
2.7814041951551519E18
2012-03-15 17:05:00.0
OK

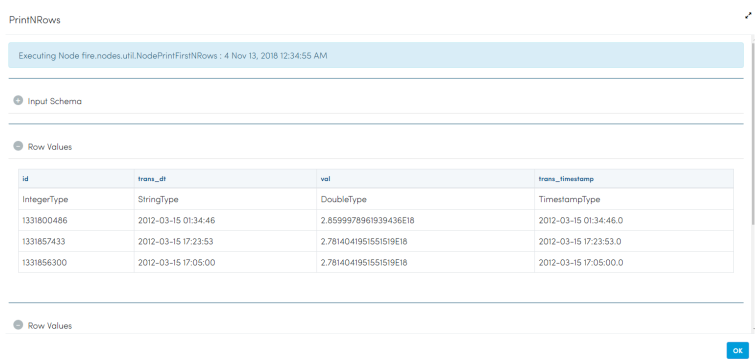
### Prints the Results

It prints the results after converting to Timestamps.

### Processor Configuration




### Processor Output




### Data Validation

This example performs different kinds of data validation on input dataset like valid/invalid email,valid/invalid date,null/not null check etc.

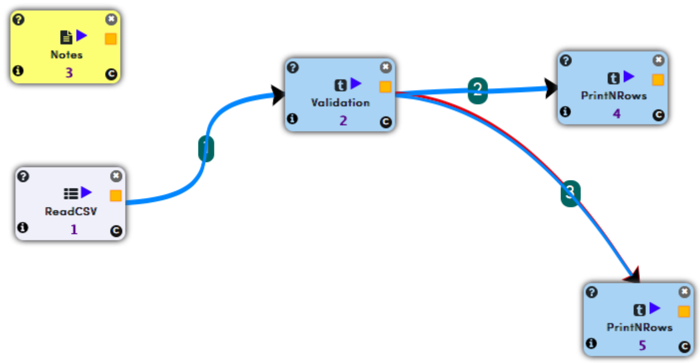
### Workflow

Below is the workflow. It does the following:

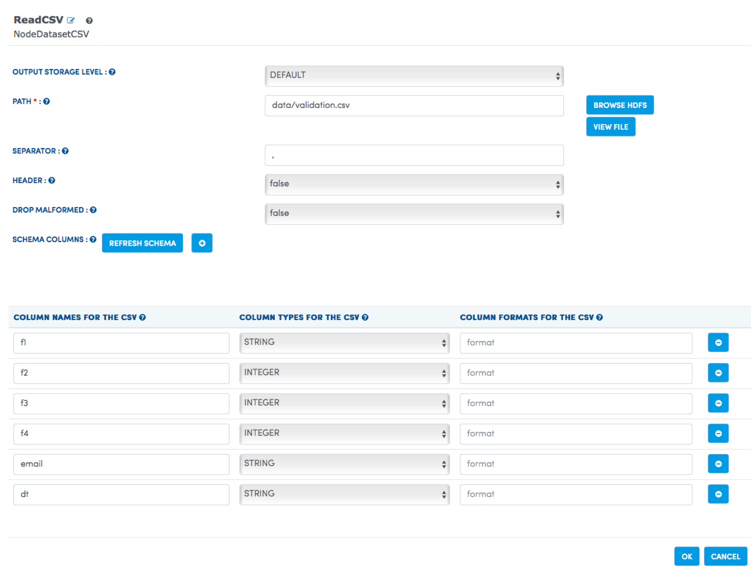
- Reads data from a CSV file.
- Performs specific validation on specific columns.

### Reading from CSV File

It reads data from a CSV file.




### Processor Configuration




### Processor Output

## Performing Validation

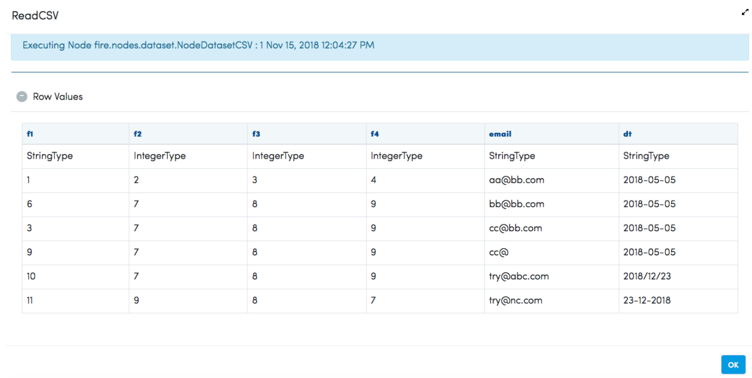
It performs different validation on different columns.

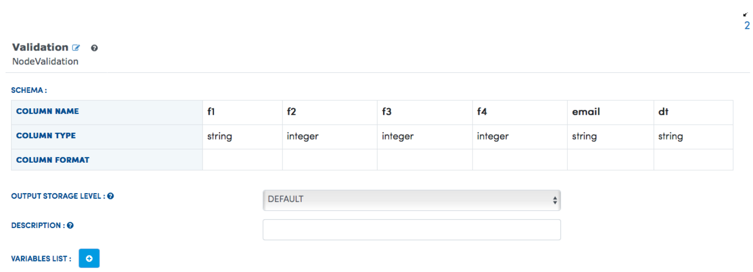
### Processor Configuration

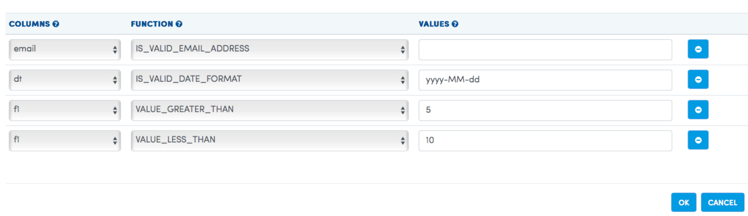
### Processor Output

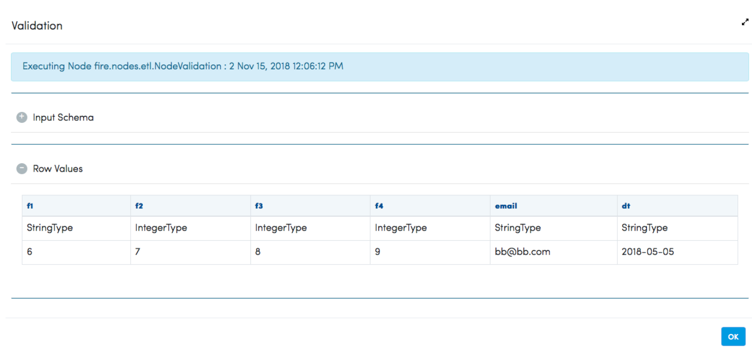
## Multi-Validation Workflow

This workflow performs multiple validations on each incoming record


ReadCSV
Executing Node fire.nodes.dataset.NodeDatasetCSV : 1 Nov 15, 2018 12:04:27 PM
Row Values
f1 f2 f3 f4 email dt
StringType IntegerType IntegerType IntegerType StringType StringType
1 2 3 4 aa@bb.com 2018-05-05
6 7 8 9 bb@bb.com 2018-05-05
3 7 8 9 cc@bb.com 2018-05-05
9 7 8 9 cc@ 2018-05-05
10 7 8 9 try@abc.com 2018/12/23
11 9 8 7 try@nc.com 23-12-2018
OK


2
Validation
NodeValidation
SCHEMA :
COLUMN NAME f1 f2 f3 f4 email dt
COLUMN TYPE string integer integer integer string string
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
DESCRIPTION :
VARIABLES LIST :


COLUMNS FUNCTION VALUES
email IS_VALID_EMAIL_ADDRESS
dt IS_VALID_DATE_FORMAT yyyy-MM-dd
f1 VALUE_GREATER_THAN 5
f1 VALUE_LESS_THAN 10
OK CANCEL


Validation
Executing Node fire.nodes.etl.NodeValidation : 2 Nov 15, 2018 12:06:12 PM
Input Schema
Row Values
f1 f2 f3 f4 email dt
StringType IntegerType IntegerType IntegerType StringType StringType
6 7 8 9 bb@bb.com 2018-05-05
OK

- Records which pass validation are output into the first edge
- Records which fail validation are output into the seconds edge

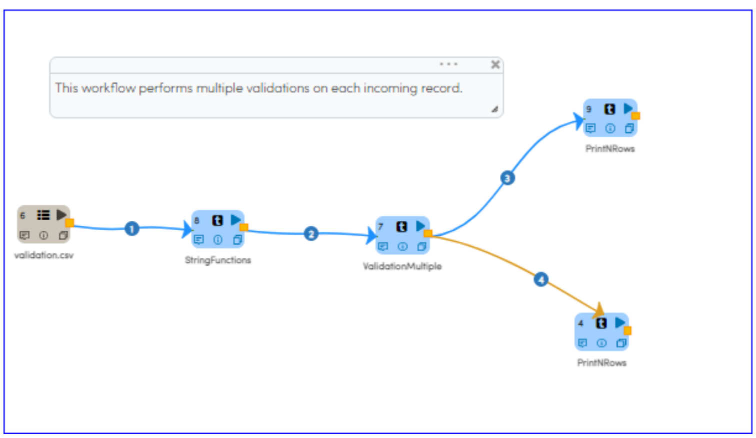
#### Validations

- Ensures that field is greater than or equal to specified string value
- Ensures that field is less than or equal to specified string value
- Ensures that field matches given datePattern
- Ensures that the email is valid
- Ensures field length is greater than or equal to specified length

#### Workflow

Below is the workflow. It does the following:

- Reads data from a CSV file.
- Performs specific validation on specific columns.

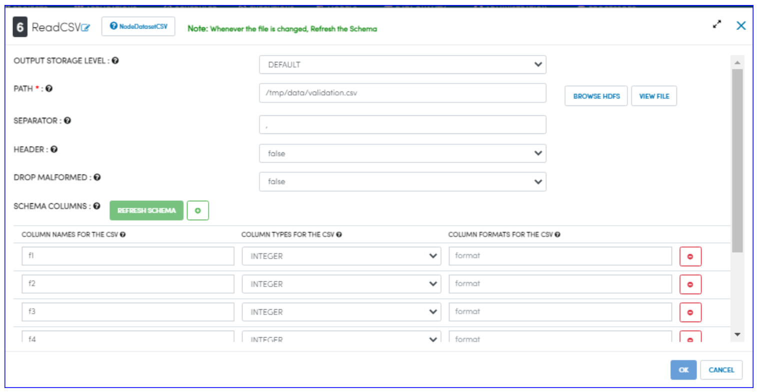


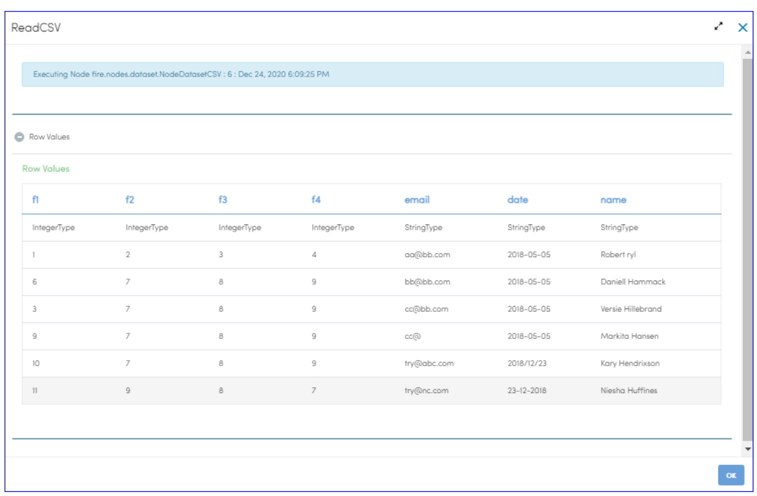

#### Reading from CSV File

`DatasetCSV` processor reads data from a CSV file.

#### Processor Configuration

#### Processor Output

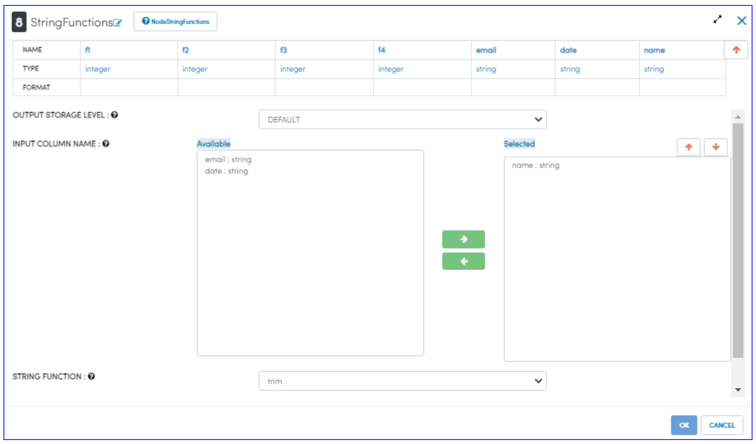

6 ReadCSV
NodeDatasetCSV
Note: Whenever the file is changed, Refresh the Schema
OUTPUT STORAGE LEVEL :
DEFAULT
PATH * :
/tmp/data/validation.csv
BROWSE HDFS
VIEW FILE
SEPARATOR :
HEADER :
false
DROP MALFORMED :
false
SCHEMA COLUMNS :
REFRESH SCHEMA
COLUMN NAMES FOR THE CSV
COLUMN TYPES FOR THE CSV
COLUMN FORMATS FOR THE CSV
f1
INTEGER
format
f2
INTEGER
format
f3
INTEGER
format
OK
CANCEL


ReadCSV
Executing Node fire.nodes.dataset.NodeDatasetCSV : 6 : Dec 24, 2020 6:09:25 PM
Row Values
Row Values
f1 f2 f3 f4 email date name
IntegerType IntegerType IntegerType IntegerType StringType StringType StringType
1 2 3 4 aa@bb.com 2018-05-05 Robert ryl
6 7 8 9 bb@bb.com 2018-05-05 Daniell Hammack
3 7 8 9 cc@bb.com 2018-05-05 Versie Hillebrand
9 7 8 9 cc@ 2018-05-05 Markita Hansen
10 7 8 9 try@abc.com 2018/12/23 Kary Hendrixson
11 9 8 7 try@nc.com 23-12-2018 Niesha Huffines
OK

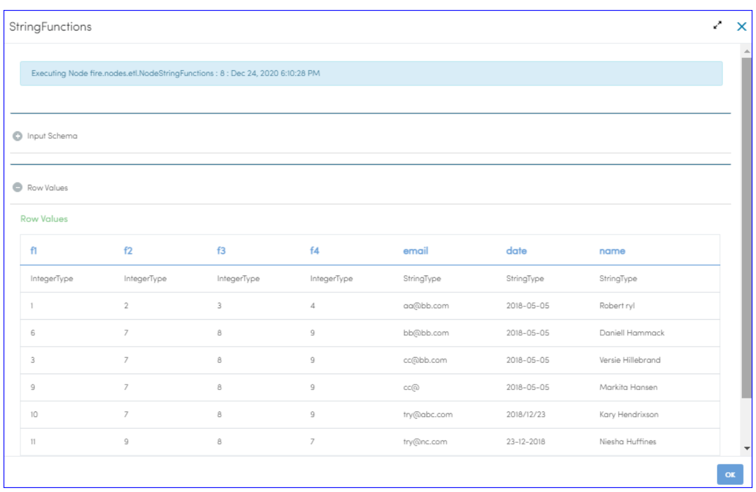
## String Functions

`StringFunctions` processor performs specified operation on the selected column (i.e. trim function for column 'name' in this case)

### Processor Configuration



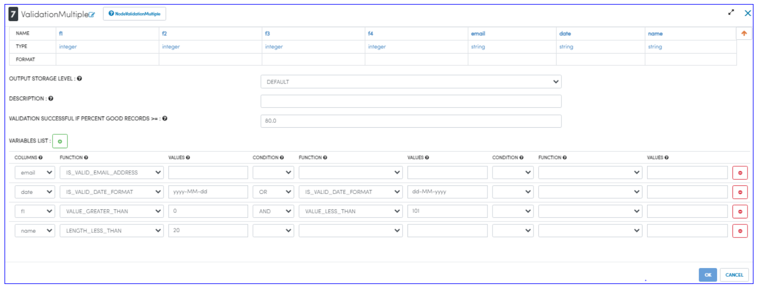
### Processor Output



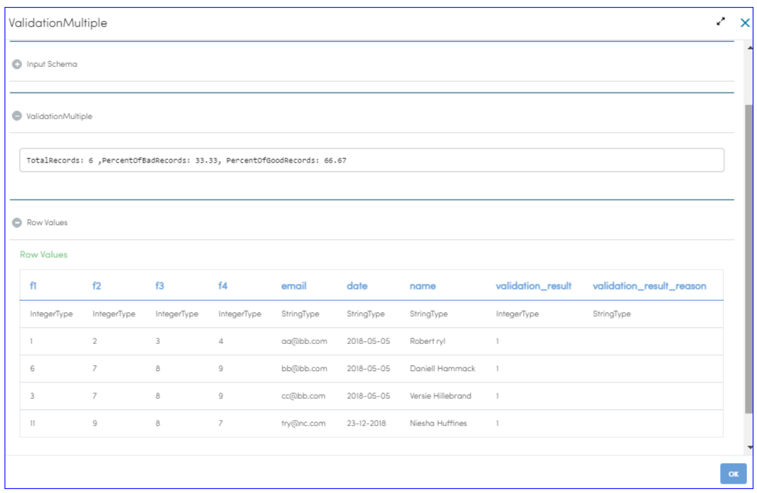
## Performing Validation

`ValidationMultiple` processor performs different validation on different columns.

### Processor Configuration



### Processor Output




### Prints the Valid Records

### Processor Output

### Prints the Invalid Records

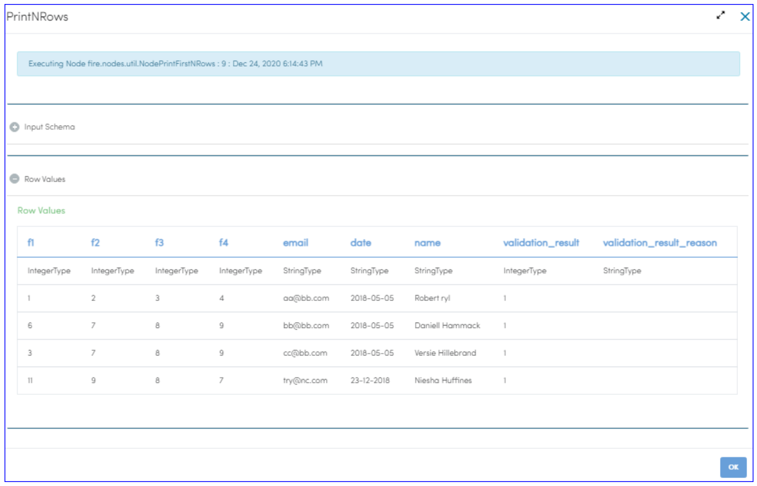
### Processor Output

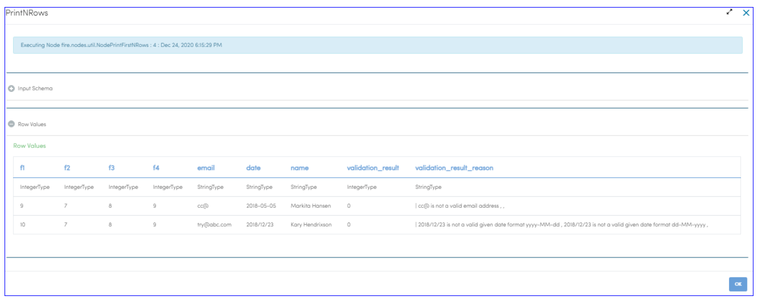
### Decision / JSON Parser / SortBy / Empty Dataset

Fire provides the following processors:

- JSON Parser Processor
- Decision Processor
- SortBy Processor
- Empty Dataset Processor

https://www.sparkflows.io/single-post/2018/09/05/New-Processors—Decision-JSON-Parser-SortBy-



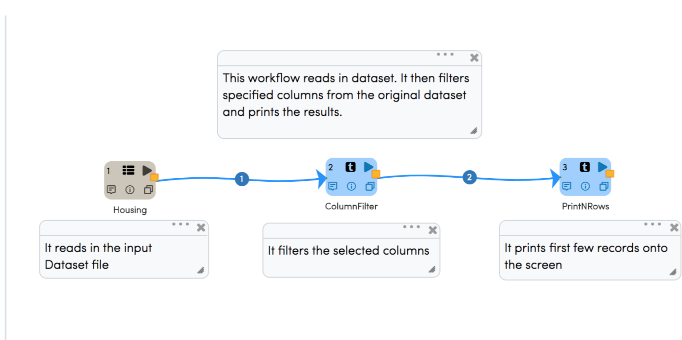

## Column Filter

This workflow reads in a dataset. It then filters specified columns from the original dataset and prints the results.

## Workflow

Below is the workflow. It does the following:

- Reads data from a dataset.
- It then filters specified columns from the original dataset.
- Prints the results.

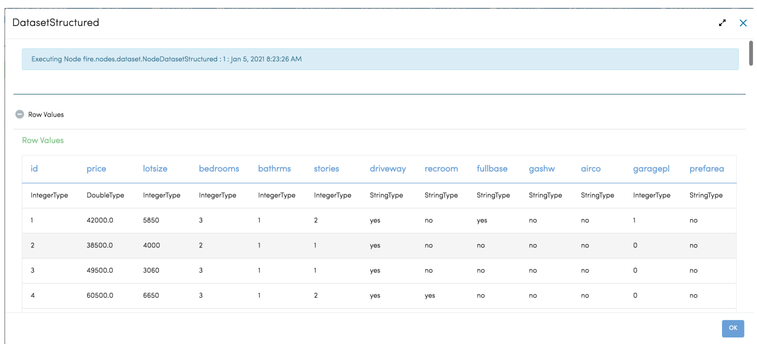


### Reading from Dataset

It reads in the input Dataset File.

### Processor Configuration

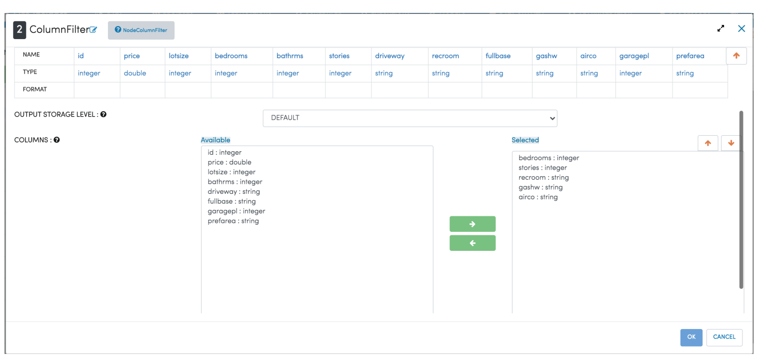

### Processor Output



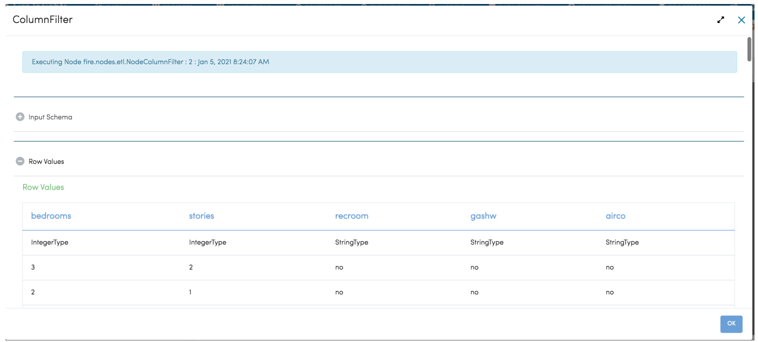

### Column Filter

It filters the selected columns.

### Processor Configuration




### Processor Output




## Prints the Results

It prints the first few records onto the screen.

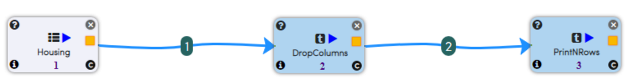
## Drop Columns

This workflow reads in a dataset. It then drops some columns from the original dataset and prints the results.

## Workflow

Below is the workflow. It does the following:

- Reads data from a dataset.
- It then drops some columns from the original dataset.
- Prints the results.

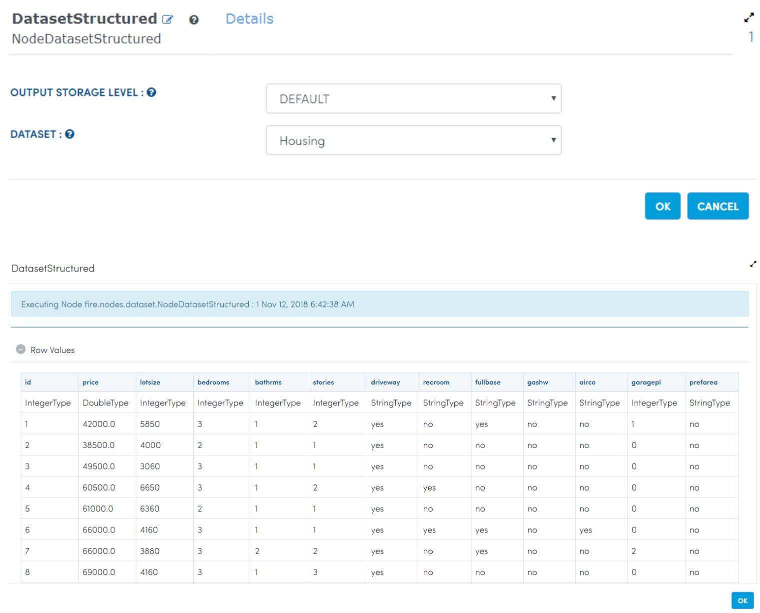



## Reading from Dataset

It reads Dataset File.

## Processor Configuration

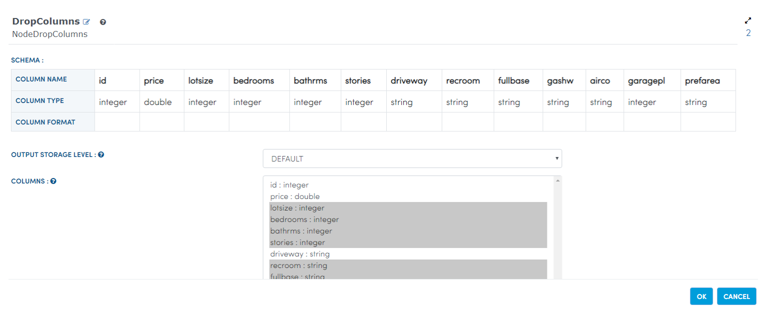
## Processor Output




## Drop Columns

It drops the columns whichever we want.

## Processor Configuration



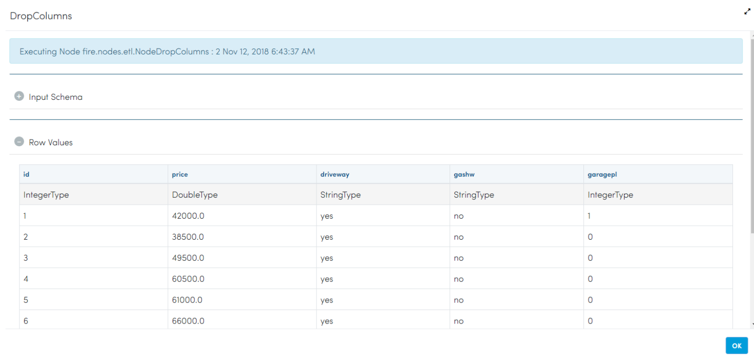

## Processor Output

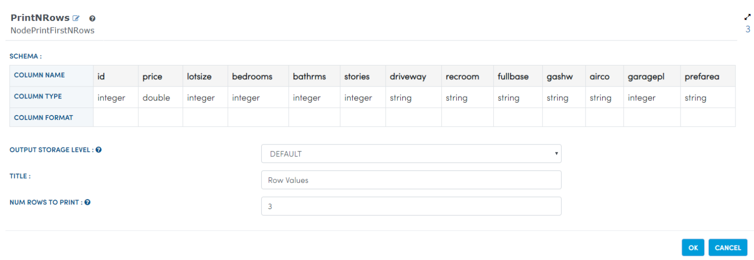
## Prints the Results

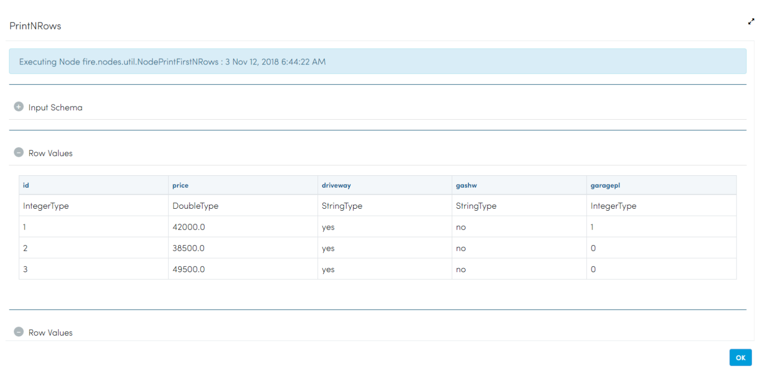
It prints the results after dropping the columns.

## Processor Configuration

## Processor Output


DropColumns
Executing Node fire.nodes.etl.NodeDropColumns : 2 Nov 12, 2018 6:43:37 AM
Input Schema
Row Values
id price driveway gashw garagepl
IntegerType DoubleType StringType StringType IntegerType
1 42000.0 yes no 1
2 38500.0 yes no 0
3 49500.0 yes no 0
4 60500.0 yes no 0
5 61000.0 yes no 0
6 66000.0 yes no 0
OK


PrintNRows
NodePrintFirstNRows
3
SCHEMA :
COLUMN NAME id price lotsize bedrooms bathrms stories driveway recroom fullbase gashw airco garagepl prefarea
COLUMN TYPE integer double integer integer integer integer string string string string string integer string
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
TITLE :
Row Values
NUM ROWS TO PRINT :
3
OK
CANCEL


PrintNRows
Executing Node fire.nodes.util.NodePrintFirstNRows : 3 Nov 12, 2018 6:44:22 AM
Input Schema
Row Values
id price driveway gashw garagepl
IntegerType DoubleType StringType StringType IntegerType
1 42000.0 yes no 1
2 38500.0 yes no 0
3 49500.0 yes no 0
Row Values
OK

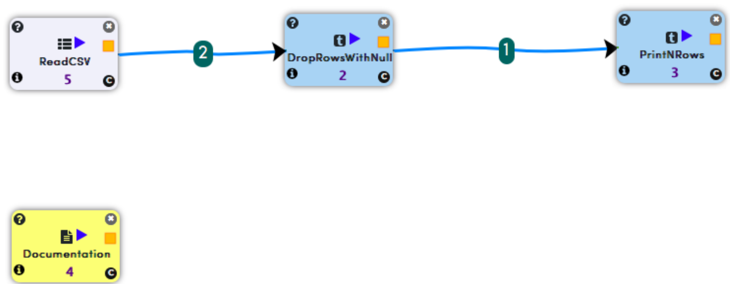
### Drop Rows With Null

This example drops/filters the rows containing any null values from the input dataset.

### Workflow

Below is the workflow. It does the following:

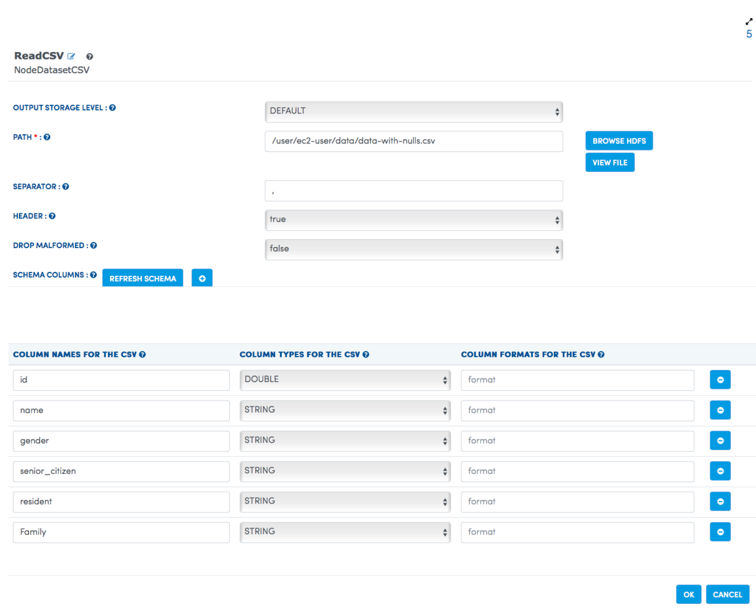
- Reads data from a CSV file.
- Drops rows having any null values in any of the columns.



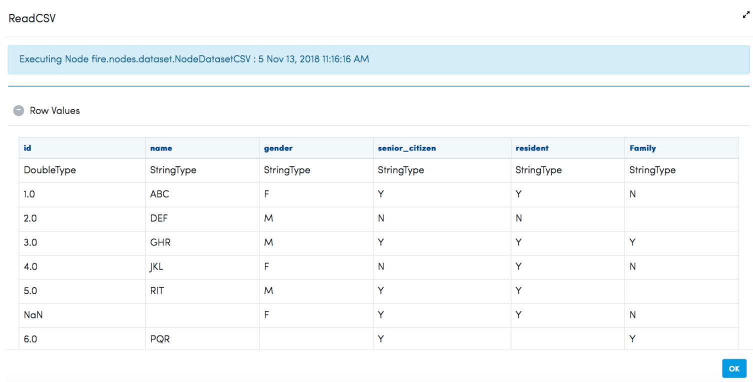

### Reading from CSV File

It reads data from a CSV file.

### Processor Configuration



### Processor Output




| id | name | gender | senior_citizen | resident | Family |
|---|---|---|---|---|---|
| DoubleType | StringType | StringType | StringType | StringType | StringType |
| 1.0 | ABC | F | Y | Y | N |
| 2.0 | DEF | M | N | N | |
| 3.0 | GHR | M | Y | Y | Y |
| 4.0 | JKL | F | N | Y | N |
| 5.0 | RIT | M | Y | Y | |
| NaN | | F | Y | Y | N |
| 6.0 | PQR | | Y | | Y |

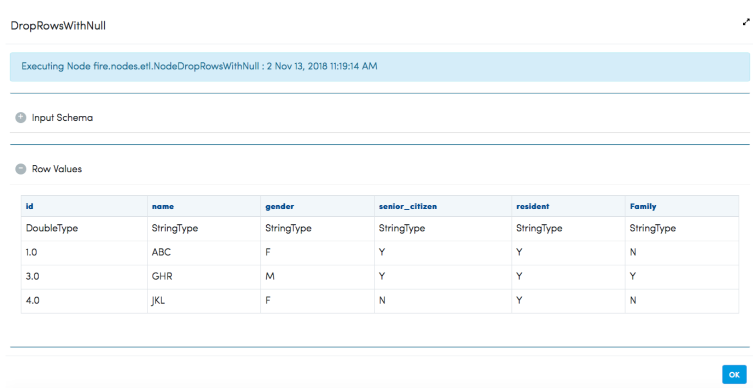
### Dropping rows with null

It drops the rows which contain any null value.

### Processor Configuration


| COLUMN NAME | id | name | gender | senior_citizen | resident | Family |
|---|---|---|---|---|---|---|
| COLUMN TYPE | double | string | string | string | string | string |
| COLUMN FORMAT | | | | | | |

### Processor Output




| id | name | gender | senior_citizen | resident | Family |
|---|---|---|---|---|---|
| DoubleType | StringType | StringType | StringType | StringType | StringType |
| 1.0 | ABC | F | Y | Y | N |
| 3.0 | GHR | M | Y | Y | Y |
| 4.0 | JKL | F | N | Y | N |

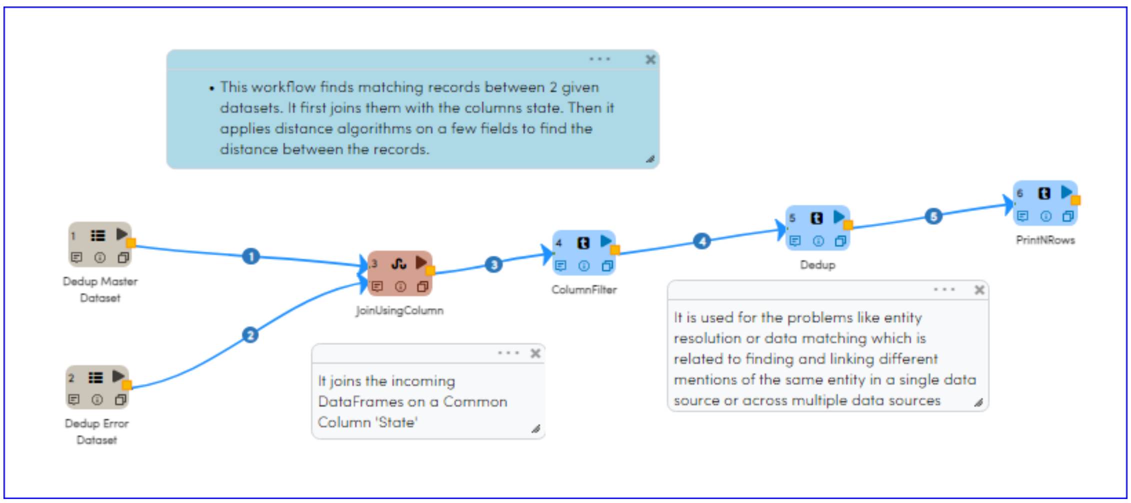
### Dedup Customers

Data deduplication refers to a technique for eliminating redundant data in a data set. In the process of deduplication, extra copies of the same data are deleted, leaving only one copy to be stored.

#### Workflow

Below is the workflow. This workflow does the following:

- Finds matching records between 2 given datasets. It first joins them with the column "State".
- Then it applies distance algorithms on a few fields to find the distance between the records.




#### Input Datasets

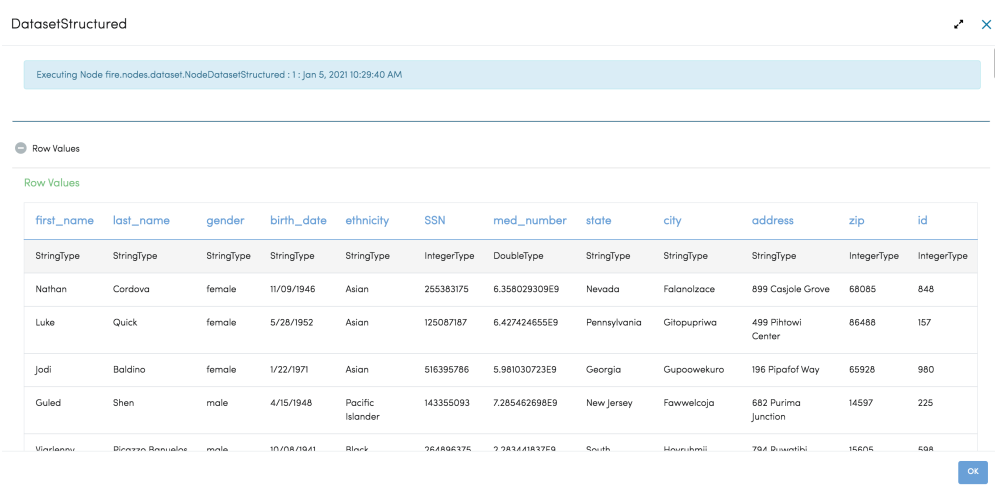
There are 2 input datasets in this case "Dedup Master Dataset" & "Dedup Error Dataset" as shown below,

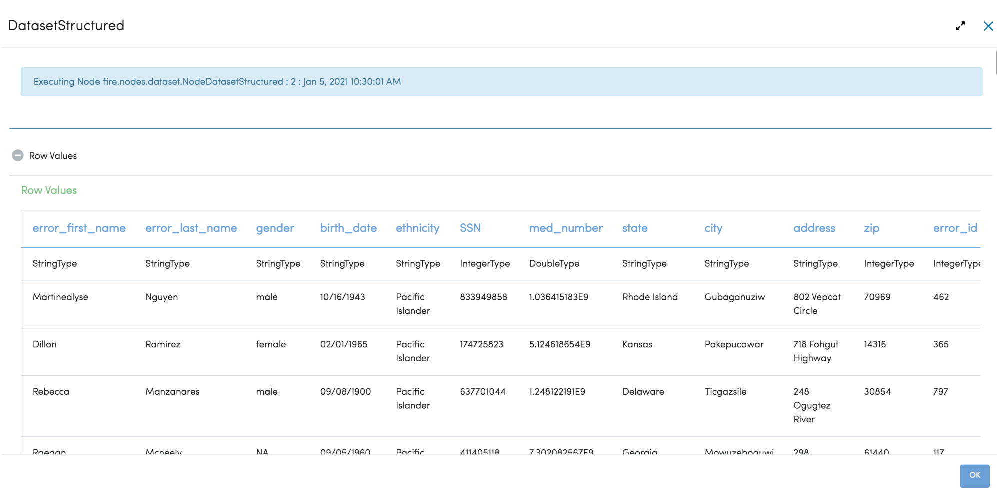
#### Dataset 1:

#### Dataset 2:

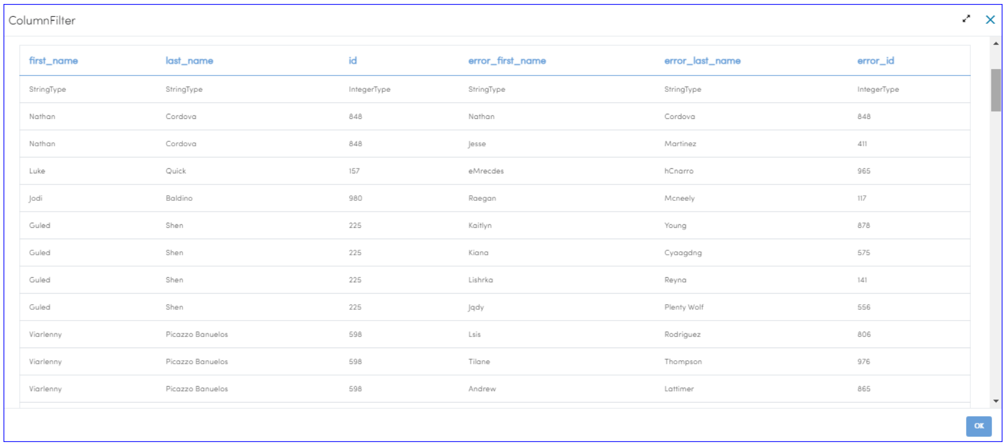
#### Join input DataFrames

`JoinUsingColumn` processor joins the incoming DataFrames on a join column


DatasetStructured
Executing Node fire.nodes.dataset.NodeDatasetStructured : 1 : Jan 5, 2021 10:29:40 AM
Row Values
Row Values
first_name last_name gender birth_date ethnicity SSN med_number state city address zip id
StringType StringType StringType StringType StringType IntegerType DoubleType StringType StringType StringType IntegerType IntegerType
Nathan Cordova female 11/09/1946 Asian 255383175 6.358029309E9 Nevada Falanolzace 899 Casjole Grove 68085 848
Luke Quick female 5/28/1952 Asian 125087187 6.427424655E9 Pennsylvania Gitopupriwa 499 Pihtowi Center 86488 157
Jodi Baldino female 1/22/1971 Asian 516395786 5.981030723E9 Georgia Gupoowekuro 196 Pipafof Way 65928 980
Guled Shen male 4/15/1948 Pacific Islander 143355093 7.285462698E9 New Jersey Fawwelcoja 682 Purima Junction 14597 225
OK


DatasetStructured
Executing Node fire.nodes.dataset.NodeDatasetStructured : 2 : Jan 5, 2021 10:30:01 AM
Row Values
Row Values
error_first_name error_last_name gender birth_date ethnicity SSN med_number state city address zip error_id
StringType StringType StringType StringType StringType IntegerType DoubleType StringType StringType StringType IntegerType
Martinealyse Nguyen male 10/16/1943 Pacific Islander 833949858 1.036415183E9 Rhode Island Gubaganuziw 802 Vepcat Circle 70969 462
Dillon Ramirez female 02/01/1965 Pacific Islander 174725823 5.124618654E9 Kansas Pakepucawar 718 Fohgut Highway 14316 365
Rebecca Manzanares male 09/08/1900 Pacific Islander 637701044 1.248122191E9 Delaware Ticgazsile 248 Ogugtez River 30854 797
OK

"State". `ColumnFilter` processor filters the columns to get the required DataFrame as shown below:


ColumnFilter

| first_name | last_name | id | error_first_name | error_last_name | error_id |
|---|---|---|---|---|---|
| StringType | StringType | IntegerType | StringType | StringType | IntegerType |
| Nathan | Cordova | 848 | Nathan | Cordova | 848 |
| Nathan | Cordova | 848 | Jesse | Martinez | 411 |
| Luke | Quick | 157 | eMrecdes | hCnarro | 965 |
| Jodi | Baldino | 980 | Raegan | Mcneely | 117 |
| Guled | Shen | 225 | Kaitlyn | Young | 878 |
| Guled | Shen | 225 | Kiana | Cyaagdng | 575 |
| Guled | Shen | 225 | Lishrka | Reyna | 141 |
| Guled | Shen | 225 | Jqdy | Plenty Wolf | 556 |
| Viarlenny | Picazzo Banuelos | 598 | Lsis | Rodriguez | 806 |
| Viarlenny | Picazzo Banuelos | 598 | Tilane | Thompson | 976 |
| Viarlenny | Picazzo Banuelos | 598 | Andrew | Lattimer | 865 |

OK

### Data Deduplication

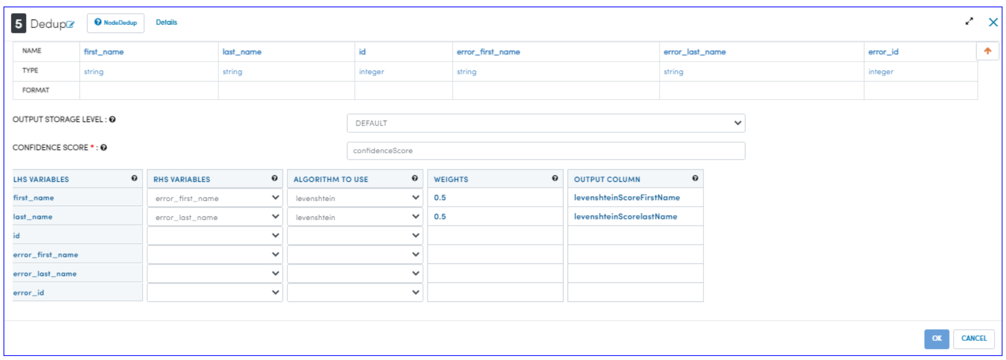
`Dedup` is used for the problems like entity resolution or data matching. Entity resolution or data matching is the problem of finding and linking different mentions of the same entity in a single data source or across multiple data sources. Here Levenshtein Algorithm is used for data Deduplication. There are more options for Algorithms that can be used:

- Full matching: Full matching makes use of all individuals in the data by forming a series of matched sets in which each set has either 1 treated individual and multiple comparison individuals or 1 comparison individual and multiple treated individuals
- Levenshtein: It counts the number of edits (insertions, deletions, or substitutions) needed to convert one string to the other.
- Jaro-Winkler: The Jaro–Winkler distance is a string metric measuring an edit distance between two sequences. Jaro-Winkler are suited for comparing smaller strings like words and names.
- Jaccard (3 gram) : This takes consecutive words and group them as a single object. A 3-gram is a consecutive set of 3 words. Used for emails or small documents.
- Longest Common Subsequence : If a set of sequences are given, the longest com-

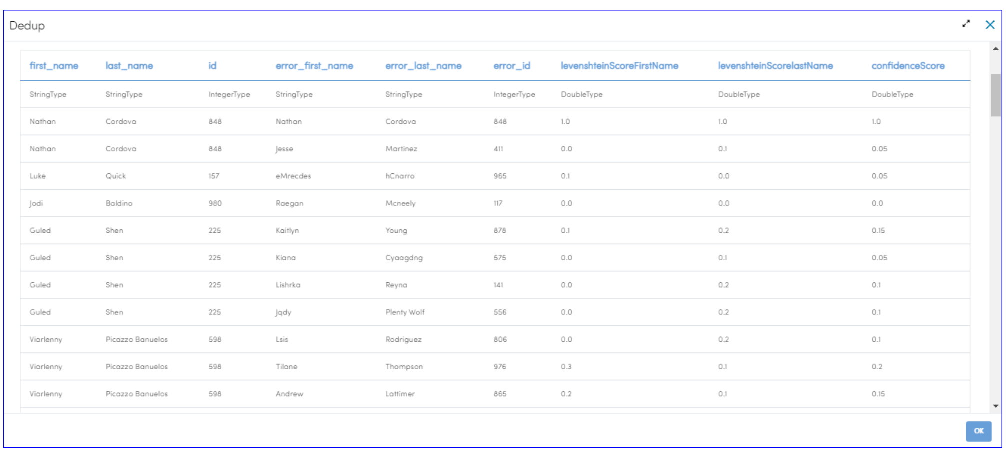
mon subsequence problem is to find a common subsequence of all the sequences that is of maximal length used in revision control systems, such as SVN and Git, for reconciling multiple changes made to a revision-controlled collection of files.

- Date Difference: Calculates the number of days between two dates.
- Notional Distance

## Dedup Processor Configuration




## Dedup Processor Output




| first_name | last_name | id | error_first_name | error_last_name | error_id | levenshteinScoreFirstName | levenshteinScorelastName | confidenceScore |
|---|---|---|---|---|---|---|---|---|
| StringType | StringType | IntegerType | StringType | StringType | IntegerType | DoubleType | DoubleType | DoubleType |
| Nathan | Cordova | 848 | Nathan | Cordova | 848 | 1.0 | 1.0 | 1.0 |
| Nathan | Cordova | 848 | Jesse | Martinez | 411 | 0.0 | 0.1 | 0.05 |
| Luke | Quick | 157 | eMrecdes | hCnarro | 965 | 0.1 | 0.0 | 0.05 |
| Jodi | Baldino | 980 | Raegan | Mcneely | 117 | 0.0 | 0.0 | 0.0 |
| Guled | Shen | 225 | Kaitlyn | Young | 878 | 0.1 | 0.2 | 0.15 |
| Guled | Shen | 225 | Kiana | Cyaagdng | 575 | 0.0 | 0.1 | 0.05 |
| Guled | Shen | 225 | Lishrka | Reyna | 141 | 0.0 | 0.2 | 0.1 |
| Guled | Shen | 225 | Jqdy | Plenty Wolf | 556 | 0.0 | 0.2 | 0.1 |
| Viarlenny | Picazzo Banuelos | 598 | Lsis | Rodriguez | 806 | 0.0 | 0.2 | 0.1 |
| Viarlenny | Picazzo Banuelos | 598 | Tilane | Thompson | 976 | 0.3 | 0.1 | 0.2 |
| Viarlenny | Picazzo Banuelos | 598 | Andrew | Lattimer | 865 | 0.2 | 0.1 | 0.15 |

## Prints the Results

It prints the first few records onto the screen.

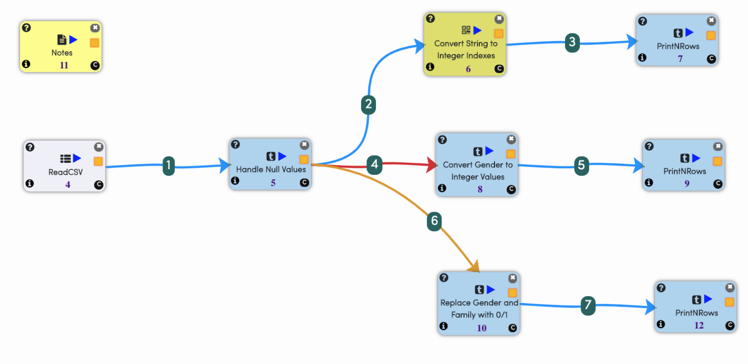
### Handling Null Values

This example removes null values from the input dataset.

#### Workflow

Below is the workflow. It does the following:

- Reads data from a CSV file.
- Replaces null values in certain columns with constant values.
- Converts certain columns to 0/1 based on their value. It does it in 3 different ways.
  - Using StringIndexer Processor
  - Using CaseWhen Processor
  - Using FindAndReplaceUsingRegex Processor




#### Reading from CSV File

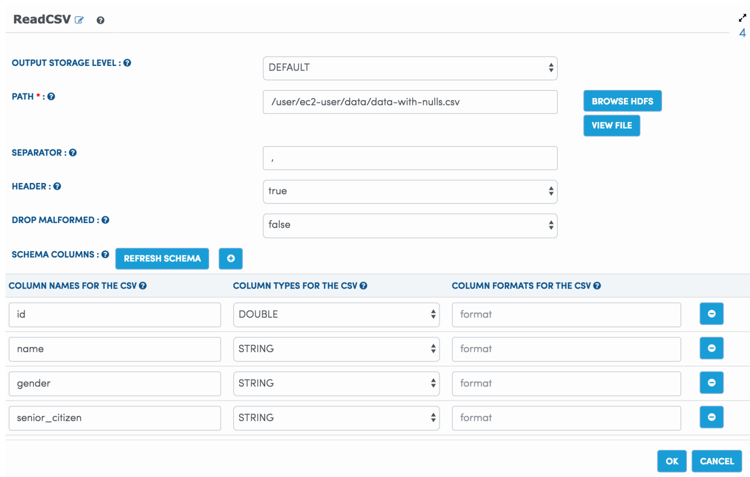
It reads in the CSV file data-with-nulls.csv.

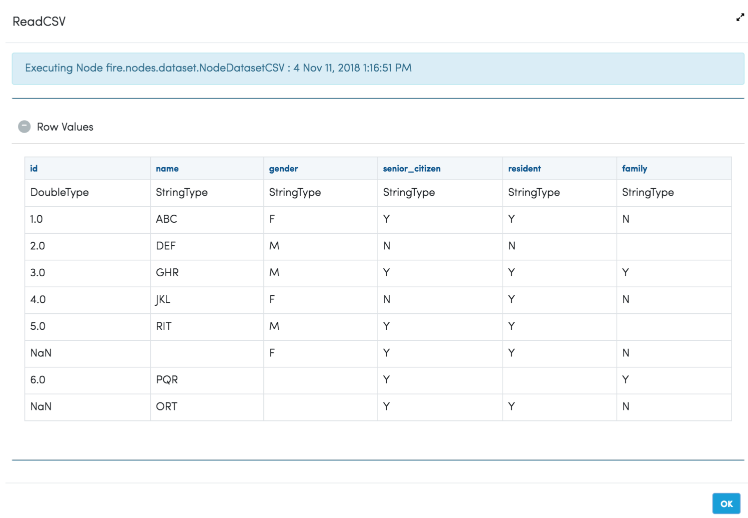
#### Processor Configuration

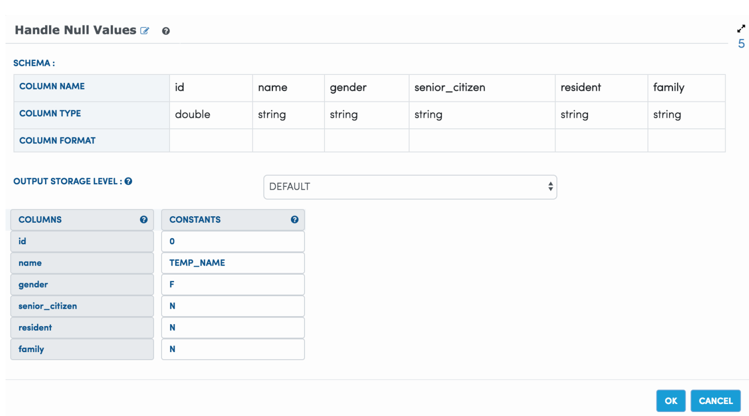
#### Processor Output

#### Replacing null values

It replaces null values in certain columns with user defined constant values.


ReadCSV
4
OUTPUT STORAGE LEVEL :
DEFAULT
PATH * :
/user/ec2-user/data/data-with-nulls.csv
BROWSE HDFS
VIEW FILE
SEPARATOR :
,
HEADER :
true
DROP MALFORMED :
false
SCHEMA COLUMNS :
REFRESH SCHEMA
COLUMN NAMES FOR THE CSV
COLUMN TYPES FOR THE CSV
COLUMN FORMATS FOR THE CSV
id
DOUBLE
format
name
STRING
format
gender
STRING
format
senior_citizen
STRING
format
OK
CANCEL


ReadCSV
Executing Node fire.nodes.dataset.NodeDatasetCSV : 4 Nov 11, 2018 1:16:51 PM
Row Values
id name gender senior_citizen resident family
DoubleType StringType StringType StringType StringType StringType
1.0 ABC F Y Y N
2.0 DEF M N N
3.0 GHR M Y Y Y
4.0 JKL F N Y N
5.0 RIT M Y Y
NaN F Y Y N
6.0 PQR Y Y
NaN ORT Y Y N
OK


Handle Null Values
5
SCHEMA :
COLUMN NAME id name gender senior_citizen resident family
COLUMN TYPE double string string string string string
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
COLUMNS
CONSTANTS
id 0
name TEMP_NAME
gender F
senior_citizen N
resident N
family N
OK
CANCEL

### Processor Configuration

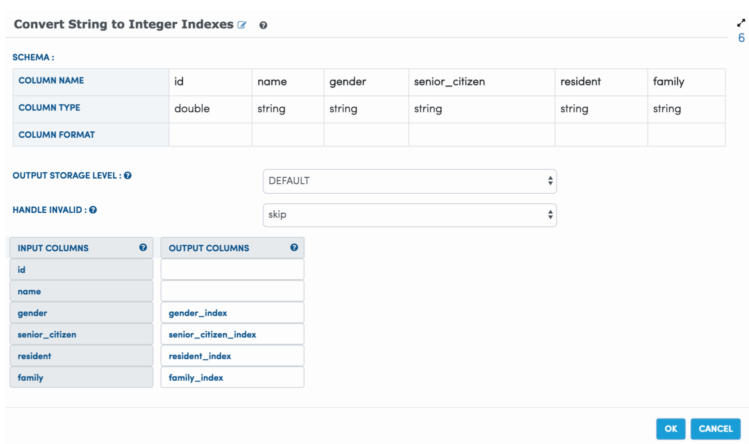
### Processor Output

| id | name | gender | senior_citizen | resident | family |
|---|---|---|---|---|---|
| DoubleType | StringType | StringType | StringType | StringType | StringType |
| 1.0 | ABC | F | Y | Y | N |
| 2.0 | DEF | M | N | N | N |
| 3.0 | GHR | M | Y | Y | Y |
| 4.0 | JKL | F | N | Y | N |
| 5.0 | RIT | M | Y | Y | N |
| 0.0 | TEMP_NAME | F | Y | Y | N |
| 6.0 | PQR | F | Y | N | Y |
| 0.0 | ORT | F | Y | Y | N |

### Converting to 0/1 using StringIndexer

It converts strings like Y/N to 0/1 for the specified columns using the StringIndexer Processor.

### Processor Configuration



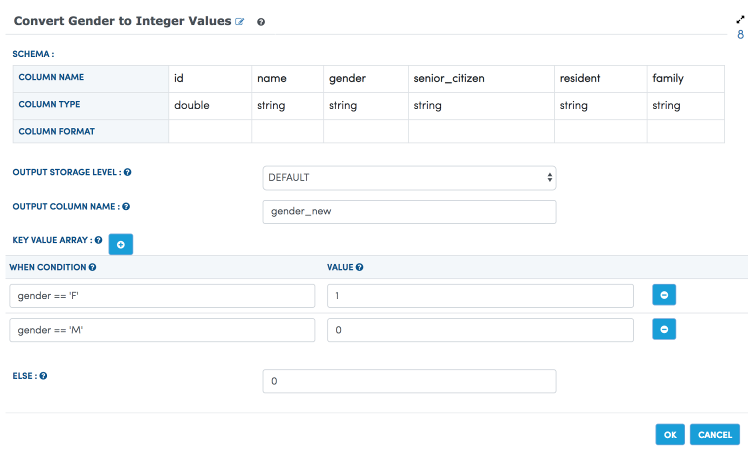

### Processor Output

### Converting to 0/1 using CaseWhen

It converts strings like Y/N to 0/1 for the specified columns using the CaseWhen Processor.

### Processor Configuration

| id | name | gender | senior_citizen | resident | family | gender_index | senior_citizen_index | resident_index | family_index |
|---|---|---|---|---|---|---|---|---|---|
| DoubleType | StringType | StringType | StringType | StringType | StringType | DoubleType | DoubleType | DoubleType | DoubleType |
| 1.0 | ABC | F | Y | Y | N | 0.0 | 0.0 | 0.0 | 0.0 |
| 2.0 | DEF | M | N | N | N | 1.0 | 1.0 | 1.0 | 0.0 |
| 3.0 | GHR | M | Y | Y | Y | 1.0 | 0.0 | 0.0 | 1.0 |
| 4.0 | JKL | F | N | Y | N | 0.0 | 1.0 | 0.0 | 0.0 |
| 5.0 | RIT | M | Y | Y | N | 1.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | TEMP_NAME | F | Y | Y | N | 0.0 | 0.0 | 0.0 | 0.0 |
| 6.0 | PQR | F | Y | N | Y | 0.0 | 0.0 | 1.0 | 1.0 |
| 0.0 | ORT | F | Y | Y | N | 0.0 | 0.0 | 0.0 | 0.0 |




### Processor Output

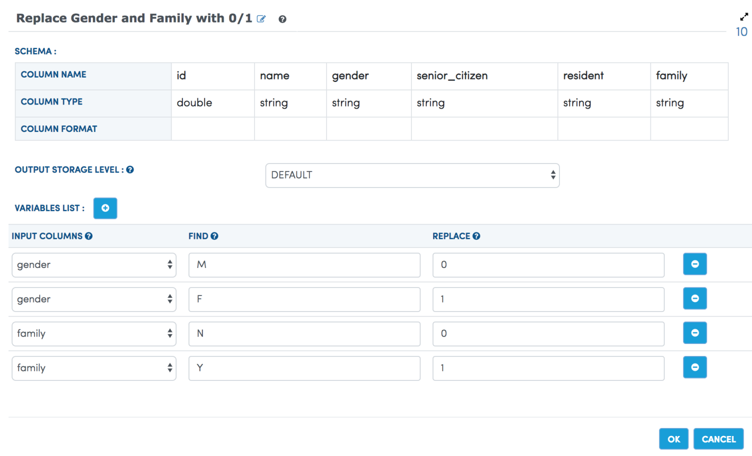
| id | name | gender | senior_citizen | resident | family | gender_new |
|---|---|---|---|---|---|---|
| DoubleType | StringType | StringType | StringType | StringType | StringType | IntegerType |
| 1.0 | ABC | F | Y | Y | N | 1 |
| 2.0 | DEF | M | N | N | N | 0 |
| 3.0 | GHR | M | Y | Y | Y | 0 |
| 4.0 | JKL | F | N | Y | N | 1 |
| 5.0 | RIT | M | Y | Y | N | 0 |
| 0.0 | TEMP_NAME | F | Y | Y | N | 1 |
| 6.0 | PQR | F | Y | N | Y | 1 |
| 0.0 | ORT | F | Y | Y | N | 1 |

## Converting to 0/1 using FindAndReplaceUsingRegex

It converts strings like Y/N to 0/1 for the specified columns using the FindAndReplaceUsingRegex Processor.

### Processor Configuration

### Processor Output




| id | name | senior_citizen | resident | gender | family |
|---|---|---|---|---|---|
| DoubleType | StringType | StringType | StringType | StringType | StringType |
| 1.0 | ABC | Y | Y | 1 | 0 |
| 2.0 | DEF | N | N | 0 | 0 |
| 3.0 | GHR | Y | Y | 0 | 1 |
| 4.0 | JKL | N | Y | 1 | 0 |
| 5.0 | RIT | Y | Y | 0 | 0 |
| 0.0 | TEMP_NAME | Y | Y | 1 | 0 |
| 6.0 | PQR | Y | N | 1 | 1 |
| 0.0 | ORT | Y | Y | 1 | 0 |

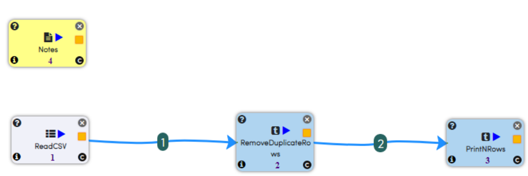
## Remove Duplicate Rows

This workflow reads CSV file. It then removes duplicate rows from the original CSV file and prints the results.

### Workflow

Below is the workflow. It does the following:

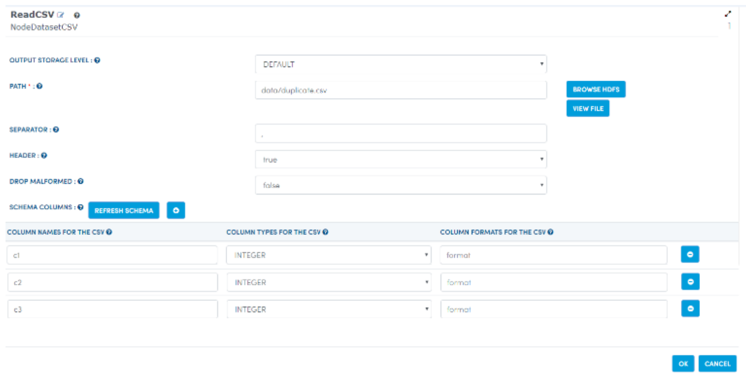
- Reads data from a CSV file.
- It then removes duplicate rows from the original CSV file.
- Prints the results.



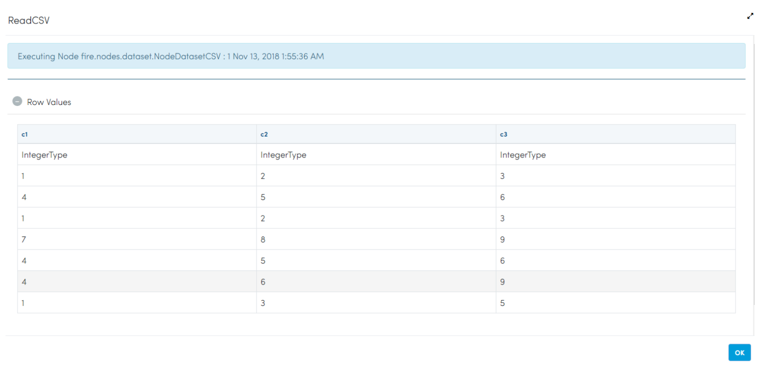
### Reading from CSV file

It reads CSV file.

#### Processor Configuration




#### Processor Output




### Remove Duplicate Rows

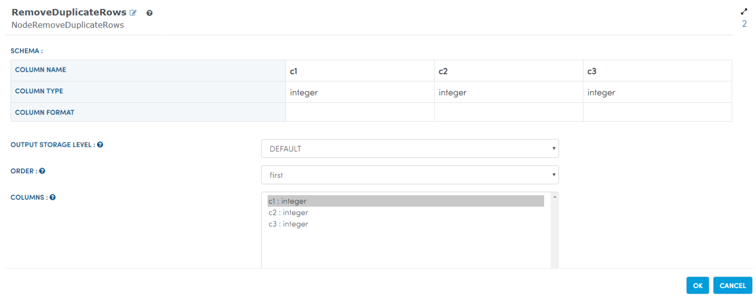
It removes Duplicate Rows available.

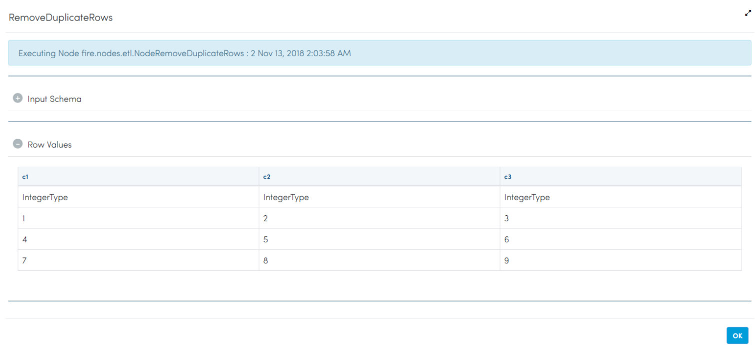
#### Processor Configuration

#### Processor Output

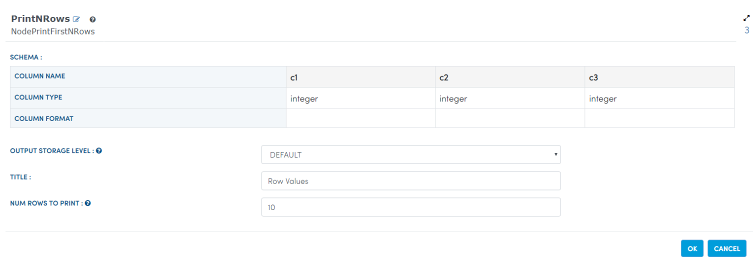
### Prints the Results

It prints the results after Removing Duplicate Rows.




## Processor Configuration




## Processor Output

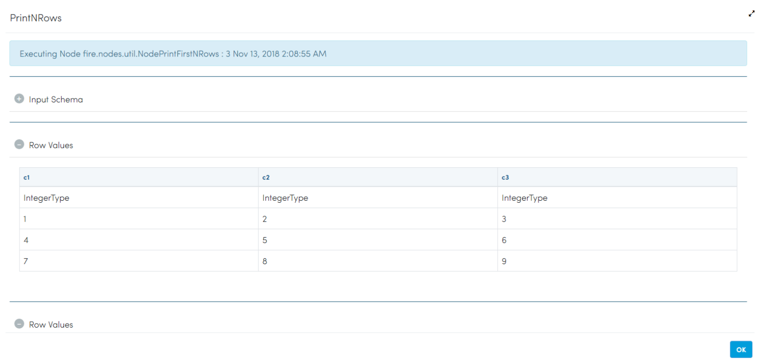
## Removing Special Characters

This workflow reads in a dataset. It then removes the special characters from columns of the original dataset and prints the results.

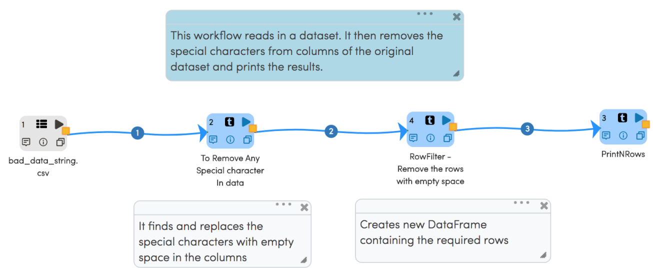
## Workflow

Below is the workflow. It does the following:

- It reads the CSV and creates a DataFrame.
- It find and replaces the special characters with empty space in the columns



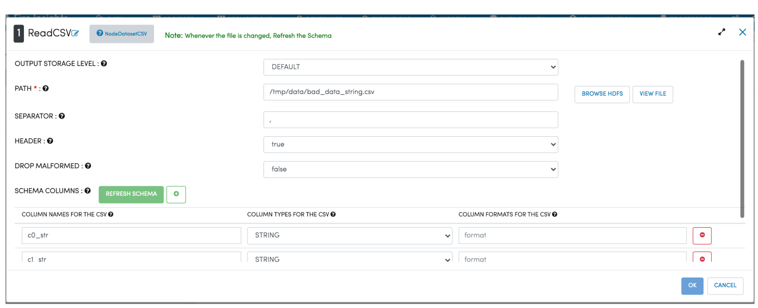

- Create new DataFrame containing the rows that satisfy the given condition (i.e. removes the rows with empty space)
- Print the specified number of records in the DataFrame after execution of workflow



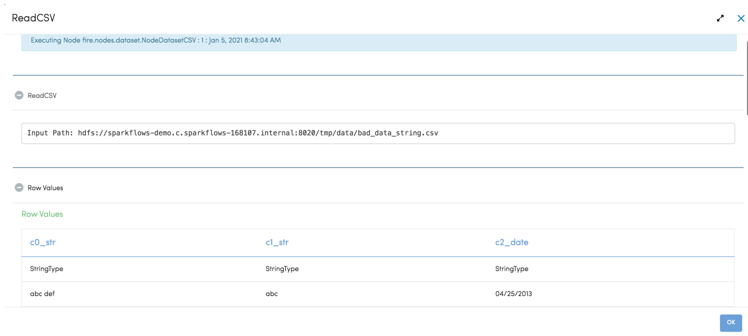

## Reading from Dataset

DatasetCSV processor reads in the input Dataset file and creates DataFrame.

## Processor Configuration



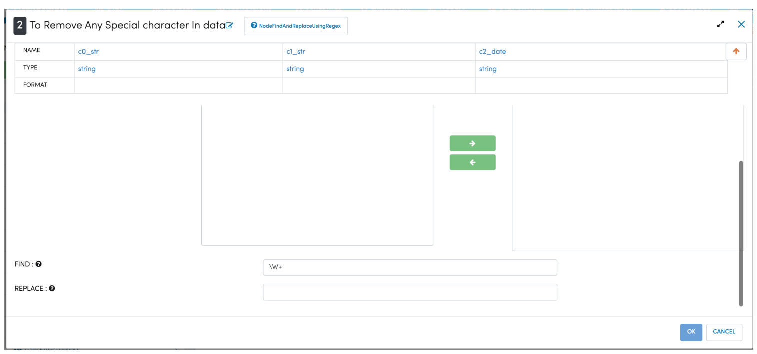

## Processor Output



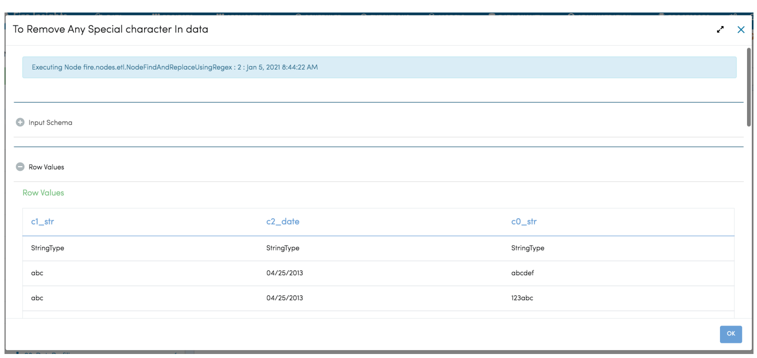

### To Remove Any Special character in data

FindAndReplaceUsingRegex processor find and replaces the special characters with empty space in the columns

### Processor Configuration



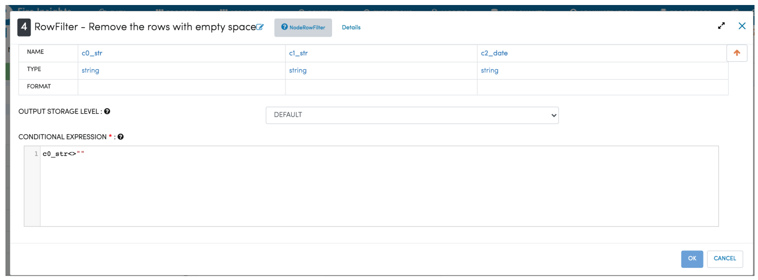

### Processor Output



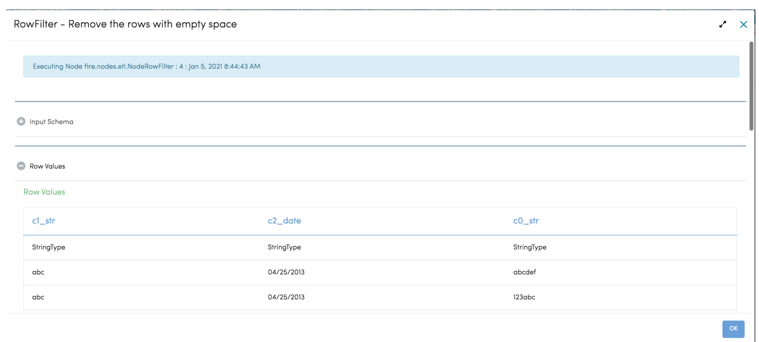

### RowFilter - Remove the rows with empty space

RowFilter processor creates new DataFrame containing the rows that satisfy the condition provided (For example : Removes the rows with empty spaces as shown below)

### Processor Configuration




### Processor Output




### Prints the Results

It prints the first few records onto the screen.

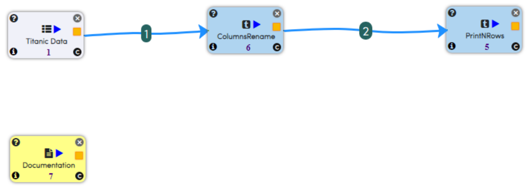
### Rename Columns

This workflow reads in a dataset. It then renames columns from the original dataset and prints the results.

### Workflow

Below is the workflow. It does the following:

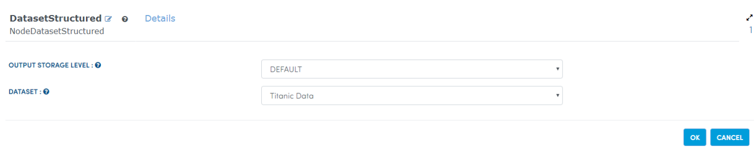
- Reads data from a dataset.
- It then renames columns from the original dataset.
- Prints the results.



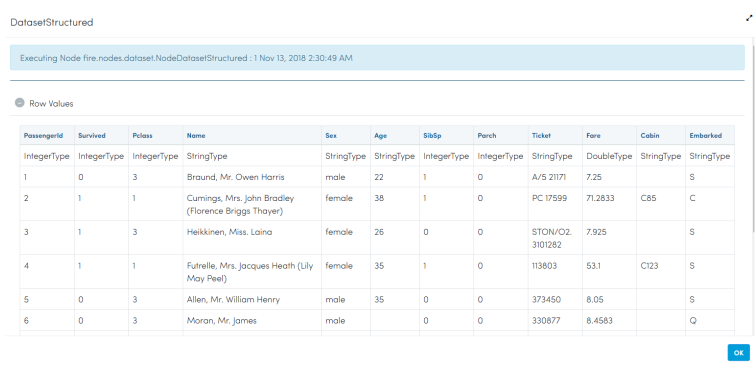

## Reading from Dataset

It reads Dataset file.

## Processor Configuration




## Processor Output


DatasetStructured

Executing Node fire.nodes.dataset.NodeDatasetStructured : 1 Nov 13, 2018 2:30:49 AM

Row Values

| PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | IntegerType | IntegerType | StringType | StringType | StringType | IntegerType | IntegerType | StringType | DoubleType | StringType | StringType |
| 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22 | 1 | 0 | A/5 21171 | 7.25 | | S |
| 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Thayer) | female | 38 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26 | 0 | 0 | STON/O2. 3101282 | 7.925 | | S |
| 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35 | 1 | 0 | 113803 | 53.1 | C123 | S |
| 5 | 0 | 3 | Allen, Mr. William Henry | male | 35 | 0 | 0 | 373450 | 8.05 | | S |
| 6 | 0 | 3 | Moran, Mr. James | male | | 0 | 0 | 330877 | 8.4583 | | Q |

OK

## Rename Columns

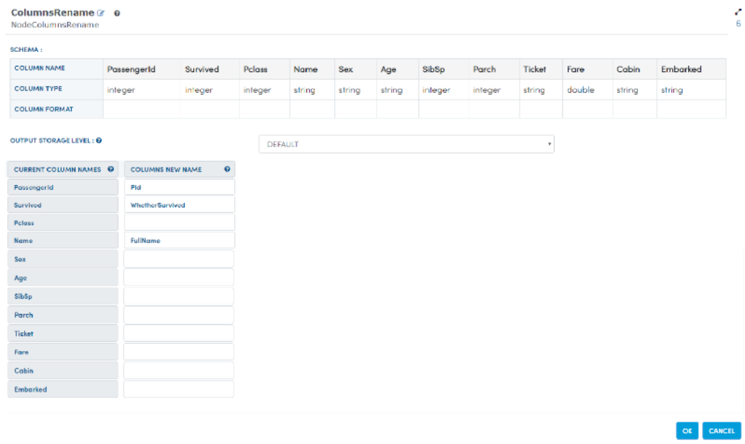
It renames columns we want.

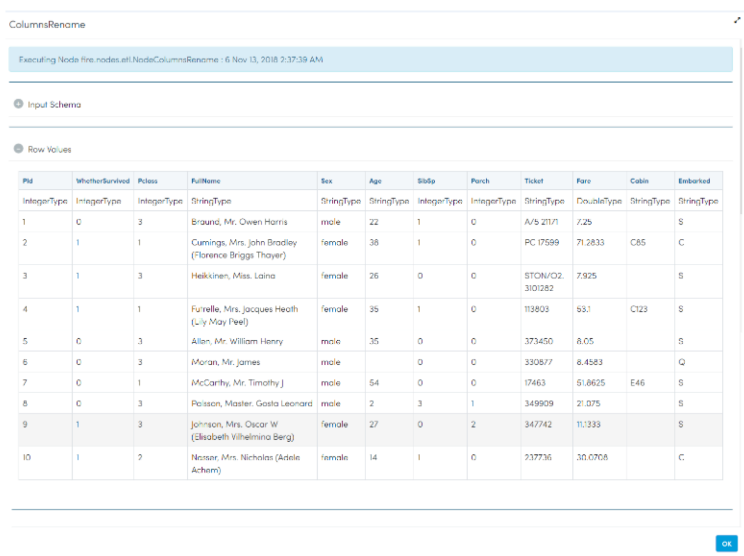
## Processor Configuration

## Processor Output

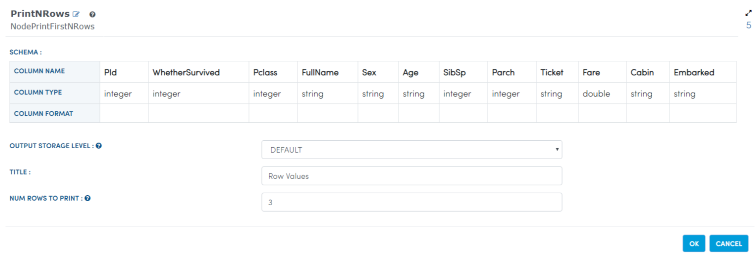
## Prints the Results

It prints the results after Renaming Columns.

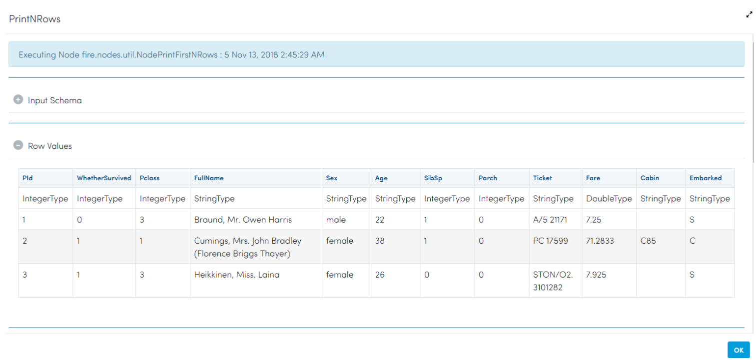

ColumnsRename
NodeColumnsRename
SCHEMA :
COLUMN NAME
COLUMN TYPE
COLUMN FORMAT
PassengerId
Survived
Pclass
Name
Sex
Age
SibSp
Parch
Ticket
Fare
Cabin
Embarked
integer
string
double
OUTPUT STORAGE LEVEL :
DEFAULT
CURRENT COLUMN NAMES
COLUMNS NEW NAME
PId
WhetherSurvived
FullName
OK
CANCEL


ColumnsRename
Input Schema
Row Values
PId
WhetherSurvived
Pclass
FullName
Sex
Age
SibSp
Parch
Ticket
Fare
Cabin
Embarked
IntegerType
StringType
DoubleType
OK

### Processor Configuration




### Processor Output



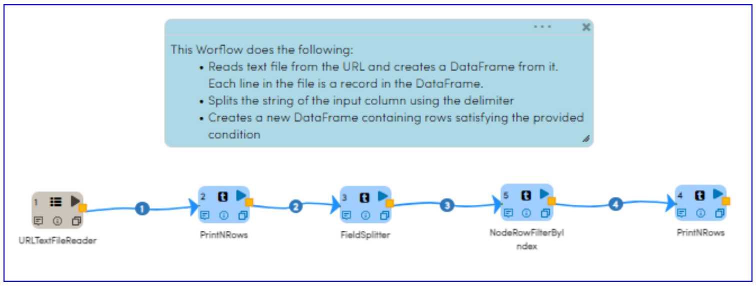

### REST - CSV Reader & Parse

This workflow reads in a dataset from URL. It then parses the dataset and prints the results.

### Workflow

Below is the workflow. It does the following:

- Reads data from the URL and creates a DataFrame
- Prints few records
- Splits the string of the input column using the delimiter
- Creates a new DataFrame containing rows satisfying the provided condition
- Prints the result

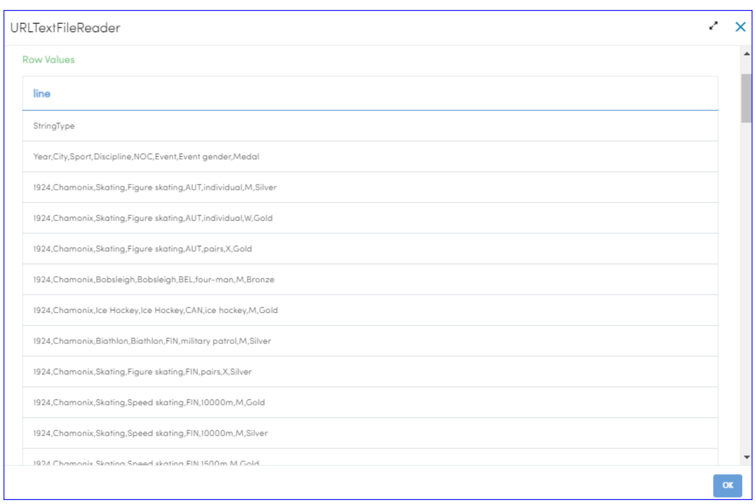



### Reading from URL

DatasetURLTextFileReader processor uses the passed URL to download the data and create the DataFrame.

### Processor Configuration


### Processor Output



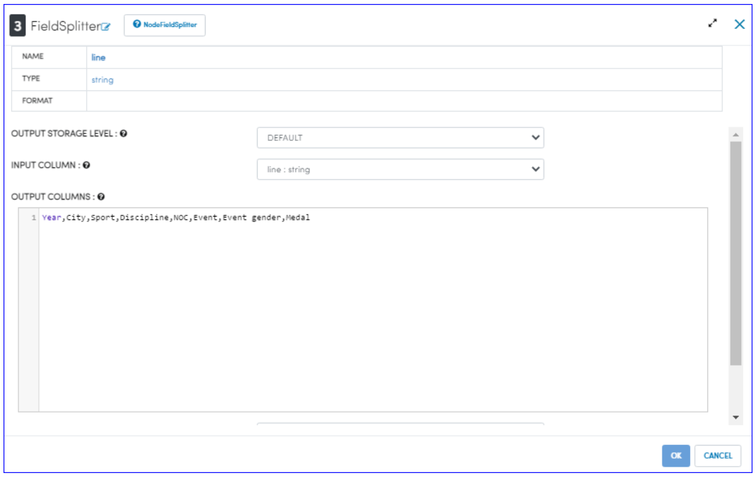

### Prints the Records

It prints the first few records onto the screen.

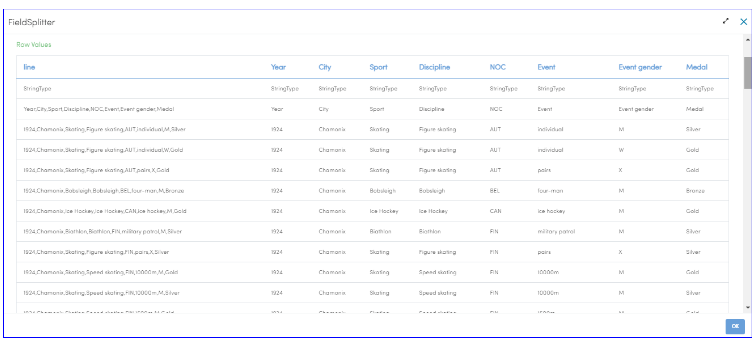
### Parsing the DataFrame

`FieldSplitter` processor parses and creates new DataFrame by splitting the string of the input column using the delimiter as shown below:

### Processor Configuration




### Processor Output

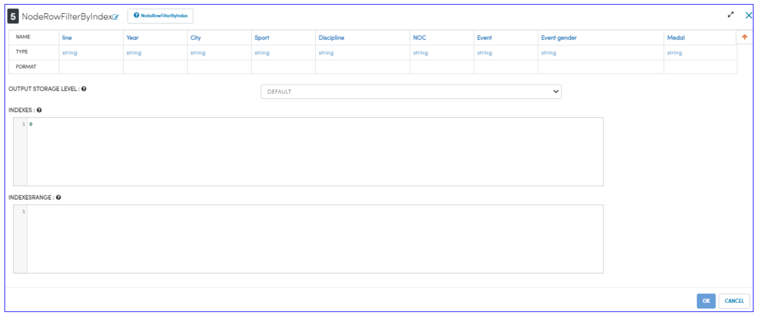


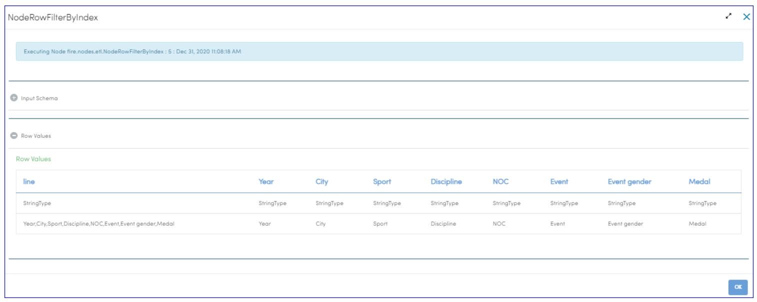

### Row Filter by Index

`RowFilterByIndex` processor creates a new DataFrame containing required rows as shown below:

### Processor Configuration

### Processor Output




### Prints the Results

It prints the result onto the screen.

### REST Read And Parse JSON

This workflow reads in single record JSON from the given URL. It then parses the dataset and prints the results.

### Workflow

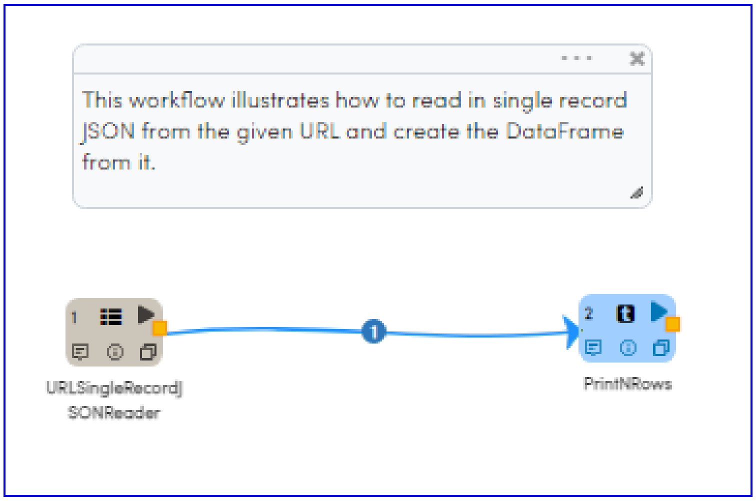
Below is the workflow that shows:

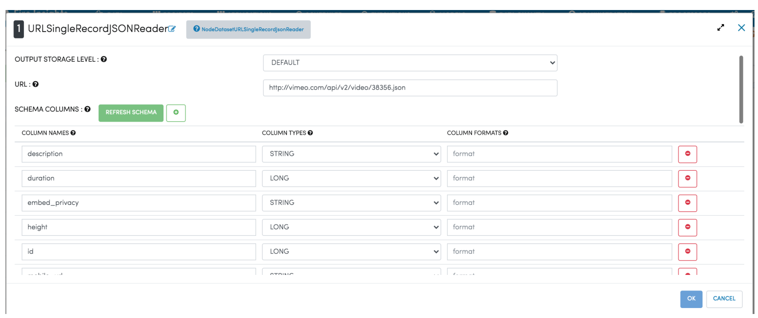
- How to read in single record JSON from the given URL and create the DataFrame from it
- Prints the result

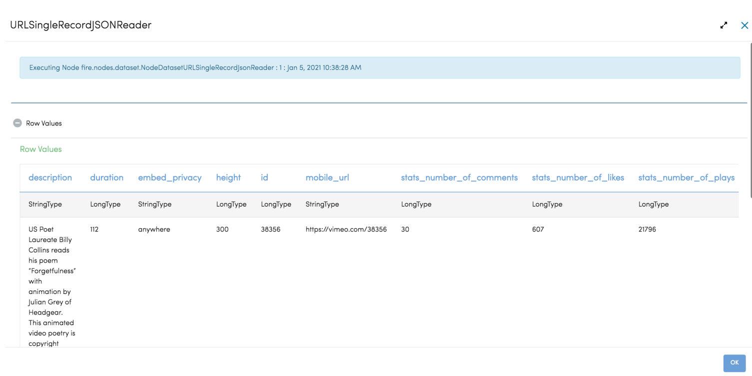
### Reading from URL And Parsing

`DatasetURLSingleRecordJSONReader` processor uses the passed URL to download single record JSON, parse the dataset and create the DataFrame.

### Processor Configuration


This workflow illustrates how to read in single record JSON from the given URL and create the DataFrame from it.
URLSingleRecordJSONReader
PrintNRows


URLSingleRecordJSONReader
OUTPUT STORAGE LEVEL :
DEFAULT
URL :
http://vimeo.com/api/v2/video/38356.json
SCHEMA COLUMNS :
REFRESH SCHEMA
COLUMN NAMES
COLUMN TYPES
COLUMN FORMATS
description
STRING
duration
LONG
embed_privacy
STRING
height
LONG
id
LONG
OK
CANCEL


URLSingleRecordJSONReader
Executing Node fire.nodes.dataset.NodeDatasetURLSingleRecordJsonReader : 1 : Jan 5, 2021 10:38:28 AM
Row Values
Row Values
description
duration
embed_privacy
height
id
mobile_url
stats_number_of_comments
stats_number_of_likes
stats_number_of_plays
StringType
LongType
StringType
LongType
LongType
StringType
LongType
LongType
LongType
US Poet Laureate Billy Collins reads his poem "Forgetfulness" with animation by Julian Grey of Headgear. This animated video poetry is copyright
112
anywhere
300
38356
https://vimeo.com/38356
30
607
21796
OK

#### Processor Output

#### Prints the Results

It prints the result onto the screen.

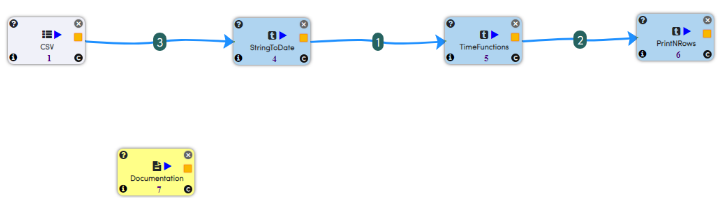
### String To Date Timefunctions

This workflow reads a CSV file. It then converts it into stringtodate and then to timefunctions and prints the results.

#### Workflow

Below is the workflow. It does the following:

- Reads a CSV file.
- It then converts it into stringtodate using stringtodate Node.
- Convert it into timefunctions using timefunctions Node.
- Prints the results.



#### Reading from CSV file

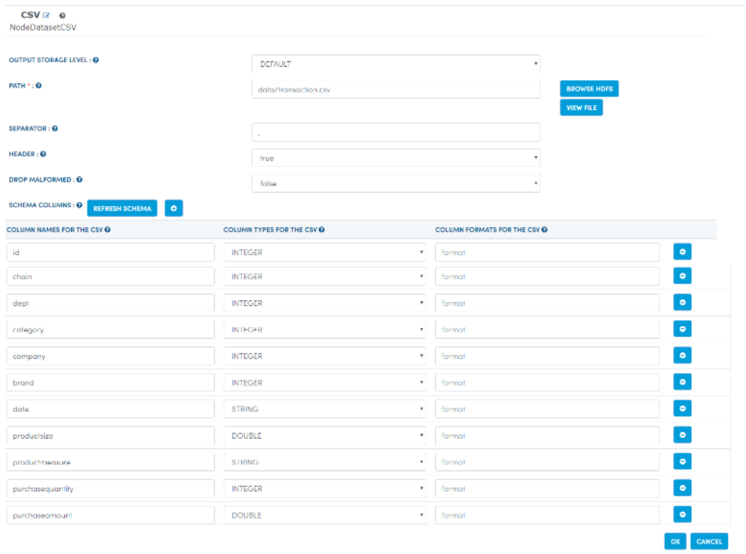
It reads Data from CSV file.

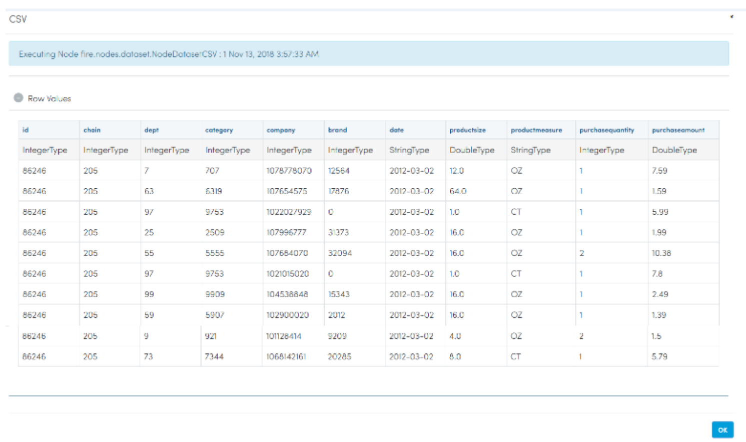
#### Processor Configuration

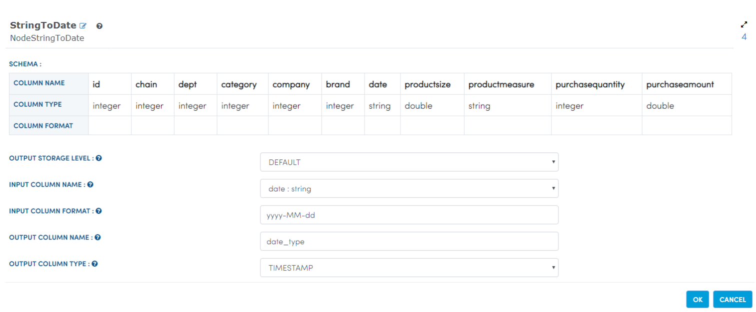
#### Processor Output

#### String to Date

It converts it into stringtodate using stringtodate Node.

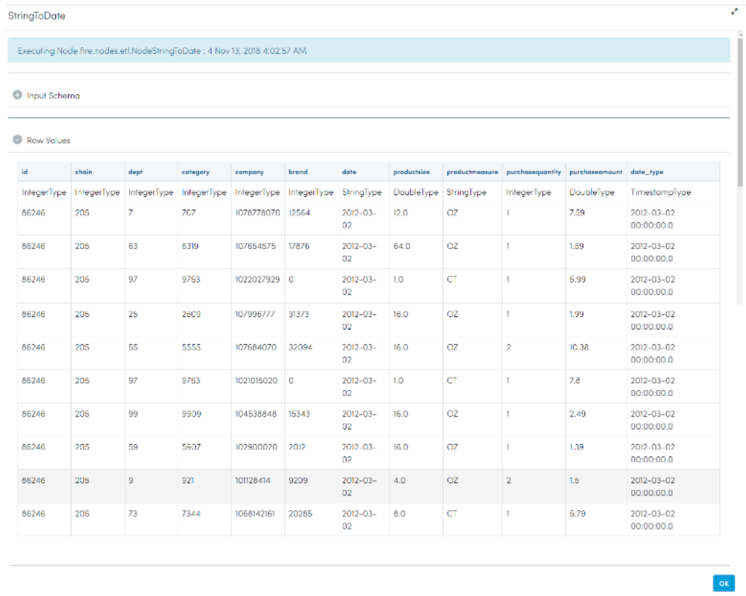

CSV
NodeDatasetCSV
OUTPUT STORAGE LEVEL :
DEFAULT
PATH * :
data/transaction.csv
BROWSE HDFS
VIEW FILE
SEPARATOR :
HEADER :
true
DROP MALFORMED :
false
SCHEMA COLUMNS :
REFRESH SCHEMA
COLUMN NAMES FOR THE CSV
COLUMN TYPES FOR THE CSV
COLUMN FORMATS FOR THE CSV
OK
CANCEL


CSV
Row Values
OK


StringToDate
NodeStringToDate
SCHEMA :
COLUMN NAME
COLUMN TYPE
COLUMN FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
INPUT COLUMN NAME :
date : string
INPUT COLUMN FORMAT :
yyyy-MM-dd
OUTPUT COLUMN NAME :
date_type
OUTPUT COLUMN TYPE :
TIMESTAMP
OK
CANCEL

### Processor Configuration

### Processor Output

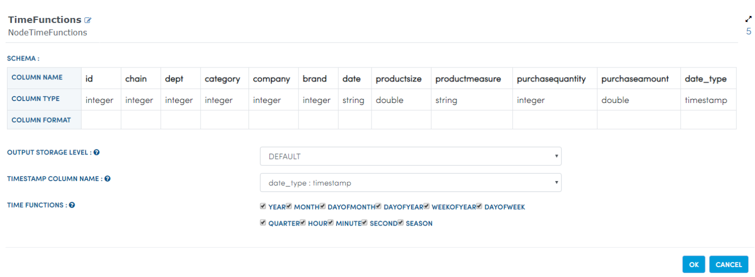



| id | chain | dept | category | company | brand | date | productsize | productmeasure | purchasequantity | purchaseamount | date_type |
|---|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | StringType | DoubleType | StringType | IntegerType | DoubleType | TimestampType |
| 86246 | 205 | 7 | 707 | 1078778070 | 12564 | 2012-03-02 | 12.0 | OZ | 1 | 7.59 | 2012-03-02 00:00:00.0 |
| 86246 | 205 | 63 | 6319 | 107654575 | 17876 | 2012-03-02 | 64.0 | OZ | 1 | 1.59 | 2012-03-02 00:00:00.0 |
| 86246 | 205 | 97 | 9753 | 1022027929 | 0 | 2012-03-02 | 1.0 | CT | 1 | 5.99 | 2012-03-02 00:00:00.0 |
| 86246 | 205 | 25 | 2509 | 107996777 | 31373 | 2012-03-02 | 16.0 | OZ | 1 | 1.99 | 2012-03-02 00:00:00.0 |
| 86246 | 205 | 55 | 5555 | 107684070 | 32094 | 2012-03-02 | 16.0 | OZ | 2 | 10.38 | 2012-03-02 00:00:00.0 |
| 86246 | 205 | 97 | 9753 | 1021015020 | 0 | 2012-03-02 | 1.0 | CT | 1 | 7.8 | 2012-03-02 00:00:00.0 |
| 86246 | 205 | 99 | 9909 | 104538848 | 15343 | 2012-03-02 | 16.0 | OZ | 1 | 2.49 | 2012-03-02 00:00:00.0 |
| 86246 | 205 | 59 | 5907 | 102900020 | 2012 | 2012-03-02 | 16.0 | OZ | 1 | 1.39 | 2012-03-02 00:00:00.0 |
| 86246 | 205 | 9 | 921 | 101128414 | 9209 | 2012-03-02 | 4.0 | OZ | 2 | 1.5 | 2012-03-02 00:00:00.0 |
| 86246 | 205 | 73 | 7344 | 1068142161 | 20285 | 2012-03-02 | 8.0 | CT | 1 | 5.79 | 2012-03-02 00:00:00.0 |

### Time Functions

It converts it into timefunctions using timefunctions Node.

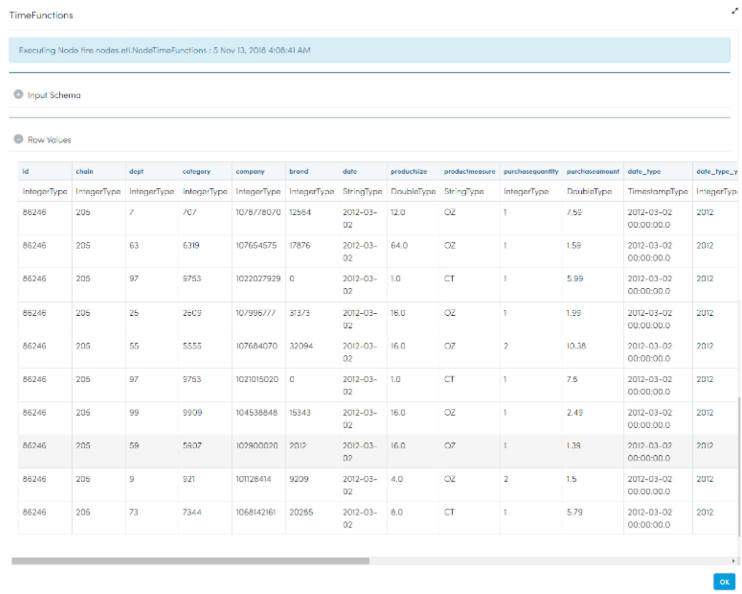
### Processor Configuration



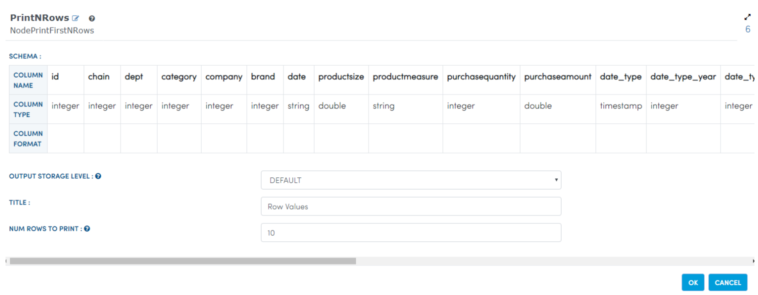

### Processor Output

### Prints the Results

It prints the results after using string to date timefunctions.




## Processor Configuration




## Processor Output

## Date-Time Field Extract

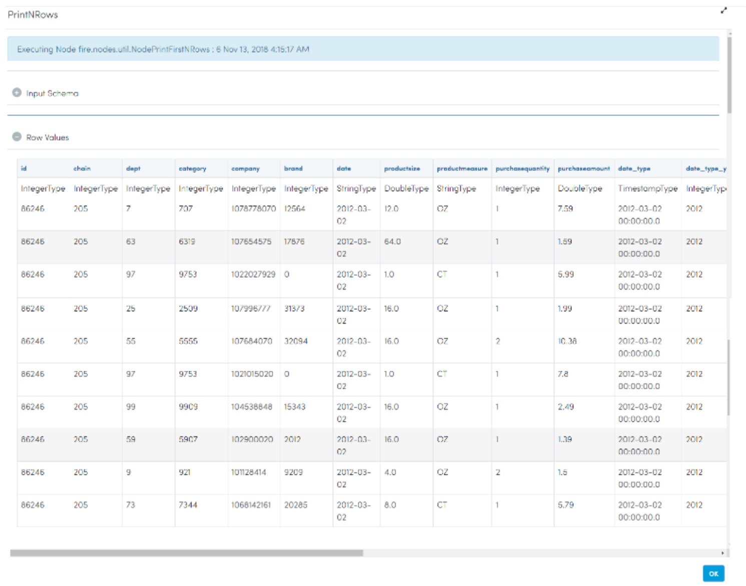
## Workflow

Below is the workflow. It does the following:

- Reads data from a dataset.
- It creates a new DataFrame by extracting Year, Month, Day of month, Hour, Minute, Second fields from "TimeStamp"
- Prints the results.

## Reading from Dataset

It reads in the input Dataset File.



PrintNRows

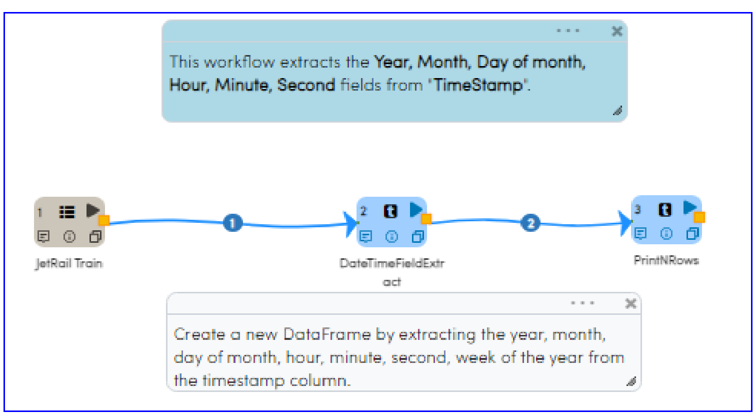
Executing Node fire.nodes.util.NodePrintFirstNRows : 6 Nov 13, 2018 4:15:17 AM

Input Schema

Row Values

| id | chain | dept | category | company | brand | date | productsize | productmeasure | purchasequantity | purchaseamount | date_type | date_type_y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | StringType | DoubleType | StringType | IntegerType | DoubleType | TimestampType | IntegerTyp |
| 86246 | 205 | 7 | 707 | 1078778070 | 12564 | 2012-03-02 | 12.0 | OZ | 1 | 7.59 | 2012-03-02 00:00:00.0 | 2012 |
| 86246 | 205 | 63 | 6319 | 107654575 | 17876 | 2012-03-02 | 64.0 | OZ | 1 | 1.59 | 2012-03-02 00:00:00.0 | 2012 |
| 86246 | 205 | 97 | 9753 | 1022027929 | 0 | 2012-03-02 | 1.0 | CT | 1 | 5.99 | 2012-03-02 00:00:00.0 | 2012 |
| 86246 | 205 | 25 | 2509 | 107996777 | 31373 | 2012-03-02 | 16.0 | OZ | 1 | 1.99 | 2012-03-02 00:00:00.0 | 2012 |
| 86246 | 205 | 55 | 5555 | 107684070 | 32094 | 2012-03-02 | 16.0 | OZ | 2 | 10.38 | 2012-03-02 00:00:00.0 | 2012 |
| 86246 | 205 | 97 | 9753 | 1021015020 | 0 | 2012-03-02 | 1.0 | CT | 1 | 7.8 | 2012-03-02 00:00:00.0 | 2012 |
| 86246 | 205 | 99 | 9909 | 104538848 | 15343 | 2012-03-02 | 16.0 | OZ | 1 | 2.49 | 2012-03-02 00:00:00.0 | 2012 |
| 86246 | 205 | 59 | 5907 | 102900020 | 2012 | 2012-03-02 | 16.0 | OZ | 1 | 1.39 | 2012-03-02 00:00:00.0 | 2012 |
| 86246 | 205 | 9 | 921 | 101128414 | 9209 | 2012-03-02 | 4.0 | OZ | 2 | 1.5 | 2012-03-02 00:00:00.0 | 2012 |
| 86246 | 205 | 73 | 7344 | 1068142161 | 20285 | 2012-03-02 | 8.0 | CT | 1 | 5.79 | 2012-03-02 00:00:00.0 | 2012 |

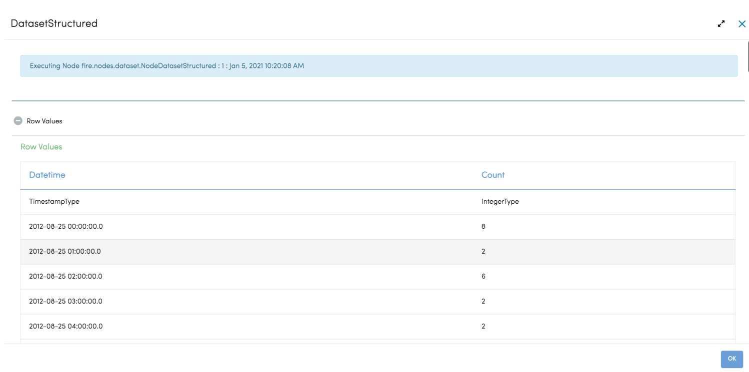
OK



### Processor Configuration

tutorials/data-engineering/../../_assets/tutorials/data-engineering/date-t

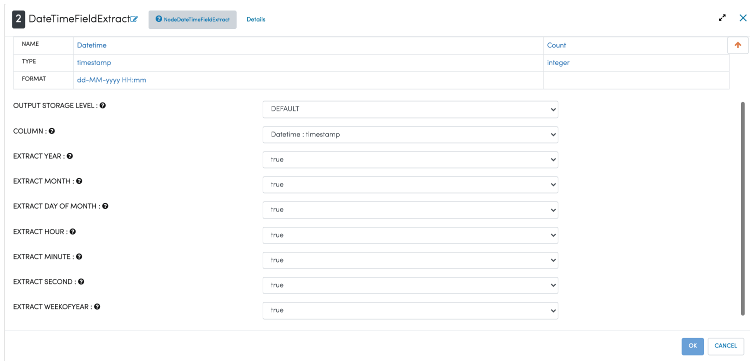
### Processor Output



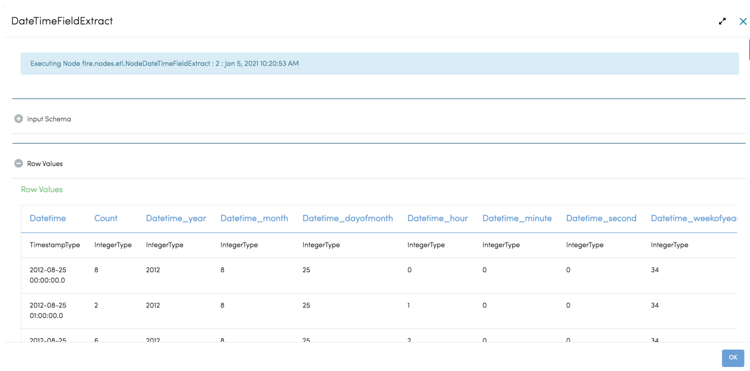

### Date-Time Field Extract

It creates a new DataFrame by extracting the year, month, day of month, hour, minute, second, week of the year from the timestamp column.

### Processor Configuration



## Processor Output



## Prints the Results

It prints the first few records onto the screen.

## Concat Columns

This example concats columns in the input dataset with the specified separator.

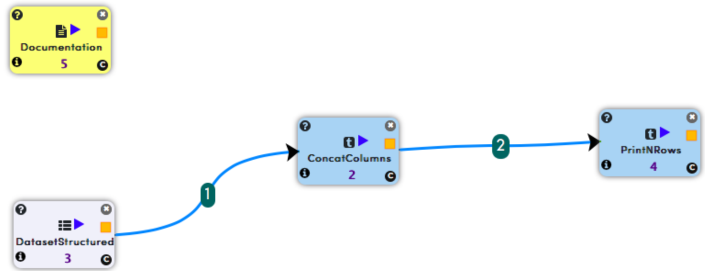
## Workflow

Below is the workflow. It does the following:

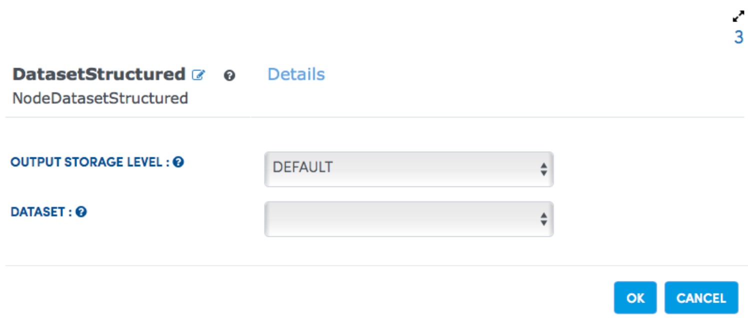
- Reads data from file present on HDFS.
- Concats the specified columns with specified separator.

## Reading from HDFS File

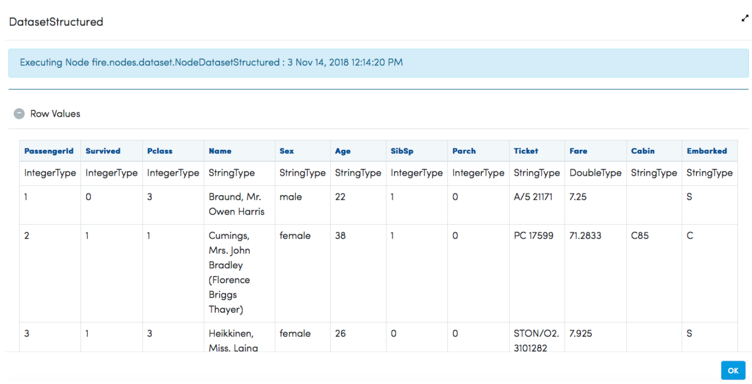
It reads data from a file present on HDFS.




### Processor Configuration




### Processor Output



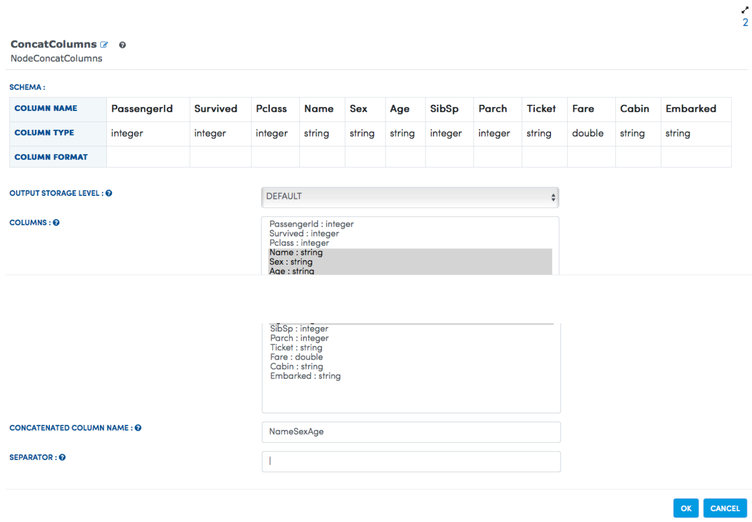

| PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | IntegerType | IntegerType | StringType | StringType | StringType | IntegerType | IntegerType | StringType | DoubleType | StringType | StringType |
| 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22 | 1 | 0 | A/5 21171 | 7.25 | | S |
| 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Thayer) | female | 38 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26 | 0 | 0 | STON/O2. 3101282 | 7.925 | | S |

### Concating columns

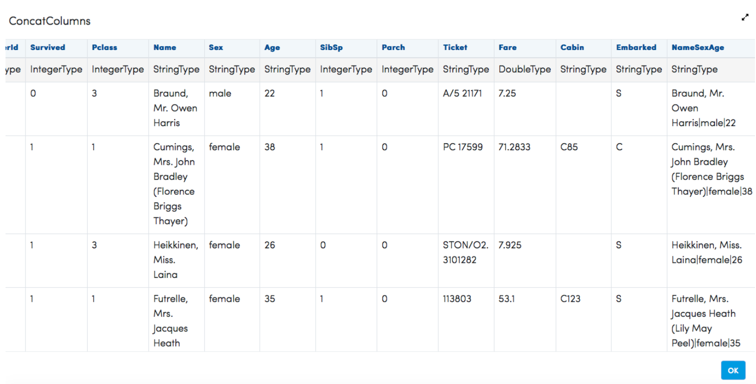
It concats the specified columns in cofiguration with the specified separator.

### Processor Configuration

We need to provide all the desired columns to be concatenated without any separator or space, like NameSexAge etc. Columns would get concatenated in same order defined in configuration like Name then Sex then Age.




## Processor Output


ConcatColumns

| rId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked | NameSexAge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ype | IntegerType | IntegerType | StringType | StringType | StringType | IntegerType | IntegerType | StringType | DoubleType | StringType | StringType | StringType |
| | 0 | 3 | Braund, Mr. Owen Harris | male | 22 | 1 | 0 | A/5 21171 | 7.25 | | S | Braund, Mr. Owen Harris\|male\|22 |
| | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Thayer) | female | 38 | 1 | 0 | PC 17599 | 71.2833 | C85 | C | Cumings, Mrs. John Bradley (Florence Briggs Thayer)\|female\|38 |
| | 1 | 3 | Heikkinen, Miss. Laina | female | 26 | 0 | 0 | STON/O2. 3101282 | 7.925 | | S | Heikkinen, Miss. Laina\|female\|26 |
| | 1 | 1 | Futrelle, Mrs. Jacques Heath | female | 35 | 1 | 0 | 113803 | 53.1 | C123 | S | Futrelle, Mrs. Jacques Heath (Lily May Peel)\|female\|35 |

OK

## Joining Multiple Datasets

Fire Insights allows you to quickly do complex data preparation and ETL on Big Data.

Fire Insights has a number of features for enabling it including:

- Reading data from multiple sources
- Cleaning data
- oins, GroupBy, Cube, SQL etc. to transform data
- Writing results to various sinks

Fire Insights also gives you detailed control over your Spark jobs with Repartition, Coalesce, Cache etc.

#### Overview

In this example, we start with 5 datasets, read them in & understand their schema in the process, perform data cleaning and then apply appropriate aggregations and joins.

The cleansed and tranformed datasets are written to HDFS as CSV files. These dataset can as well we written as Parquet, Avro, JSON, XML files or to HIVE/Relational tables as needed.

#### Datasets

- facts.dat : Contains fixed length records of products sold to customers
- geo.csv : Contains mapping of geo ids to geo names
- product.csv : Contains mapping of product ids to product names
- customer.csv : Contains mapping of customer ids to customer names
- time.csv : Contains mapping of various time interval ids to corresponding names

#### Workflow

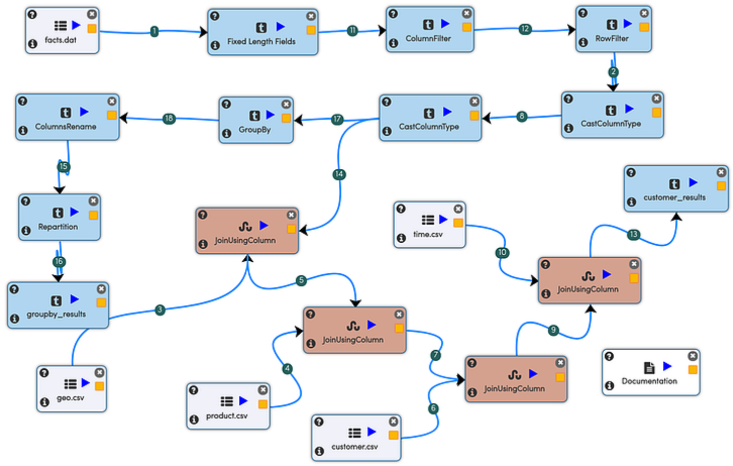
The workflow achieves the following tasks:

- Parses the facts data and performs various cleanup operations on it.
- Performs groupby with aggregations operations and saves it to a file.
- Joins the fact data with various dimensions to create a large table and saves it to a CSV file.

The workflow is shown below:

#### Data Parsing and Cleaning

While the various dimension data is available as CSV files, the fact data is in fixed field size format.




Each record has a fixed number of characters. In each record each field consists of fixed number of characters. The steps for data parsing and cleaning are as follows:

- Read in the fixed length record
- Filter out invalid records
- Cast some columns to numeric values

## Group By and Aggregates

The data is then aggregated and counted and averages are calculated. It is then saved as CSV file.

## Joins with various Dimension Data

The fact data is then joined with various dimension data. These include:

- Geo
- Product
- Customer
- Time

The final dataset is saved as CSV file.

## Time Function

There are many instances when you want to do time-series analysis. Fire Insights provides Date-Time features with TimeFunctions operator.

Creating additional features from the timestamp column helps you to know more about the data and run modeling algorithms on them. Fire Insights has NodeTimeFunctions for creating these time series features.

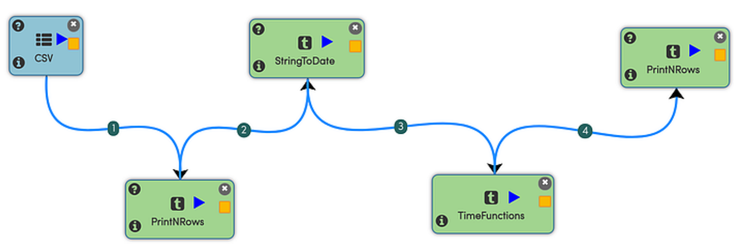
### Dataset

Let us take a Transaction Dataset which is in CSV format on HDFS. The dataset has a "DATE" column.

| ID | CHAIN | DEPT | CATEGORY | COMPANY | BRAND | DATE | PRODUCTSIZE | PRODUCT |
|---|---|---|---|---|---|---|---|---|
| IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | StringType | IntegerType | StringType |
| 86246 | 205 | 7 | 707 | 1078778070 | 12564 | 2012-03-02 | 12 | OZ |
| 86246 | 205 | 63 | 6319 | 107654575 | 17876 | 2012-03-02 | 64 | OZ |
| 86246 | 205 | 97 | 9753 | 1022027929 | 0 | 2012-03-02 | 1 | CT |
| 86246 | 205 | 25 | 2509 | 107996777 | 31373 | 2012-03-02 | 16 | OZ |
| 86246 | 205 | 55 | 5555 | 107684070 | 32094 | 2012-03-02 | 16 | OZ |
| 86246 | 205 | 97 | 9753 | 1021015020 | 0 | 2012-03-02 | 1 | CT |

### Workflow for applying TimeFunctions

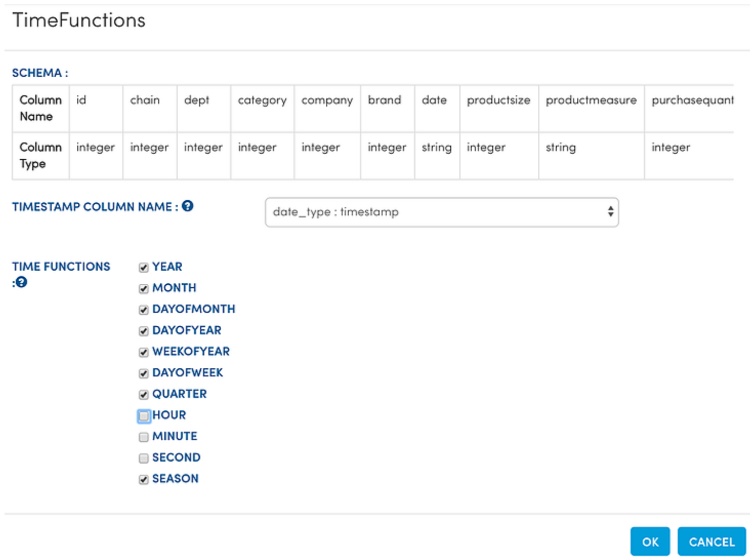
In the example workflow below, additional date time features are being created from the date column.




In the above workflow:

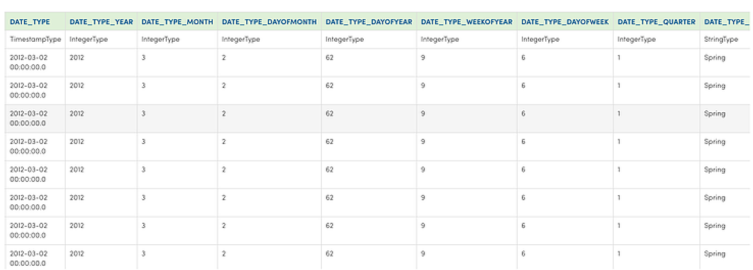
- The 'CSV' processor reads in the CSV data from HDFS.
- The 'StringToDate' processor converts the column Date, which is in string format to 'timestamp'.
- The 'TimeFunctions' processor takes in the timestamp column and then applies various timefunctions to it to generate additional output columns.

The diagram below shows the dialog box for the TimeFunctions processor. Timestamp column was selected as input, and various time functions were applied to it.




## Workflow Execution

When the example workflow is executed, additional columns are produced for the various time functions that were selected.



| DATE_TYPE | DATE_TYPE_YEAR | DATE_TYPE_MONTH | DATE_TYPE_DAYOFMONTH | DATE_TYPE_DAYOFYEAR | DATE_TYPE_WEEKOFYEAR | DATE_TYPE_DAYOFWEEK | DATE_TYPE_QUARTER | DATE_TYPE_ |
|---|---|---|---|---|---|---|---|---|
| TimestampType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | StringType |
| 2012-03-02 00:00:00.0 | 2012 | 3 | 2 | 62 | 9 | 6 | 1 | Spring |
| 2012-03-02 00:00:00.0 | 2012 | 3 | 2 | 62 | 9 | 6 | 1 | Spring |
| 2012-03-02 00:00:00.0 | 2012 | 3 | 2 | 62 | 9 | 6 | 1 | Spring |
| 2012-03-02 00:00:00.0 | 2012 | 3 | 2 | 62 | 9 | 6 | 1 | Spring |
| 2012-03-02 00:00:00.0 | 2012 | 3 | 2 | 62 | 9 | 6 | 1 | Spring |
| 2012-03-02 00:00:00.0 | 2012 | 3 | 2 | 62 | 9 | 6 | 1 | Spring |
| 2012-03-02 00:00:00.0 | 2012 | 3 | 2 | 62 | 9 | 6 | 1 | Spring |
| 2012-03-02 00:00:00.0 | 2012 | 3 | 2 | 62 | 9 | 6 | 1 | Spring |

## Split Dataset By Expression

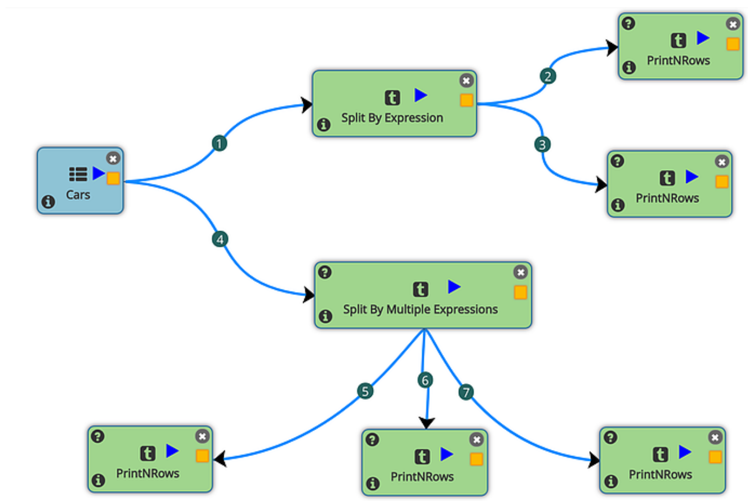
Fire Insights allows you to split incoming dataframes. Based on your needs, use the processors described below:

- 'SplitByExpression': This processor splits the incoming dataset based on an expression. Rows satisfying the expression go into one dataframe and the rest go into another dataframe.
- 'SplitByMultipleExpressions': This processor splits the incoming dataset into multiple

dataframes based on up to five conditional expressions.The output of each expression is routed to a separate output path.

- 'Split': This processor splits the incoming dataframe into two based on the percentage specified for the split. Split processor is especially useful in machine learning workflows.

### Workflow



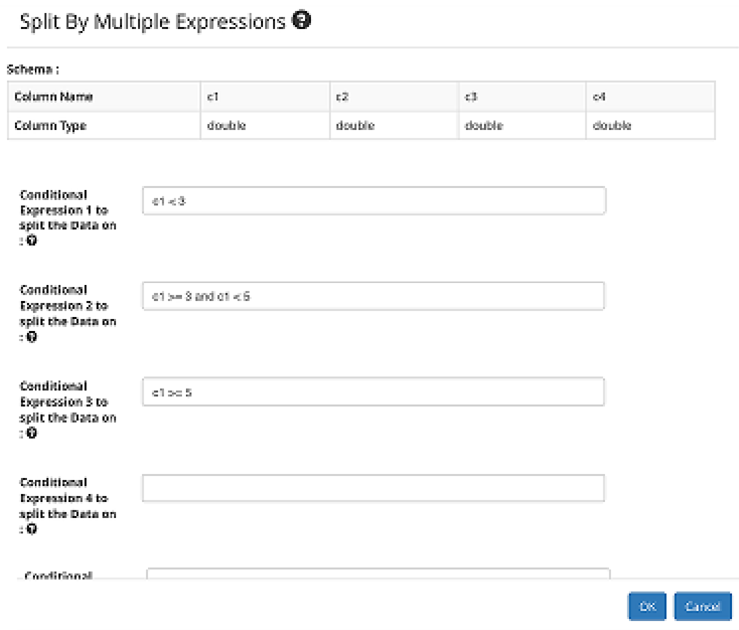

In the example workflow above, 'Split By Multiple Expressions' processor splits the incoming dataframe into three output dataframes. The three conditions are on column c1 - "c1<3" , "c1>=3 and c1<5", and "c1>=5". As mentioned earlier, 'SplitByMultipleExpressions' can split incoming dataframe into up to five dataframes.

### Output

For the example workflow, the three output dataframes are shown below:

### String Functions

String functions are useful to tranform strings in your dataframe. The "StringFunction" processors allows you to apply common string operations such as 'trim', 'upper', 'lower', 'lefttrim', 'righttrim' etc. to strings.




| c1 | c2 | c3 | c4 |
|---|---|---|---|
| DoubleType | DoubleType | DoubleType | DoubleType |
| 1.0 | 0.0 | 2.3 | 3.0 |
| 2.0 | 1.0 | 3.0 | 2.0 |
| 1.0 | 0.0 | 2.3 | 3.0 |
| 2.0 | 1.0 | 3.0 | 2.0 |
| 1.0 | 0.0 | 2.3 | 3.0 |
| 2.0 | 1.0 | 3.0 | 2.0 |

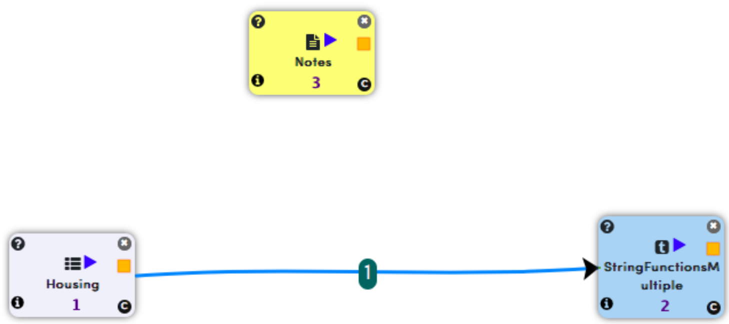
| c1 | c2 | c3 | c4 |
|---|---|---|---|
| DoubleType | DoubleType | DoubleType | DoubleType |
| 3.0 | 0.0 | 1.1 | 1.0 |
| 4.0 | 0.0 | 4.1 | 5.0 |
| 3.0 | 0.0 | 1.1 | 1.0 |
| 4.0 | 0.0 | 4.1 | 5.0 |
| 3.0 | 0.0 | 1.1 | 1.0 |
| 4.0 | 0.0 | 4.1 | 5.0 |

| c1 | c2 | c3 | c4 |
|---|---|---|---|
| DoubleType | DoubleType | DoubleType | DoubleType |
| 5.0 | 0.0 | 3.1 | 6.0 |
| 6.0 | 1.0 | 2.1 | 2.0 |
| 5.0 | 0.0 | 3.1 | 6.0 |
| 6.0 | 1.0 | 2.1 | 2.0 |
| 5.0 | 0.0 | 3.1 | 6.0 |
| 6.0 | 1.0 | 2.1 | 2.0 |

In the example below, different string functions are applied to input dataset.

### Workflow

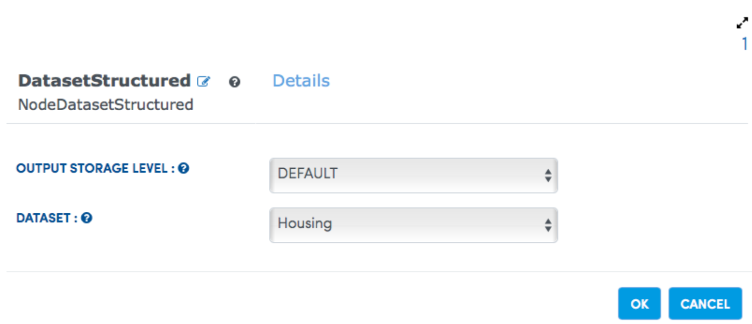
The example workflow below, read data from HDFS/Hive and applies different string functions on different columns of the dataset.




### Read data from HDFS

The "Housing" processor above, reads an existing dataset on HDFS.

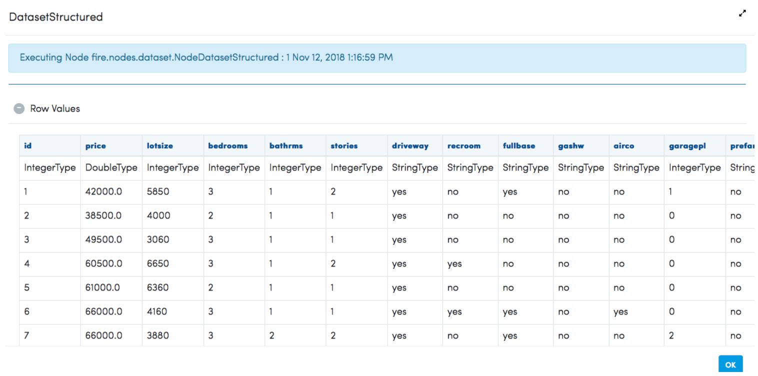
### Processor Configuration




### Processor Output

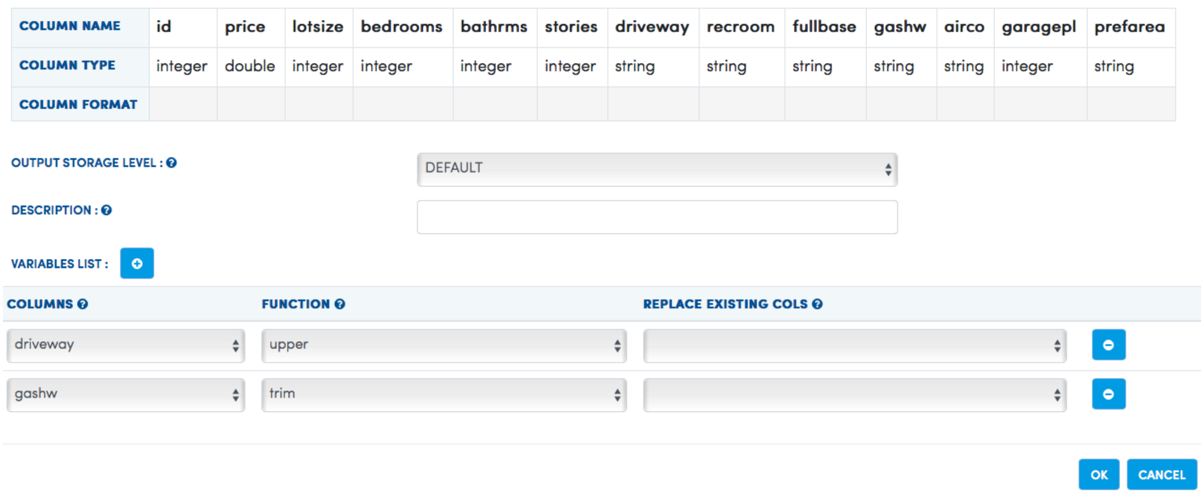
### Apply string functions

The 'StringFunctionMultiple' processor below, converts contents of 'driveway' column to upper case and trims contents of 'gashw' column.




| id | price | lotsize | bedrooms | bathrms | stories | driveway | recroom | fullbase | gashw | airco | garagepl | prefar |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | DoubleType | IntegerType | IntegerType | IntegerType | IntegerType | StringType | StringType | StringType | StringType | StringType | IntegerType | String |
| 1 | 42000.0 | 5850 | 3 | 1 | 2 | yes | no | yes | no | no | 1 | no |
| 2 | 38500.0 | 4000 | 2 | 1 | 1 | yes | no | no | no | no | 0 | no |
| 3 | 49500.0 | 3060 | 3 | 1 | 1 | yes | no | no | no | no | 0 | no |
| 4 | 60500.0 | 6650 | 3 | 1 | 2 | yes | yes | no | no | no | 0 | no |
| 5 | 61000.0 | 6360 | 2 | 1 | 1 | yes | no | no | no | no | 0 | no |
| 6 | 66000.0 | 4160 | 3 | 1 | 1 | yes | yes | yes | no | yes | 0 | no |
| 7 | 66000.0 | 3880 | 3 | 2 | 2 | yes | no | yes | no | no | 2 | no |

## Processor Configuration




| COLUMN NAME | id | price | lotsize | bedrooms | bathrms | stories | driveway | recroom | fullbase | gashw | airco | garagepl | prefarea |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COLUMN TYPE | integer | double | integer | integer | integer | integer | string | string | string | string | string | integer | string |
| COLUMN FORMAT | | | | | | | | | | | | | |

| COLUMNS | FUNCTION | REPLACE EXISTING COLS |
|---|---|---|
| driveway | upper | |
| gashw | trim | |

## Processor Output

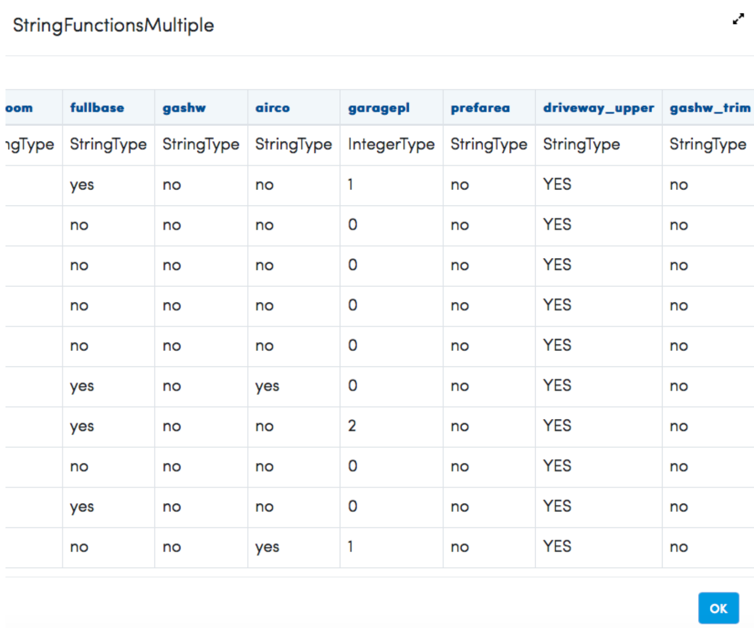
## Data Preparation-1

Data preparation is the process of cleaning and transforming raw data prior to processing and analysis. It is an important step prior to processing and often involves reformatting data, making corrections to data and the combining of data sets to enrich data.

## Workflow

Below is the workflow. It does the following:

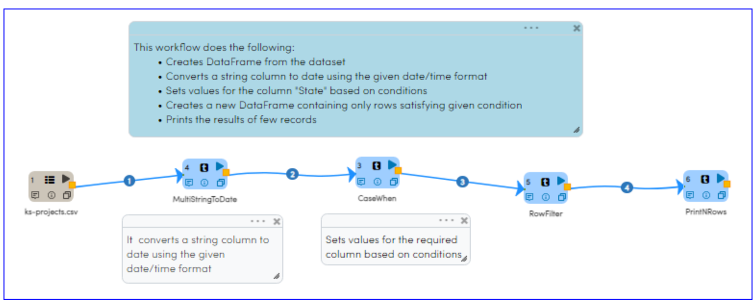
- Reads data from the dataset
- converts a string column to date using the given date/time format


StringFunctionsMultiple

| oom | fullbase | gashw | airco | garagepl | prefarea | driveway_upper | gashw_trim |
|---|---|---|---|---|---|---|---|
| ngType | StringType | StringType | StringType | IntegerType | StringType | StringType | StringType |
| | yes | no | no | 1 | no | YES | no |
| | no | no | no | 0 | no | YES | no |
| | no | no | no | 0 | no | YES | no |
| | no | no | no | 0 | no | YES | no |
| | no | no | no | 0 | no | YES | no |
| | yes | no | yes | 0 | no | YES | no |
| | yes | no | no | 2 | no | YES | no |
| | no | no | no | 0 | no | YES | no |
| | yes | no | no | 0 | no | YES | no |
| | no | no | yes | 1 | no | YES | no |

OK

- Sets values for the column “State” based on conditions
- Creates a new DataFrame containing only rows satisfying given condition
- Prints the results of few records




### Reading from Dataset

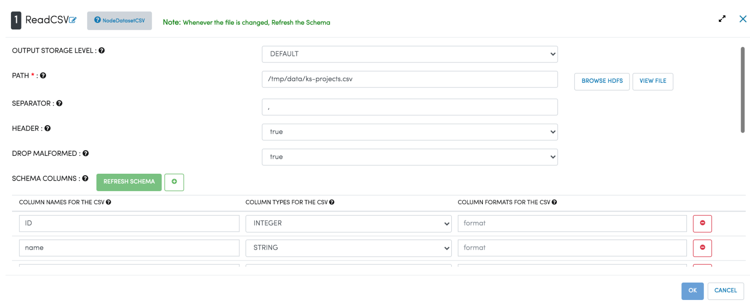
It reads in the input Dataset File.

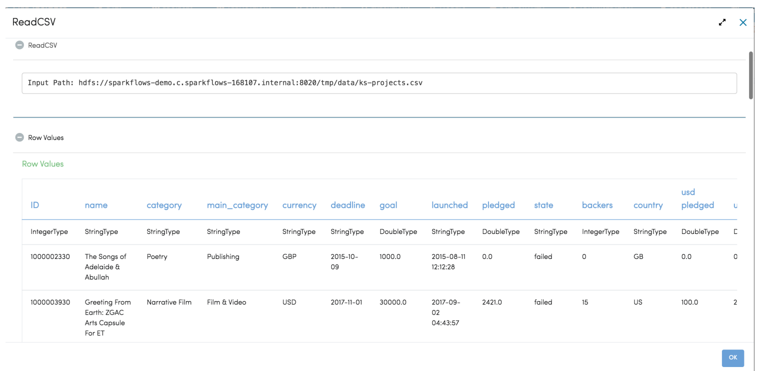
### Processor Configuration

### Processor Output

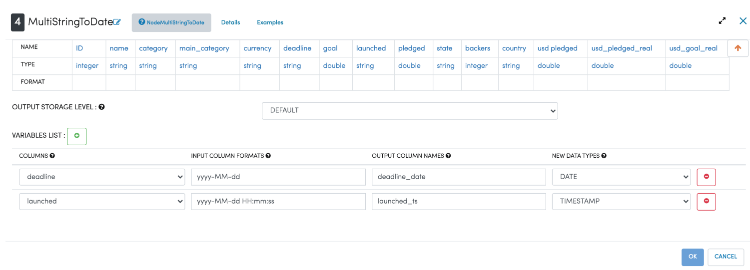
### Convert String to Date

`MultiStringToDate` converts a string column to date using the given date/time format.




## Processor Configuration



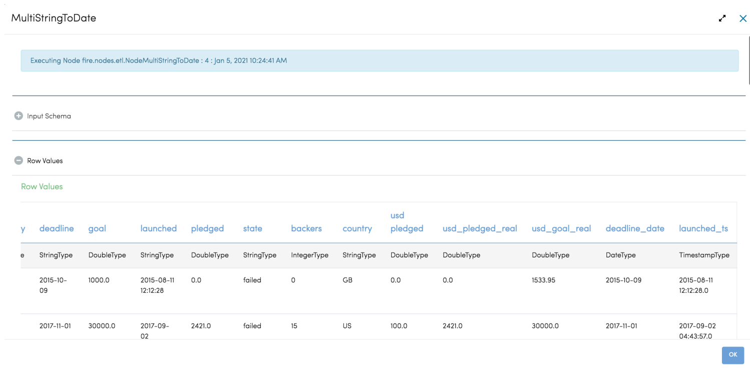

## Processor Output

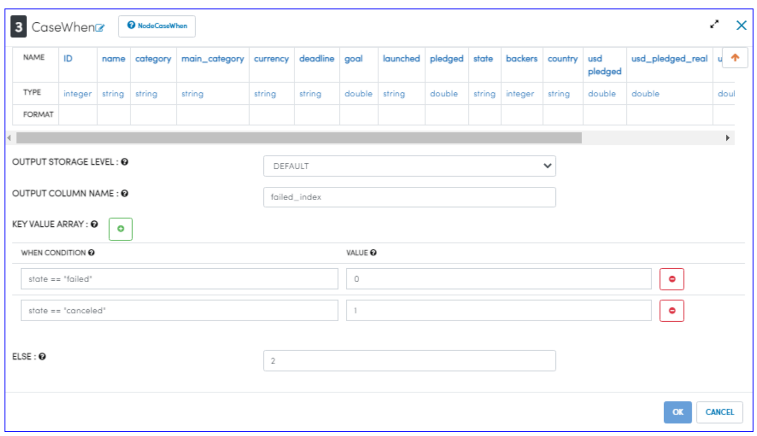
## Settings values for required Column

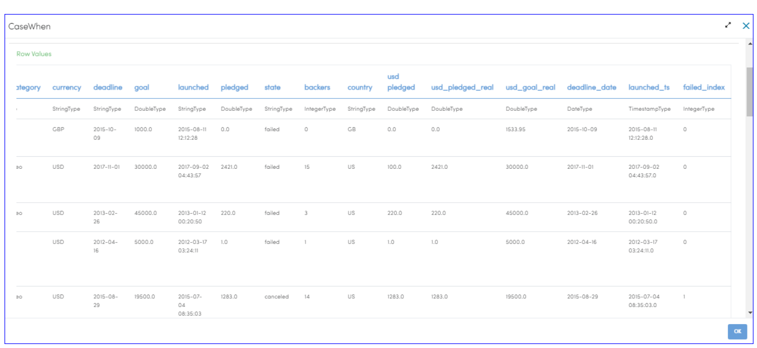
`CaseWhen` sets values for the required column based on conditions as shown in example below:

## Processor Configuration

## Processor Output



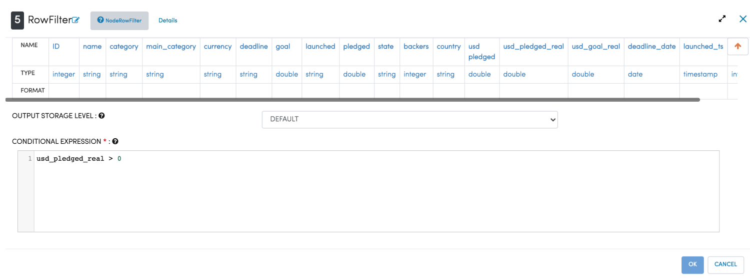




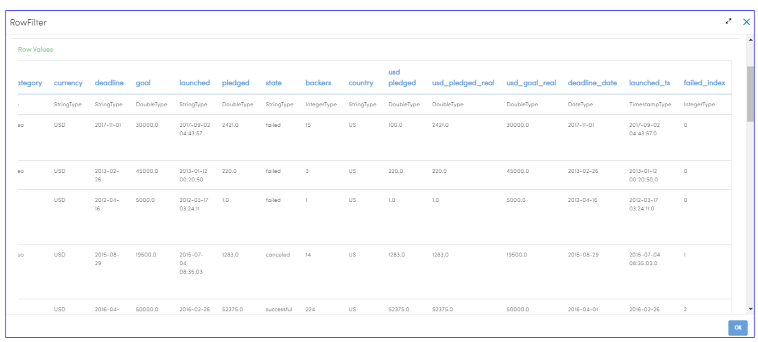
### Creating DataFrame with required rows

`RowFilter` creates a new DataFrame containing only rows required.

### Processor Configuration



### Processor Output



### Prints the Results

It prints the first few records onto the screen.

### Data Cleaning

This workflow cleans the input data. It does the following:

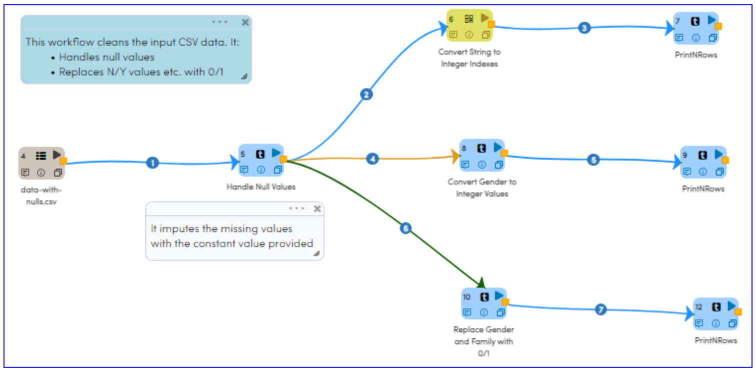
- Handles null values
- Replaces N/Y values etc. with 0/1

### Workflow

Below is the workflow. It does the following:

- Reads data from a dataset

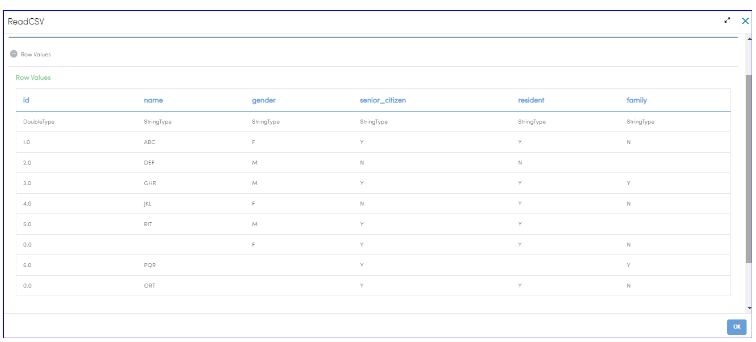
- Handles the null values by imputing the missing values with the constant value provided in the specified columns
- Convert Strings to Integer Indexes
- Convert Gender to Integer Values
- Replace Gender and Family with 0/1




### Reading from Dataset

`DatasetCSV` reads in the input Dataset file and creates DataFrame from it.

### Processor Output

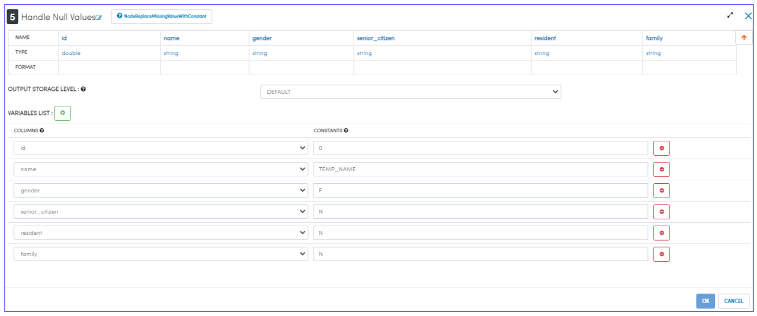



| id | name | gender | senior_citizen | resident | family |
|---|---|---|---|---|---|
| DoubleType | StringType | StringType | StringType | StringType | StringType |
| 1.0 | ABC | F | Y | Y | N |
| 2.0 | DEF | M | N | N | |
| 3.0 | GHR | M | Y | Y | Y |
| 4.0 | JKL | F | N | Y | N |
| 5.0 | RIT | M | Y | Y | |
| 0.0 | | F | Y | Y | N |
| 6.0 | PQR | | Y | | Y |
| 0.0 | ORT | | Y | Y | N |

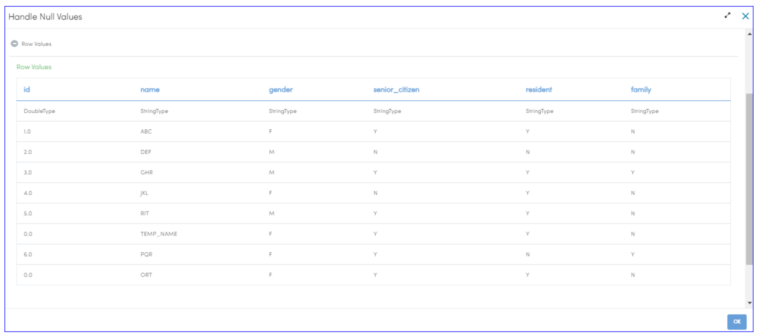
### Handling Null Values

`ReplaceMissingValueWithConstant` processor handles the null values by imputing the missing values with the constant value provided in the specified columns.

### Processor Configuration



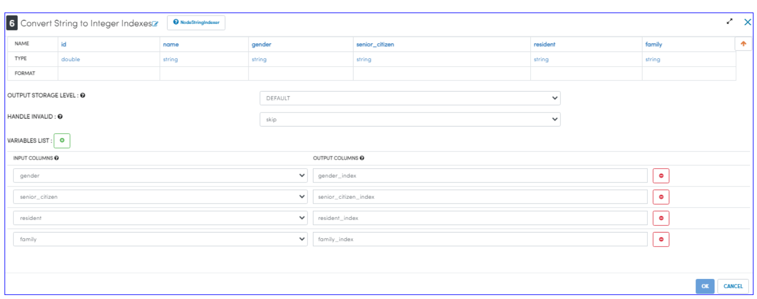
### Processor Output



### Convert Strings to Integer Indexes

`StringIndexer` processor encodes a string type column to a column of label indices.

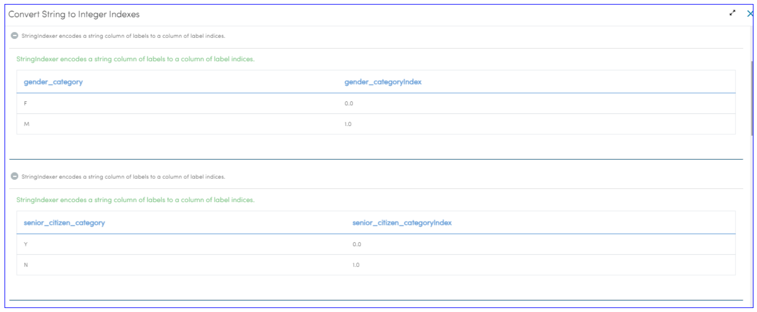
### Processor Configuration

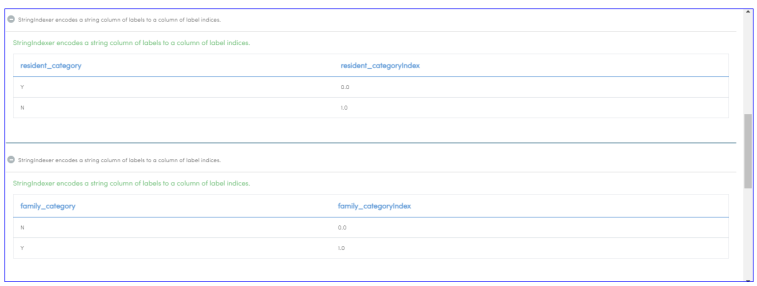


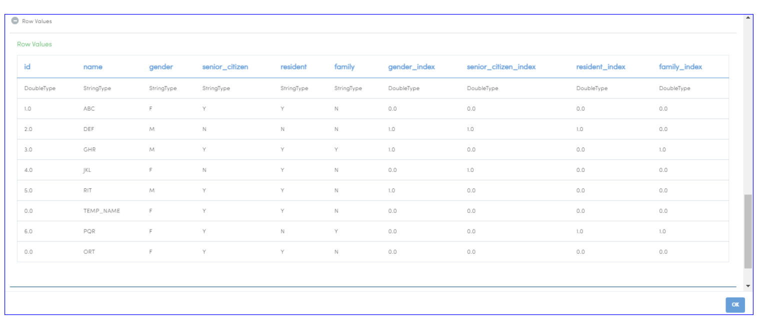
### Processor Output

### Convert Gender to Integer Values

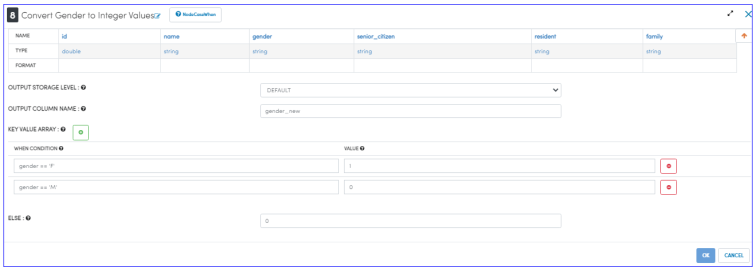
`CaseWhen` processor sets values for the variables based on conditions, as shown below:


Convert String to Integer Indexes
StringIndexer encodes a string column of labels to a column of label indices.
StringIndexer encodes a string column of labels to a column of label indices.
gender_category gender_categoryIndex
F 0.0
M 1.0
StringIndexer encodes a string column of labels to a column of label indices.
StringIndexer encodes a string column of labels to a column of label indices.
senior_citizen_category senior_citizen_categoryIndex
Y 0.0
N 1.0

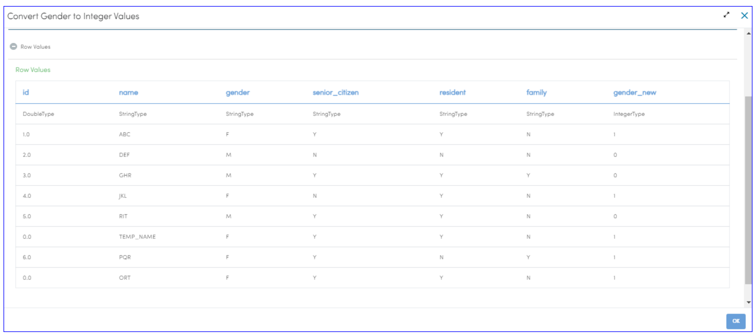

StringIndexer encodes a string column of labels to a column of label indices.
StringIndexer encodes a string column of labels to a column of label indices.
resident_category resident_categoryIndex
Y 0.0
N 1.0
StringIndexer encodes a string column of labels to a column of label indices.
StringIndexer encodes a string column of labels to a column of label indices.
family_category family_categoryIndex
N 0.0
Y 1.0


Row Values
Row Values
id name gender senior_citizen resident family gender_index senior_citizen_index resident_index family_index
DoubleType StringType StringType StringType StringType StringType DoubleType DoubleType DoubleType DoubleType
1.0 ABC F Y Y N 0.0 0.0 0.0 0.0
2.0 DEF M N N N 1.0 1.0 1.0 0.0
3.0 GHR M Y Y Y 1.0 0.0 0.0 1.0
4.0 JKL F N Y N 0.0 1.0 0.0 0.0
5.0 RIT M Y Y N 1.0 0.0 0.0 0.0
0.0 TEMP_NAME F Y Y N 0.0 0.0 0.0 0.0
6.0 PQR F Y N Y 0.0 0.0 1.0 1.0
0.0 ORT F Y Y N 0.0 0.0 0.0 0.0
OK

### Processor Configuration



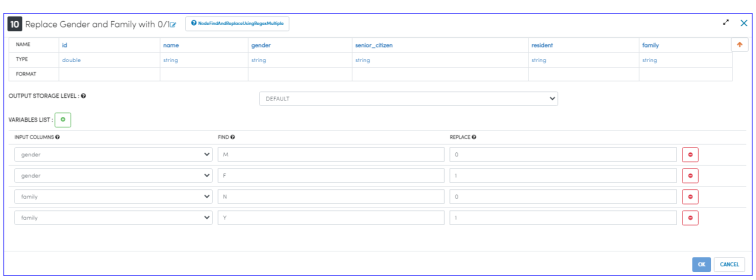
### Processor Output



## Replace Gender and Family with 0/1

`FindAndReplaceUsingRegexMultiple` processor sets values for the variables based on conditions, as shown below:

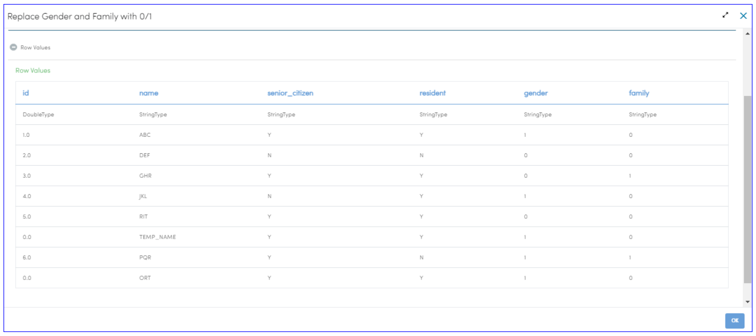
### Processor Configuration



### Processor Output

## Prints the Results

It prints the first few records onto the screen.



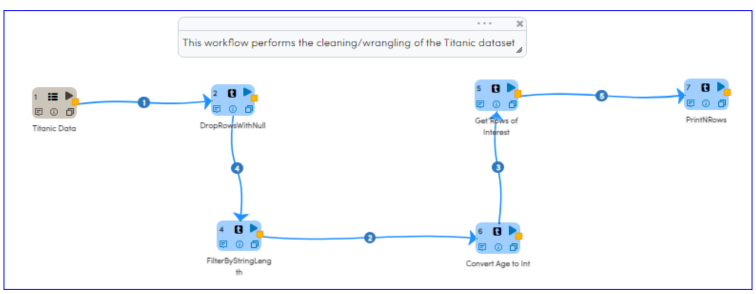

## Titanic Data Cleaning/Wrangling

This workflow shows how to wrangle the Titanic Dataset with Sparkflows.

### Workflow

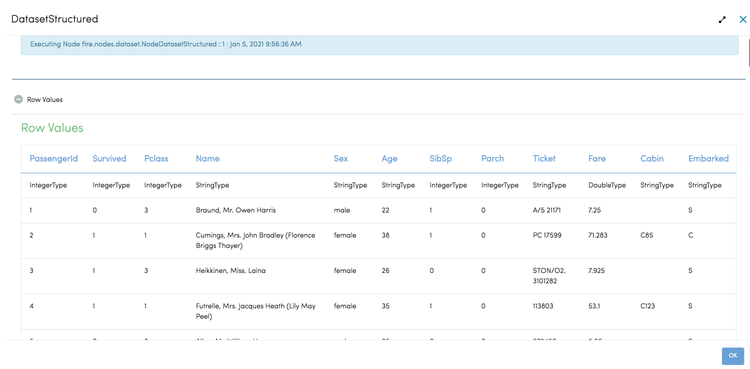
This workflow performs the following steps:

- Reads the Titanic dataset
- Drops Rows containing Null values
- Filters the Rows for whom Age has not been specified
- Changes the data type of the Age column to integer
- Filters rows for persons of age > 30 and who are female




### Reading Titanic dataset

`DatasetStructured` processor creates a Dataframe of your dataset named `Titanic Data` by reading data from HDFS, HIVE etc. which had been defined earlier in Fire by using the Dataset feature.



DatasetStructured

Executing Node fire.nodes.dataset.NodeDatasetStructured : 1 : Jan 5, 2021 9:56:36 AM

Row Values

Row Values

| PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | IntegerType | IntegerType | StringType | StringType | StringType | IntegerType | IntegerType | StringType | DoubleType | StringType | StringType |
| 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22 | 1 | 0 | A/5 21171 | 7.25 | | S |
| 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Thayer) | female | 38 | 1 | 0 | PC 17599 | 71.283 | C85 | C |
| 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26 | 0 | 0 | STON/O2. 3101282 | 7.925 | | S |
| 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35 | 1 | 0 | 113803 | 53.1 | C123 | S |

OK

## Processor Output

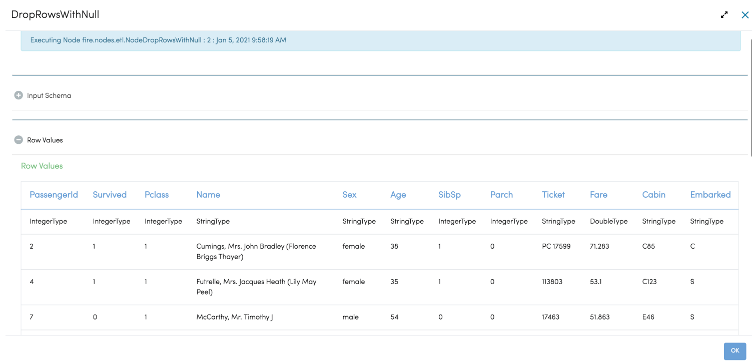
## Dropping the rows with null values

`DropRowsWithNull` processor drops the rows with null values.

## Processor Configuration

tutorials/data-engineering/../../_assets/tutorials/data-engineering/titani

## Processor Output

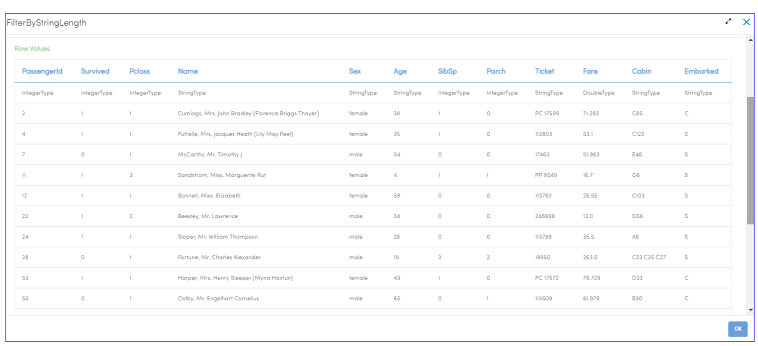


### Filter by string length

`FilterByStringLength` processor filters the rows within the provided string length

### Processor Configuration

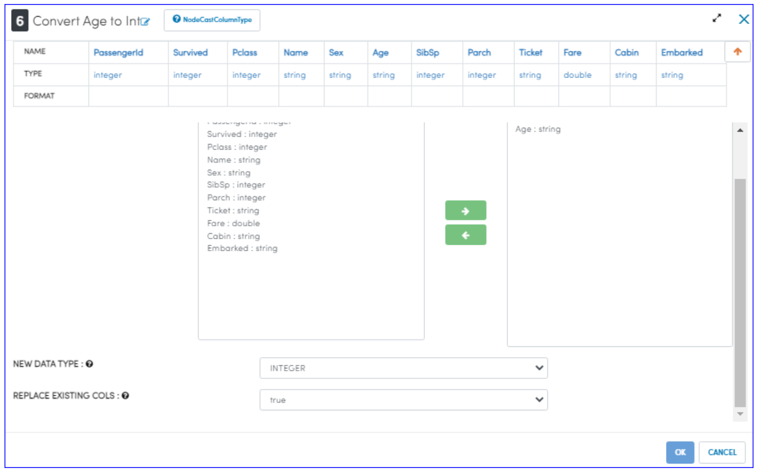
### Processor Output



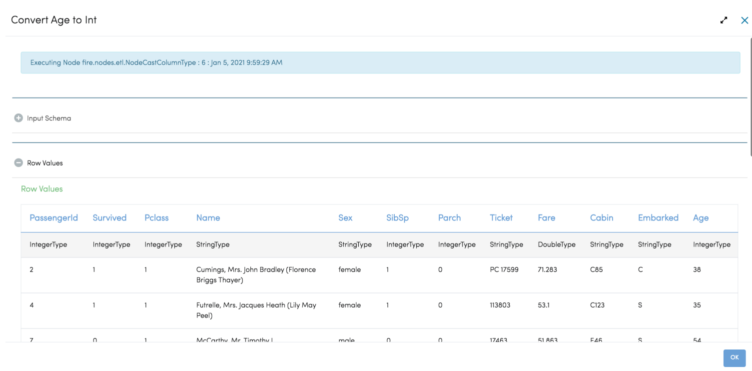
### Convert Age to Integer

`CastColumnType` processor performs conversion of Age to integer type.

### Processor Configuration




### Processor Output




### Get Rows of Interest

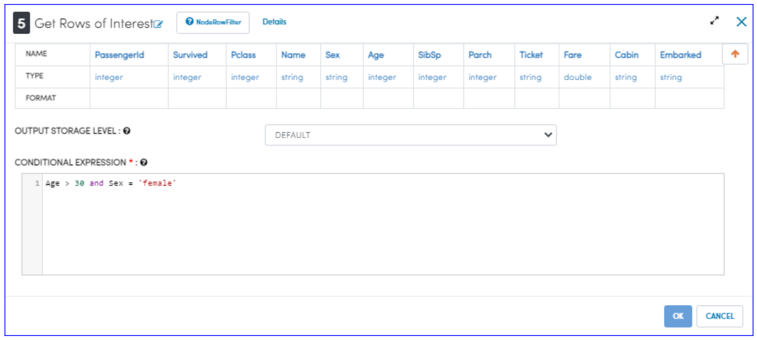
RowFilter processor filters the data based on provided conditions as shown below:

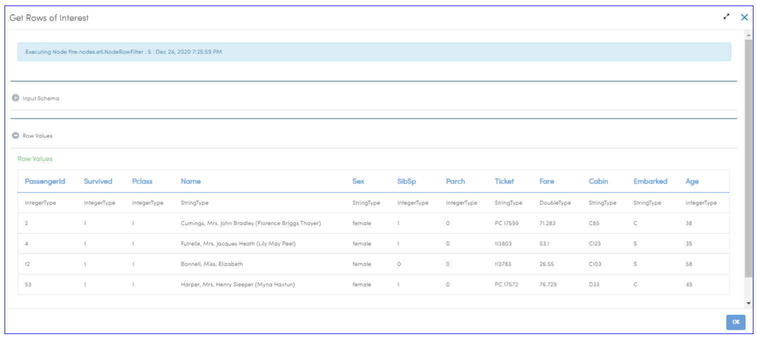
### Processor Configuration

### Processor Output

### Prints the results

It prints the first few records onto the screen.



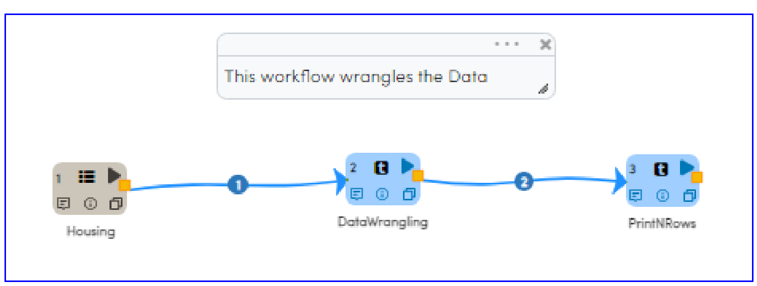

## Data Wrangling

Data wrangling is the process of gathering, selecting, and transforming data to answer an analytical question. Also known as data cleaning or "munging". This workflow reads in a dataset. It then wrangles the dataset based on provided conditions and prints the results.

## Workflow

Below is the workflow. It does the following:

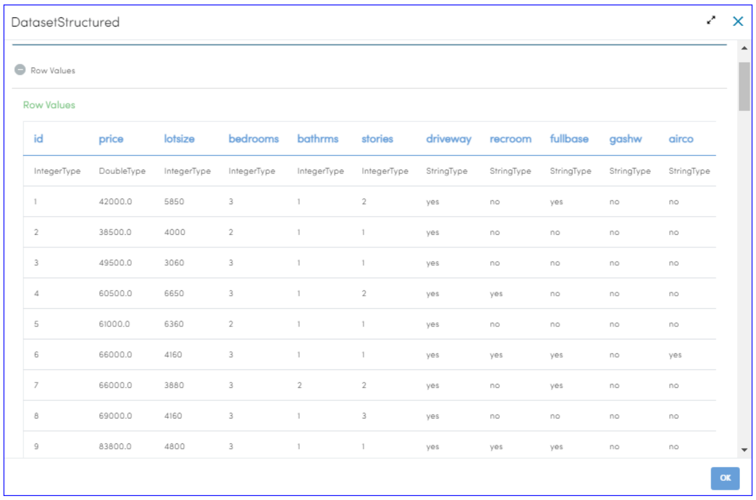
- Reads data from a dataset
- It then create new DataFrame based on the rules provided
- Prints the results



### Reading from Dataset

`DatasetStructured` processor creates a Dataframe of your dataset by reading data from HDFS, HIVE etc. which had been defined earlier in Fire by using the Dataset feature.

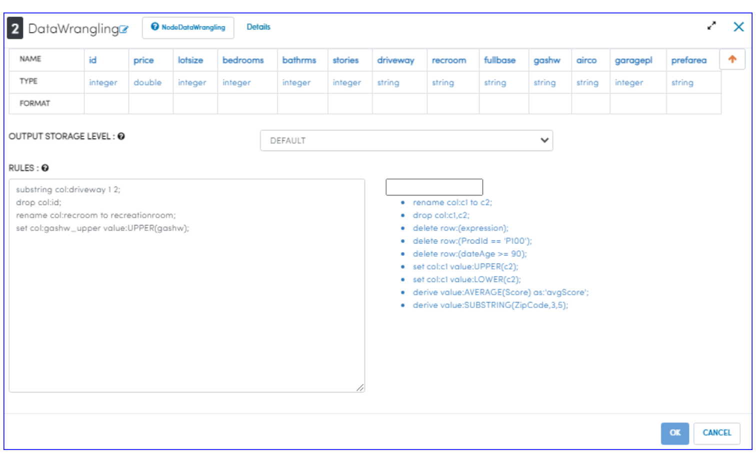
### Processor Output



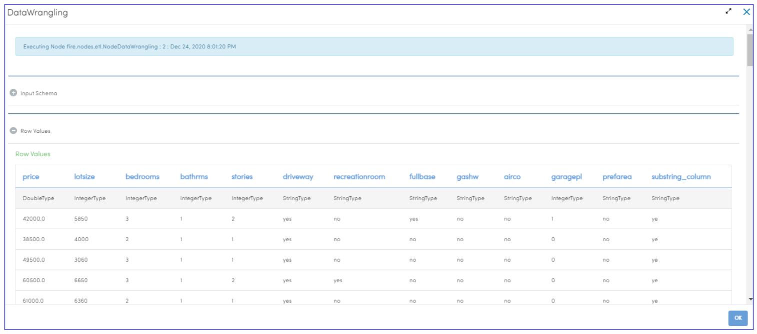

### Data Wrangling

`DataWrangling` processor creates new DataFrame after applying the provided rules

### Processor Configuration



### Processor Output




### Prints the Results

It prints the first few records onto the screen.

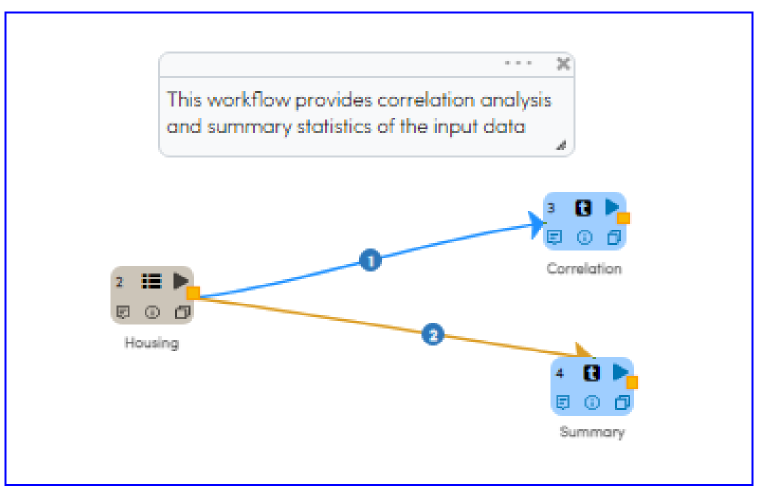
### Profiling-Correlation

This workflow reads in a dataset. It then creates the correlation analysis and summary statistics.

### Workflow

Below is the workflow. It does the following:

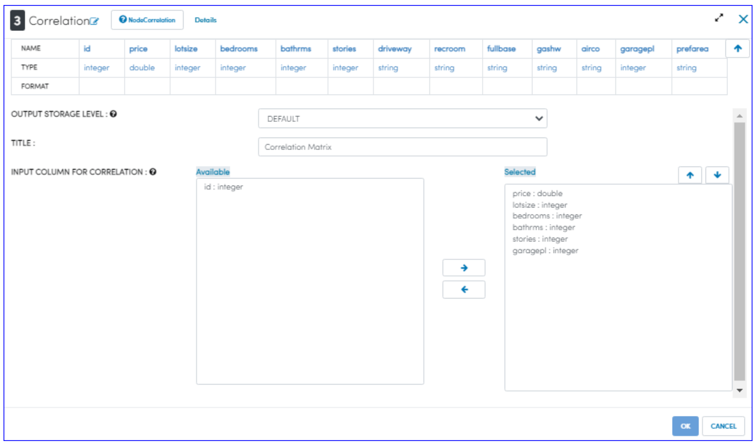
- Reads data from a dataset.
- Perform correlation analysis of the required columns
- Provide summary statistics of the dataset



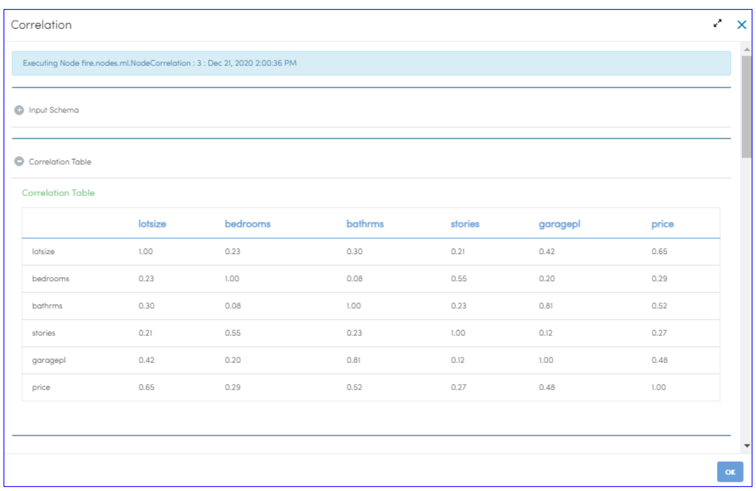
### Performing Correlation analysis

`Correlation` processor performs correlation analysis on the selected columns as shown below:

### Processor Configuration




### Processor Output - Correlation matrix

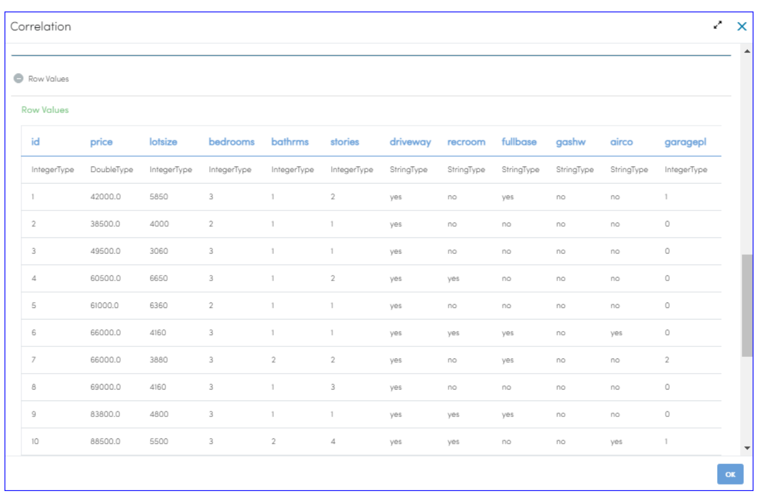



### Processor Output - Correlation Matrix Heat Map

### Processor Output - Sample Rows of Input Dataset

### Summary Statistics

`Summary` processor provides summary statistics of the input dataset.

Correlation

Row Values

Row Values

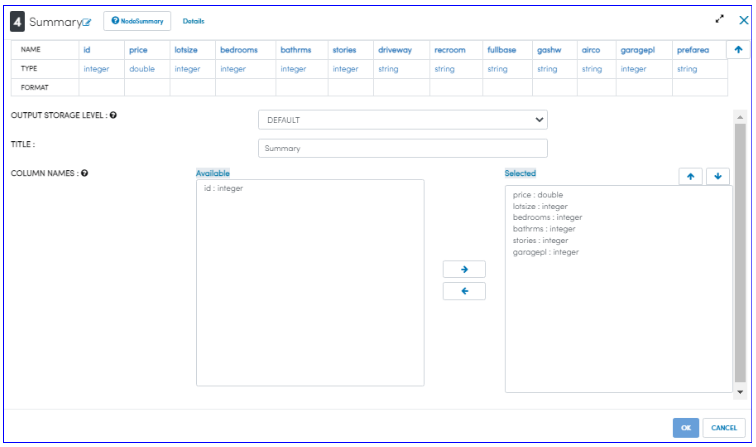
| id | price | lotsize | bedrooms | bathrms | stories | driveway | recroom | fullbase | gashw | airco | garagepl |
|---|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | DoubleType | IntegerType | IntegerType | IntegerType | IntegerType | StringType | StringType | StringType | StringType | StringType | IntegerType |
| 1 | 42000.0 | 5850 | 3 | 1 | 2 | yes | no | yes | no | no | 1 |
| 2 | 38500.0 | 4000 | 2 | 1 | 1 | yes | no | no | no | no | 0 |
| 3 | 49500.0 | 3060 | 3 | 1 | 1 | yes | no | no | no | no | 0 |
| 4 | 60500.0 | 6650 | 3 | 1 | 2 | yes | yes | no | no | no | 0 |
| 5 | 61000.0 | 6360 | 2 | 1 | 1 | yes | no | no | no | no | 0 |
| 6 | 66000.0 | 4160 | 3 | 1 | 1 | yes | yes | yes | no | yes | 0 |
| 7 | 66000.0 | 3880 | 3 | 2 | 2 | yes | no | yes | no | no | 2 |
| 8 | 69000.0 | 4160 | 3 | 1 | 3 | yes | no | no | no | no | 0 |
| 9 | 83800.0 | 4800 | 3 | 1 | 1 | yes | yes | yes | no | no | 0 |
| 10 | 88500.0 | 5500 | 3 | 2 | 4 | yes | yes | no | no | yes | 1 |

OK

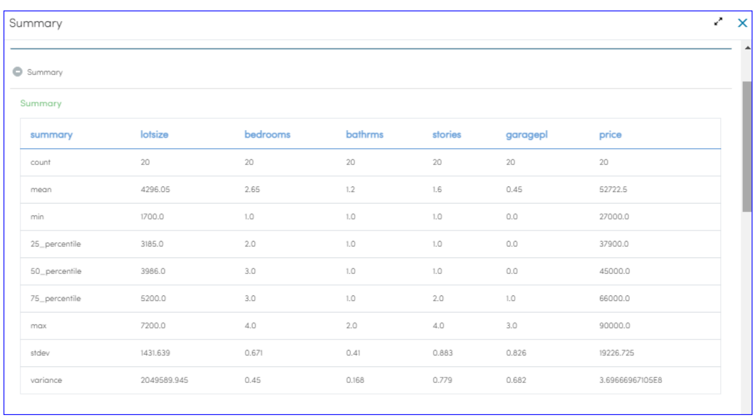
Summary statistics provides useful information about sample data. eg: measures of spread.

It provides a table with number of non-null entries (count), mean, standard deviation, and minimum and maximum value for each numerical column.

### Processor Configuration




### Processor Output: Summary Statistics



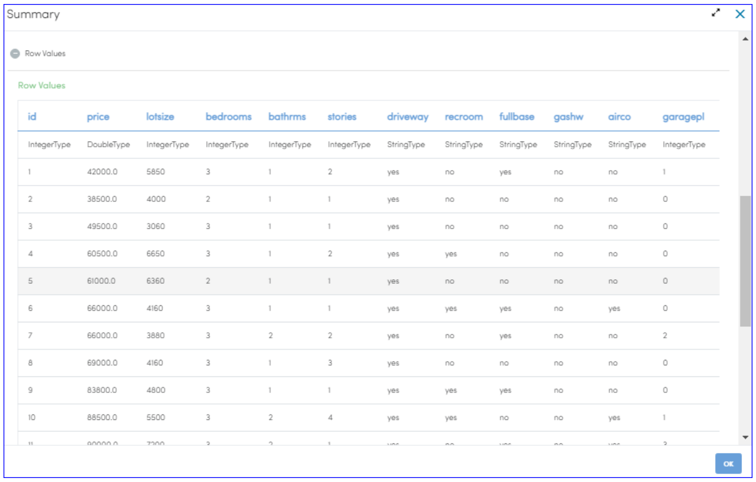

| summary | lotsize | bedrooms | bathrms | stories | garagepl | price |
|---|---|---|---|---|---|---|
| count | 20 | 20 | 20 | 20 | 20 | 20 |
| mean | 4296.05 | 2.65 | 1.2 | 1.6 | 0.45 | 52722.5 |
| min | 1700.0 | 1.0 | 1.0 | 1.0 | 0.0 | 27000.0 |
| 25_percentile | 3185.0 | 2.0 | 1.0 | 1.0 | 0.0 | 37900.0 |
| 50_percentile | 3986.0 | 3.0 | 1.0 | 1.0 | 0.0 | 45000.0 |
| 75_percentile | 5200.0 | 3.0 | 1.0 | 2.0 | 1.0 | 66000.0 |
| max | 7200.0 | 4.0 | 2.0 | 4.0 | 3.0 | 90000.0 |
| stdev | 1431.639 | 0.671 | 0.41 | 0.883 | 0.826 | 19226.725 |
| variance | 2049589.945 | 0.45 | 0.168 | 0.779 | 0.682 | 3.69666967105E8 |

### Processor Output: Sample Rows of Input Dataset

### Change Data Capture

There are many times when we need to Change Data Capture.

Below is one way to do CDC with Fire.


Summary

Row Values

Row Values

| id | price | lotsize | bedrooms | bathrms | stories | driveway | recroom | fullbase | gashw | airco | garagepl |
|---|---|---|---|---|---|---|---|---|---|---|---|
| IntegerType | DoubleType | IntegerType | IntegerType | IntegerType | IntegerType | StringType | StringType | StringType | StringType | StringType | IntegerType |
| 1 | 42000.0 | 5850 | 3 | 1 | 2 | yes | no | yes | no | no | 1 |
| 2 | 38500.0 | 4000 | 2 | 1 | 1 | yes | no | no | no | no | 0 |
| 3 | 49500.0 | 3060 | 3 | 1 | 1 | yes | no | no | no | no | 0 |
| 4 | 60500.0 | 6650 | 3 | 1 | 2 | yes | yes | no | no | no | 0 |
| 5 | 61000.0 | 6360 | 2 | 1 | 1 | yes | no | no | no | no | 0 |
| 6 | 66000.0 | 4160 | 3 | 1 | 1 | yes | yes | yes | no | yes | 0 |
| 7 | 66000.0 | 3880 | 3 | 2 | 2 | yes | no | yes | no | no | 2 |
| 8 | 69000.0 | 4160 | 3 | 1 | 3 | yes | no | no | no | no | 0 |
| 9 | 83800.0 | 4800 | 3 | 1 | 1 | yes | yes | yes | no | no | 0 |
| 10 | 88500.0 | 5500 | 3 | 2 | 4 | yes | yes | no | no | yes | 1 |

OK

### Overview

We have streaming events coming in. The events can be updates to the existing records. In the final table, we need to publish only the latest record.

### Design

We keep a staging table. This table would have all the records coming in. We do dedup at the end of the day and publish it to the final table.

Let us say that we are getting real time events of orders. As we get these events we append it to the staging table. If there are updates to an order, say an order got cancelled, we will have multiple records for that order in the staging table.

There is a final published order table where there are no duplicates. It gets updated once a day.

We join the final order table with the staging table. In doing so we get multiple order entries. We take the one with the latest timestamp and drop the others. Then for a given order we have only one record in the final table. We rewrite the final orders table with the newly calculated records.

## 12.1.6 Data Quality

### Data Quality

Data quality is an important aspect whenever we ingest data. Incomplete or wrong data can lead to more false predictions by a machine learning algorithm, we may also lose opportunities to monetize our data because of the data issues and business can lose their confidence on the data.

In sparkflows, user can create the workflow using Summary, Correlation etc nodes to get more details about the dataset.

Sample Dataset: http://eforexcel.com/wp/downloads-16-sample-csv-files-data-sets-for-testing/

Example:

#### Workflow

Below is the workflow to do Data Profile.

- Reads data from a sample dataset.
- Summary of the numeric fields.
- Correlation of the fields in dataset
- Verfiy the quality of data in sparkflows *Data Quality* tab.

#### SampleData

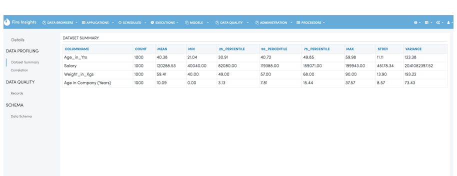
#### Summary

#### Correlation

#### Data Quality Page

#### Summary Results

**Correlation Results**




| COLUMNNAME | COUNT | MEAN | MIN | 25_PERCENTILE | 50_PERCENTILE | 75_PERCENTILE | MAX | STDEV | VARIANCE |
|---|---|---|---|---|---|---|---|---|---|
| Age_in_Yrs | 1000 | 40.38 | 21.04 | 30.91 | 40.72 | 49.85 | 59.98 | 11.11 | 123.38 |
| Salary | 1000 | 120288.53 | 40040.00 | 82080.00 | 119388.00 | 159071.00 | 199943.00 | 45178.34 | 2041082397.52 |
| Weight_in_Kgs | 1000 | 59.41 | 40.00 | 49.00 | 57.00 | 68.00 | 90.00 | 13.90 | 193.22 |
| Age in Company (Years) | 1000 | 10.09 | 0.00 | 3.13 | 7.81 | 15.44 | 37.57 | 8.57 | 73.43 |

### 12.1.7 Code

#### SQL Examples in Fire

Fire provides a SQL processer in which SQL can be written.

#### Example 1

```
select bedrooms, avg(lotsize) as avg_lotsize from fire_temp_table group by bedrooms
```

#### Example 2

```
select fire_temp_table.* , case  when fire_temp_table.DEP_DELAY_NEW > 40 then 1.0 else 0.0 END as label from fire_temp_table
```

#### Scala Examples in Fire

Fire provides a Scala processer in which Scala code can be written.

Below are a few code examples in Scala.

#### Calculate count of houses by bathrooms

```
val outDF = inDF.groupBy("bathrms").count()
outDF.registerTempTable("outDF")
```

#### For each bedroom type, find the house with the lowest price

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql._
import org.apache.spark.sql.functions._
val window = Window.partitionBy("bedrooms").orderBy("price")
val rankDF = inDF.withColumn("rank", rank() over window)
val lowestPriceDF = rankDF.filter(col("rank") === 1)
```

(continues on next page)

(continued from previous page)

```
val outDF = lowestPriceDF.
↪drop(col("rank"))
outDF.registerTempTable("outDF")
```

### Jar File Execution Example in Fire

Let's take a scenario where through CI/CD pipeline, the application jar file is built successfully and pushed into the S3 bucket.

Below are steps to execute the jar file:

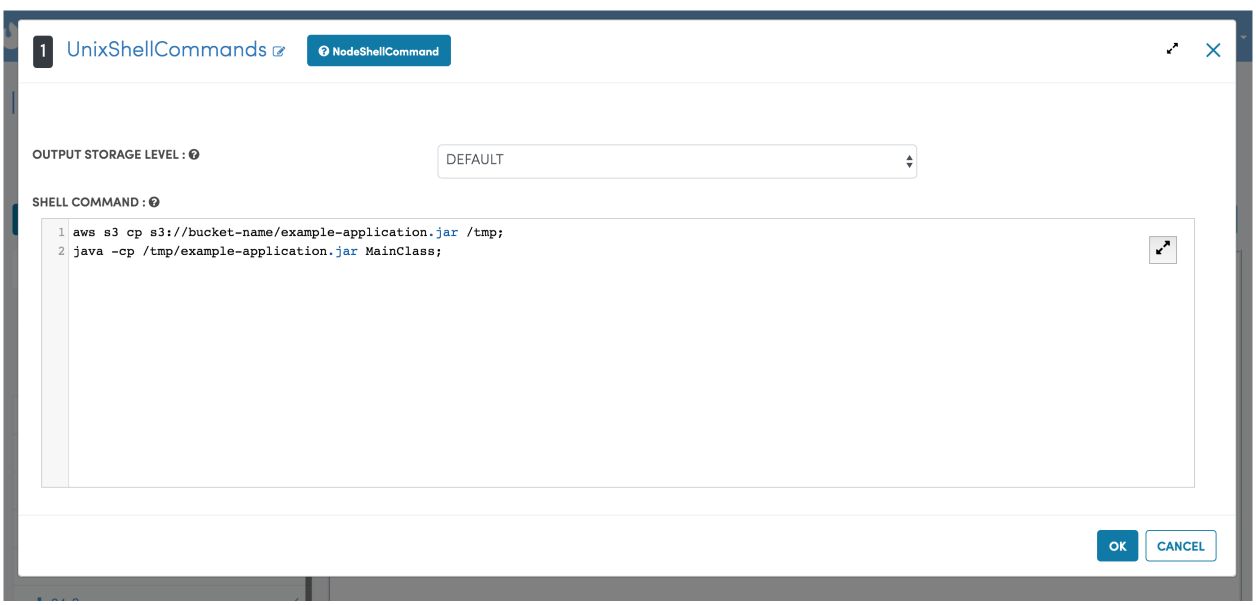
#### Step 1: Copy jar file from s3 path to /tmp directory.

```
aws s3 cp s3://bucket-name/
↪example-application.jar /tmp
```

#### Step 2: Execute jar file from /tmp directory.

```
java -cp /tmp/example-
↪application.jar MainClass
```

In the fire, both steps can be run with UnixShellCommands Node.




## 12.1.8 NLP

### Name Finder

Fire provides NameFinder Processor to easily detect named entities and numbers in text. It takes in a column name in the input DataFrame containing text. It then detects the entities and stores them into a new column.

To be able to detect entities the Name Finder needs a model. The model is dependent on the language and entity type it was trained for.

https://opennlp.apache.org/documentation/1.6.0/manual/opennlp.html#tools.namefind.recognition.cmdline

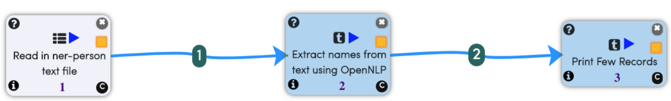
The OpenNLP project offers a number of pre-trained name finder models which are trained on various freely available corpora. They can be downloaded at the OpenNLP download page.

http://opennlp.sourceforge.net/models-1.5/

Steps for installing the OpenNLP models in Fire are covered here : http://docs.sparkflows.io/en/latest/operating/installing-opennlp.html

### Workflow

Below is a workflow which uses the NameFinder Processor.




It consists of 3 Processors:

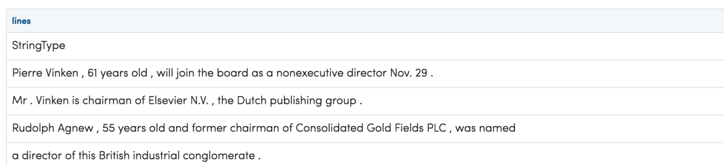
- TextFiles - It reads in the input text file and creates a row from each line of text.
- OpenNLPNameFinder - It extracts the entities from each line of text.
- PrintNRows - It prints the first 10 rows of the result.

### Textfiles

It reads in the input files from the directory data/ner-person. It places each line in the column 'line'.

### Processor Configuration

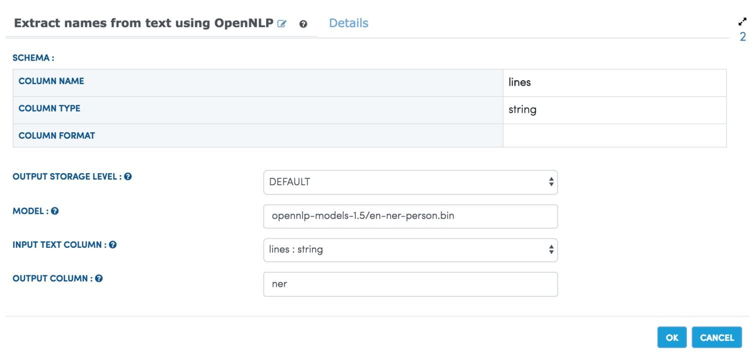

### Processor Output



| lines |
|---|
| StringType |
| Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29 . |
| Mr . Vinken is chairman of Elsevier N.V. , the Dutch publishing group . |
| Rudolph Agnew , 55 years old and former chairman of Consolidated Gold Fields PLC , was named |
| a director of this British industrial conglomerate . |

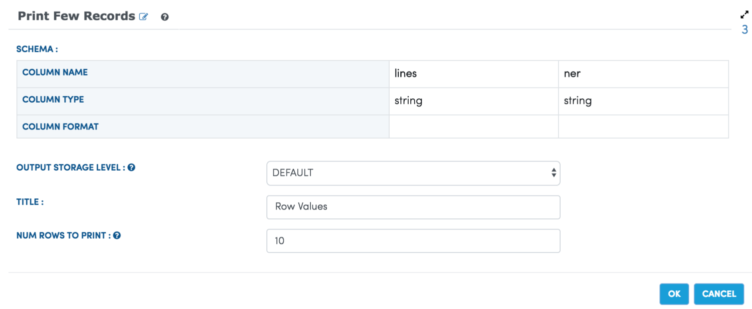
### OpenNLPNameFinder

It extracts entities from the text in the input column 'line' and stores them in the output column 'ner'. When running on the Hadoop Cluster, the model file has to be on HDFS and users have to have access to it.




### PrintNRows

It prints the first 10 rows from the result.




### 12.1.9 Streaming

#### Streaming Analytics Bike Sharing Dataset

Streaming Analytics with Apache Kafka and Apache Spark Streaming.

At Fire we are obsessed with powering our users to build amazing data analytics applications in < 30 mins.

Below we build a Streaming Analytics workflow and dashboard. It-

- Reads bike sharing data from Kafka
- Parses the incoming data
- Finds the number of rentals on an hourly basis
- Displays the results visually in a graph.

#### DataSet

The dataset contains bike rental info from 2011 and 2012 in the Capital bikeshare system, plus additional relevant information.

This dataset is from Fanaee-T and Gama (2013) and is hosted by the UCI Machine Learning Repository. It consists of 10877 rows ( can be found in /data directory of the Fire installation). Each record is count of rentals grouped by a given hour in the past and environmental factors at that time (season, holiday, temperature, wind-speed etc.)

#### Start Kafka and create Topic 'bike-sharing'

- The quick start guide of Kafka is at : https://kafka.apache.org/quickstart
- The steps for Kafka are:

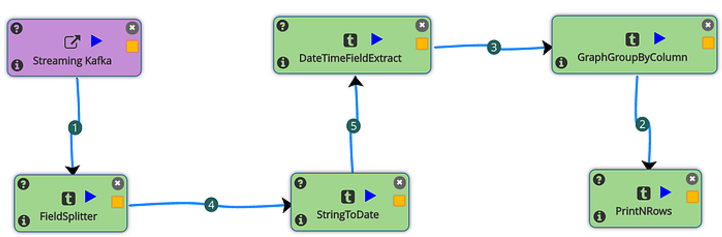
- Download Kafka
- Start zookeeper and Kafka server. You can also use an existing instance of Zookeeper/Kafka
- bin/zookeeper-server-start.sh config/zookeeper.properties
- bin/kafka-server-start.sh config/server.properties
- Create the topic 'bike-sharing'
- bin/kafka-topics.sh –create –zookeeper localhost:2181 –replication-factor 1 –partitions 1 –topic bike-sharing

### Send the data file 'bike_sharing_noheader.csv' to the Kafka Topic

- bike_sharing_noheader.csv is in the data directory of the Fire Install
- cat bike_sharing_noheader.csv | bin/kafka-console-producer.sh –broker-list localhost:9092 –topic bike-sharing

### Workflow

Below is a workflow for Streaming Analytics of the Bike Sharing dataset.



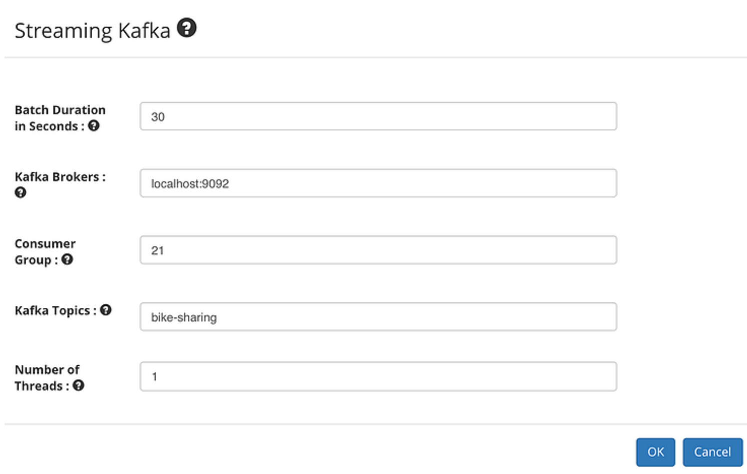

It consists of 6 Nodes:

- StreamingKafka - It reads in streaming data from the Kafka topic bike-sharing.
- FieldSplitter - It splits each line in fields.
- StringToDate - Converts the datetime column into Timestamp type.
- DateTimeFieldExtract : Extracts year, month, day, hour from the datetime column.
- GraphGroupByColumn - Groups the data on the hour column, sums it up and display it in a Graph.

- PrintNRows : Prints the first 10 records in a table.

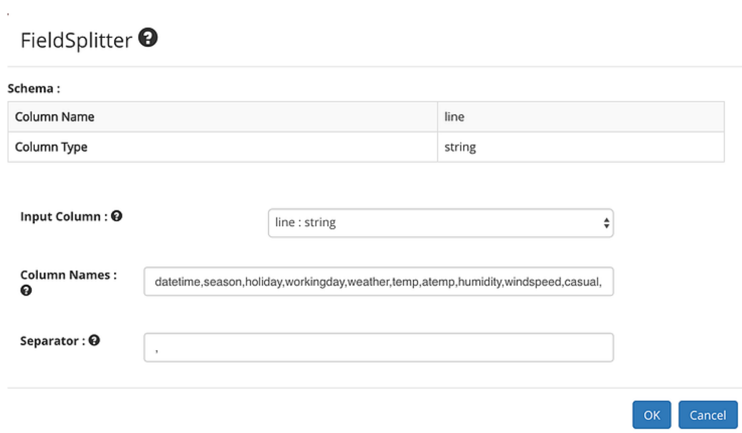
### Streaming Kafka

It reads in streaming data from Kafka and creates a dataframe with one column containing the lines.



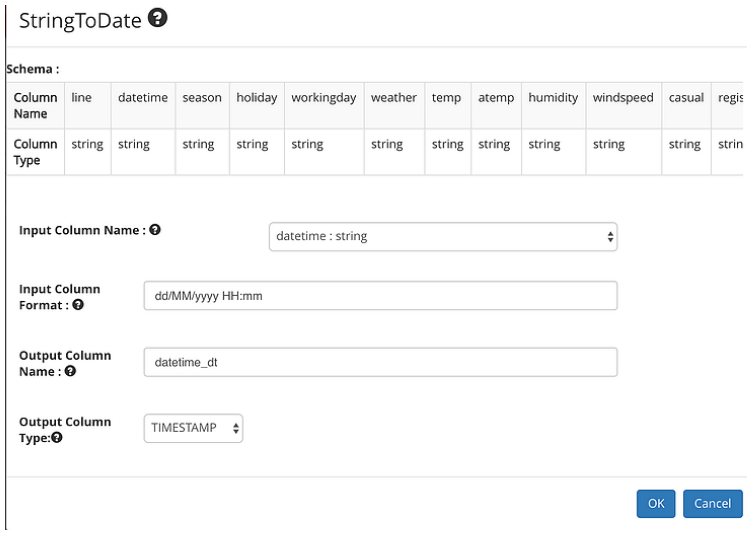

### FieldSplitter

It splits each line on the separator - comma - and outputs a new DataFrame with the columns defined.



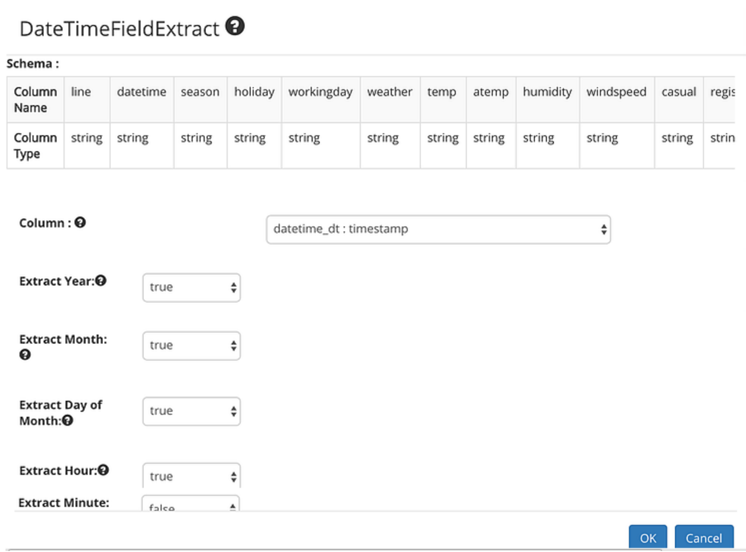

### StringToDate

It converts the datetime column into new column of type 'Timestamp'.




### DateTimeFieldExtract

It extracts the year, month, day of month and hour from the datetime_dt column.

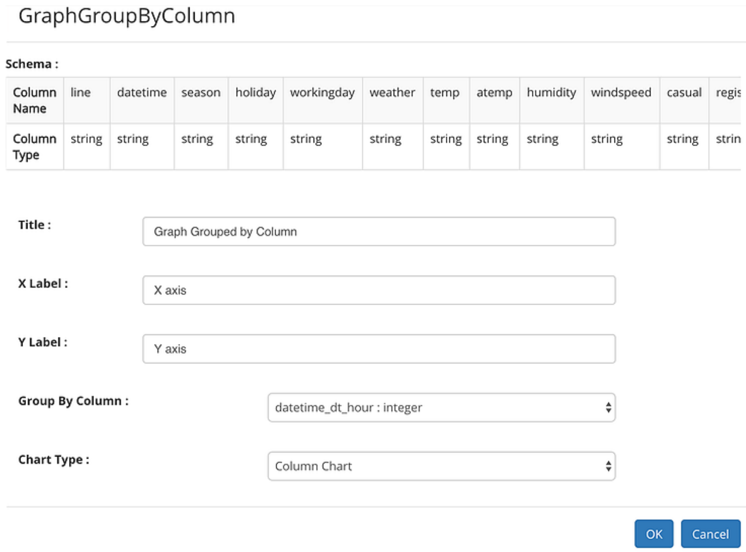



### GraphGroupByColumn

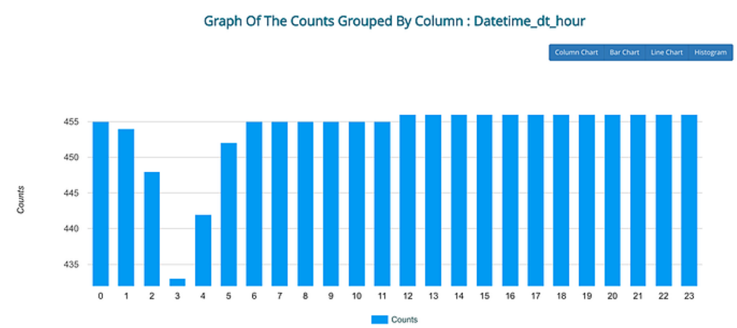
Aggregates the data on the hour column, and displays it in a Graph.

### Executing the workflow

When the workflow is executed, Fire submits a spark streaming job to the Spark cluster. The spark streaming job keeps running and




processing the incoming from Kafka. Below are some of the output produced by the job.

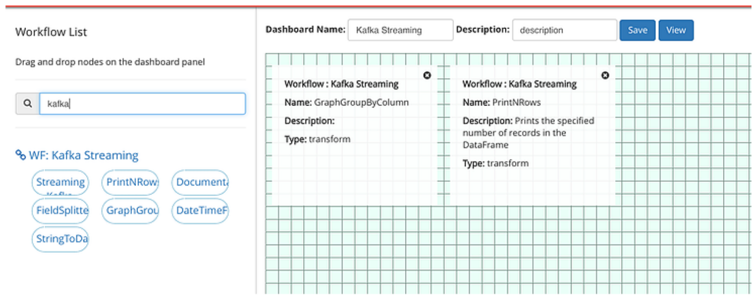


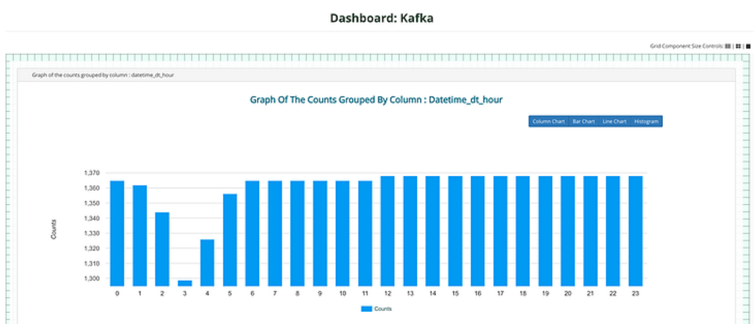

| datetime_dt_hour | count |
|---|---|
| IntegerType | LongType |
| 0 | 455 |
| 1 | 454 |
| 2 | 448 |
| 3 | 433 |
| 4 | 442 |
| 5 | 452 |
| 6 | 455 |
| 7 | 455 |
| 8 | 455 |
| 9 | 455 |

## Streaming Dashboard

Since we are still very much under 30 minutes, we also go ahead and create a Dashboard for the workflow. Since we have set the mini-batch duration to be 30 seconds, the Dashboard would update itself every 30 seconds.

Below is the Dashboard editor. Select the nodes whose output you want displayed and drag and drop them onto the canvas.




### 12.1.10 OCR

#### OCR with Tesseract

https://www.sparkflows.io/single-post/OCR-with-Tesseract-in-Sparkflows

### 12.1.11 REST API

#### Python - Infer Spark Cluster Configurations

Below is an example Python program for inferring the Apache Spark cluster configurations using the REST API.

It would infer the cluster configurations with latest changes and save the new results.

```
#!/usr/bin/python

import requests

import json


token_url = "http://localhost:8080/oauth/token"

infer_configuration_api_url = "http://localhost:8080/api/v1/configurations/infer"
```

(continues on next page)

(continued from previous page)

```
save_configuration_api_
↪url = "http://localhost:8080/
↪api/v1/configurations"

#Step A - resource owner␣
↪supplies credential #Resource␣
↪owner (enduser) credentials

#input your own username

RO_user = 'admin'

#input your own password

RO_password = 'admin'

#client␣
↪(application) credentials

client_id = 'sparkflows'
client_secret = 'secret'

#step B,␣
↪C - single call with resource␣
↪owner credentials in the␣
↪body and client credentials␣
↪as the basic auth header␣
↪will return#access_token

data = {'grant_type
↪': 'password','username': RO_
↪user, 'password': RO_password}

access_
↪token_response = requests.
↪post(token_url, data=data,
↪ verify=False, allow_
↪redirects=False, auth=(client_
↪id, client_secret))

print(access_
↪token_response.headers)
print(access_
↪token_response.text)

tokens = json.loads(access_
↪token_response.text)
print( "access token:␣
↪" + tokens['access_token'])

# Step-
↪ now use the access_token␣
↪to call infer configuration␣
↪api and its save api.

api_call_headers␣
↪= {'Authorization': 'Bearer␣
↪' + tokens['access_token']}
```

(continues on next page)

(continued from previous page)

```
print( api_call_headers)

#infer the hadoop configuration

infer_configuration_
↪api_response = requests.
↪get(infer_configuration_
↪api_url, headers=api_
↪call_headers, verify=False)
print("␣
↪infer configuration response␣
↪: "+ infer_configuration_
↪api_response.text)

#save the hadoop configuration

save_configuration_
↪api_response = requests.
↪post(save_configuration_
↪api_url,infer_configuration_
↪api_response, headers=api_
↪call_headers,   verify=False)

print(" configuration after␣
↪save : "+save_configuration_
↪api_response.text)
```

### 12.1.12 Time Series

#### Stock Forecasting

#### Objective

Stock forecasting helps production units to get an idea about raw material, pricing of goods, improvement in supply, chain management and proper control of sales.

#### Dataset

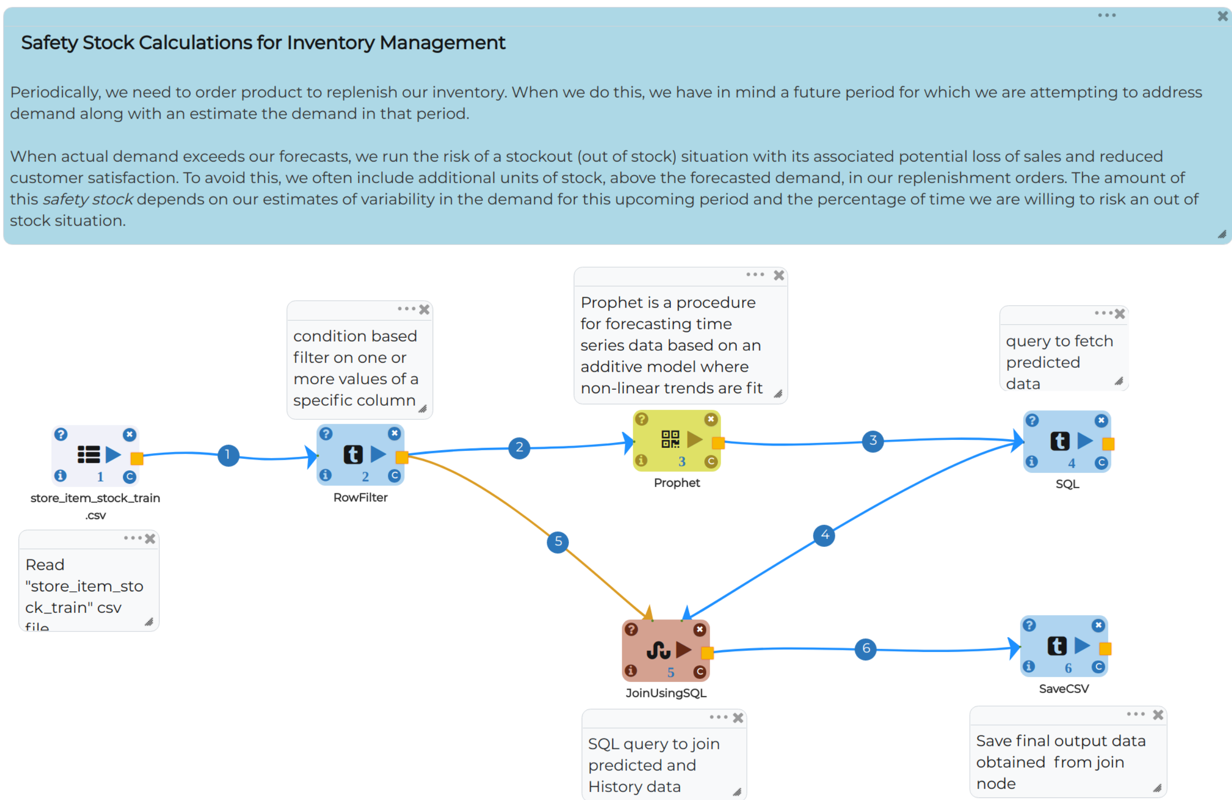
Dataset contains 4 columns as follows:-

- Date - Date when product was sold
- Store - Store id from where product got sold
- Item - Item id
- Sales - Quantity of product sold

Predict future sales of items at particular store

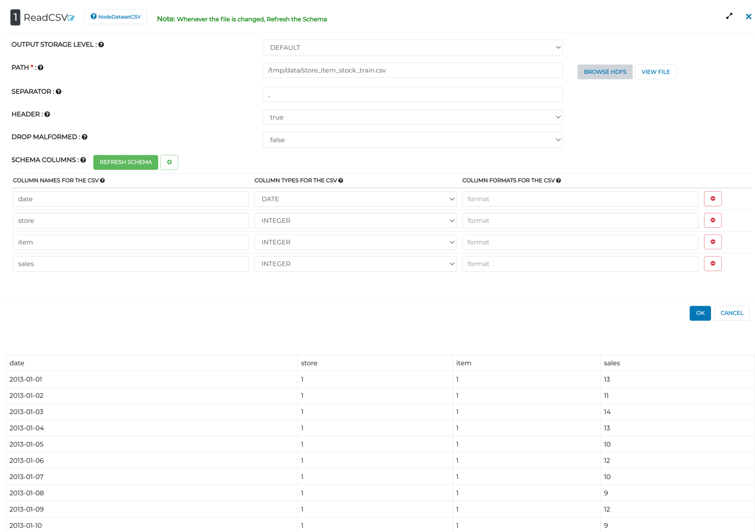
### Prophet Time Series Modelling Workflow on Multivariate Data

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends fit with yearly, weekly, daily, seasonality and holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.



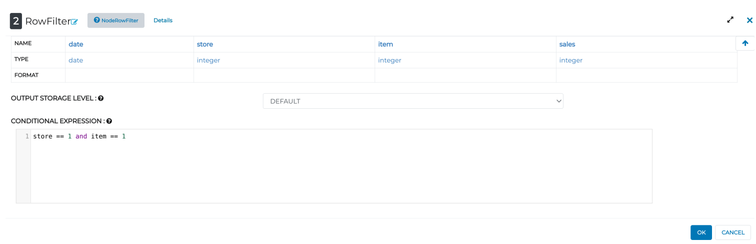

### Node 1 - ReadCSV

- Reads the given CSV file : store_item_stock_train.csv
- Below are the first 10 rows of data
- Columns contain data as datetype, store and item which are categorical variables and sales which is a continuous variable.



### Node 2 - RowFilter

- Filters data by row with respect to store and item



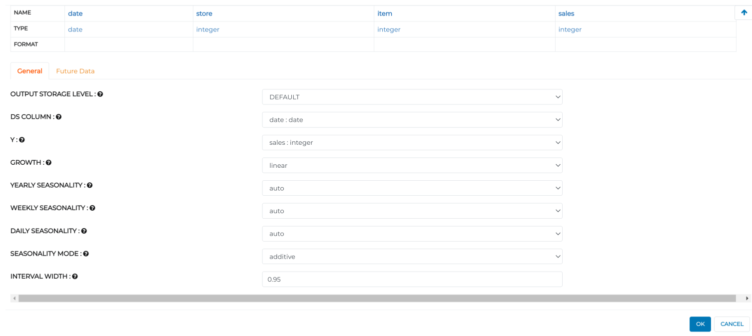
### Node 3 - Prophet

Used Facebook Prophet to create the ML model.

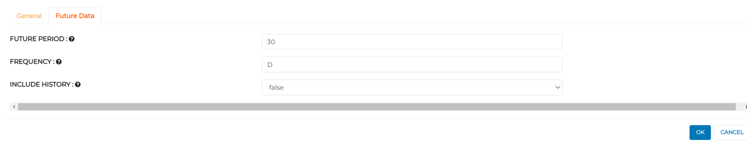
**General Section of Prophet Model**

- Set Date column in DS column field
- Y is the target variable. Set it to the Sales column
- Set Growth as linear or logistic
- We are using prophet model so it is sufficient to select seasonality in auto mode
- Set mode of seasonality as additive or multiplicative
- Set confidence Interval (0 to 1) which gives a range of plausible values for the parameter of interest.

**Future Data section of Prophet model**



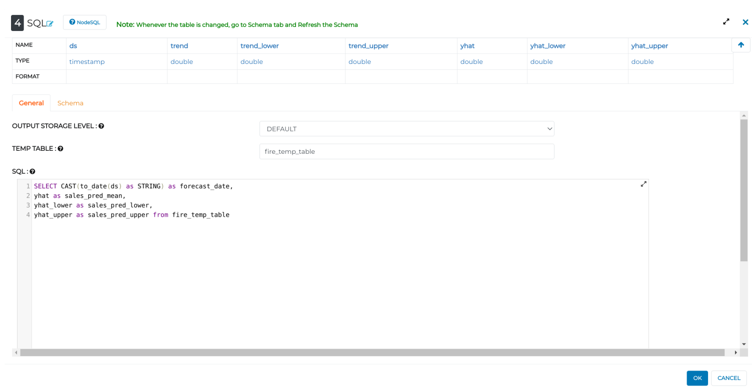

- FUTURE PERIOD block gives the number of steps we want to predict
- FREQUENCY can be Monthly or Daily
- Set INCLUDE HISTORY to true for testing the model and False for production




## Node 4 - SQL

**General Section of SQL node**

- Renames columns forecasted by Prophet

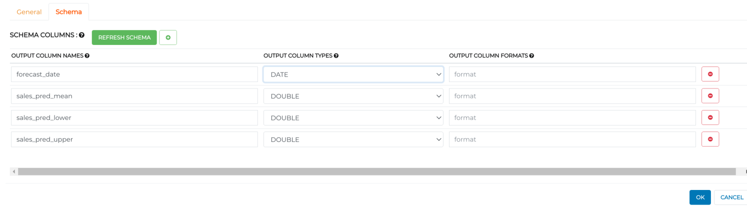



**Schema Section of SQL node**

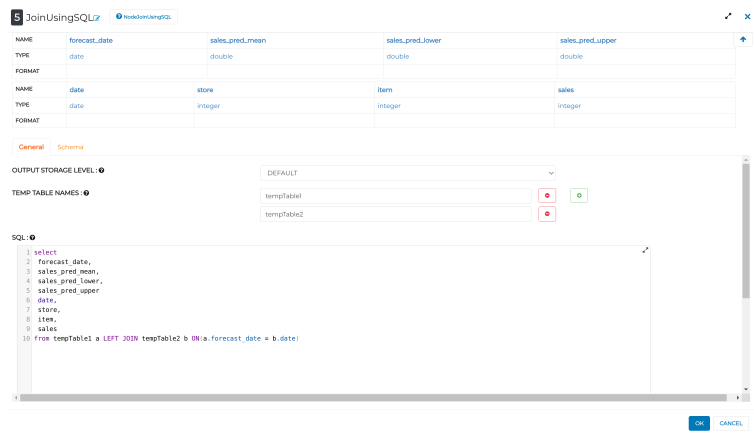
- Refreshes Schema and sets data type with respect to columns

## Node 5 - JoinUsingSQL

**General Section of JoinUsingSQL node**

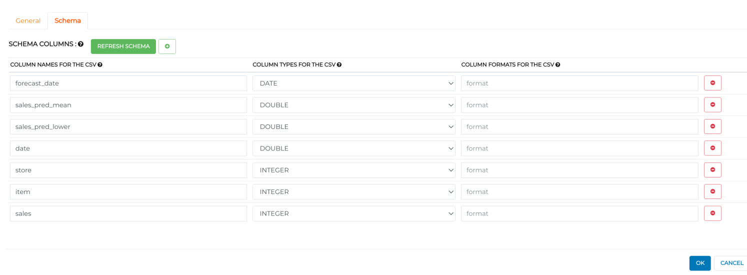



- Joins Prediction (from SQL node) and Historical Data(from RowFilter node)




**Schema Section of JoinUsingSQL node**

- Follow the same steps as in Schema Section of SQL node

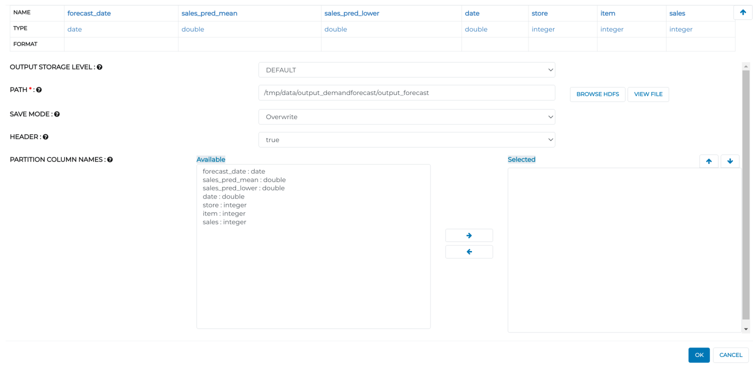



## Node 6 - SaveCSV

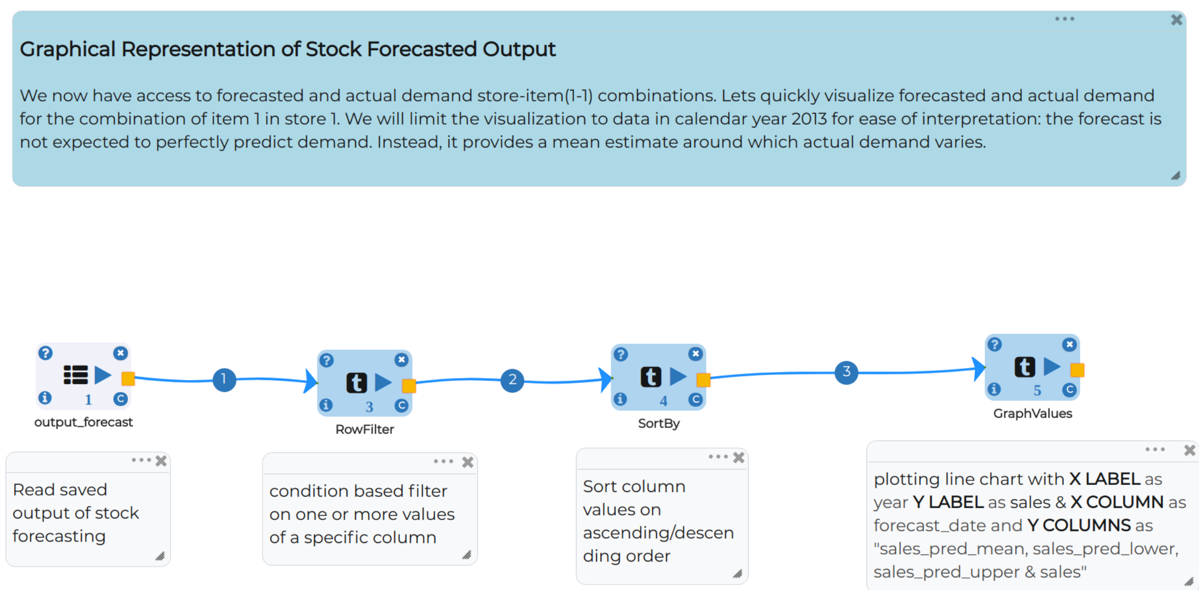
- Sets path where you want to save the final output

## Output Visualization

Graphical representation is the best way to understand insights from data. It refers to the use of charts and graphs to visually display, analyze, clarify, and interpret numerical data, functions and other qualitative structures.




Below is the workflow for Visualizing it:




## Node 1 - ReadCSV

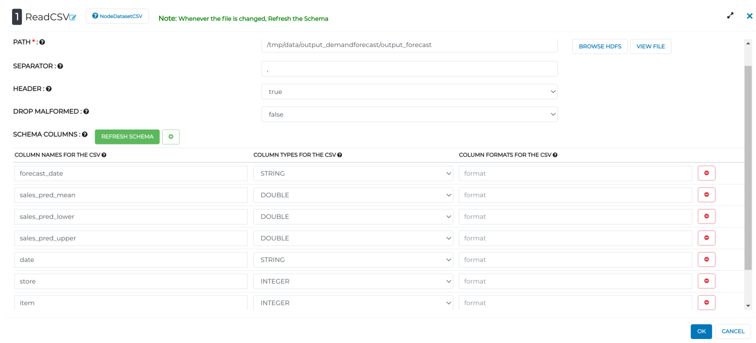
- Reads output CSV which we have saved from Stock Forecasting.

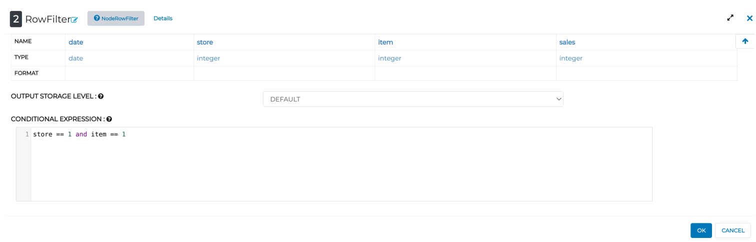
## Node 2 - RowFilter

- Filters dataframe with categorical variables like store and item

## Node 3 - SortBy

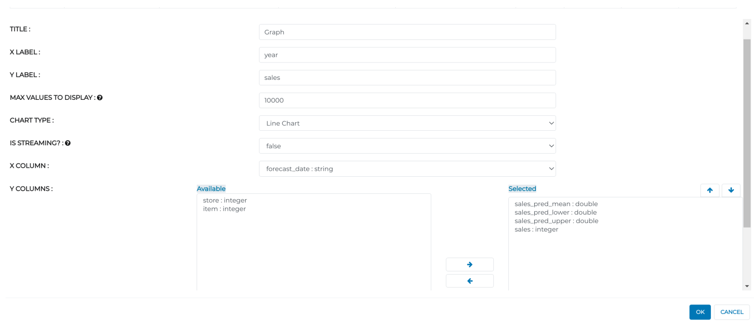
- Gives options to sort our Dataset based on columns in ascending and descending order




## Node 4 - GraphValue

- Defines labels for X-axis and Y-axis
- Sets columns for X-axis and Y-axis



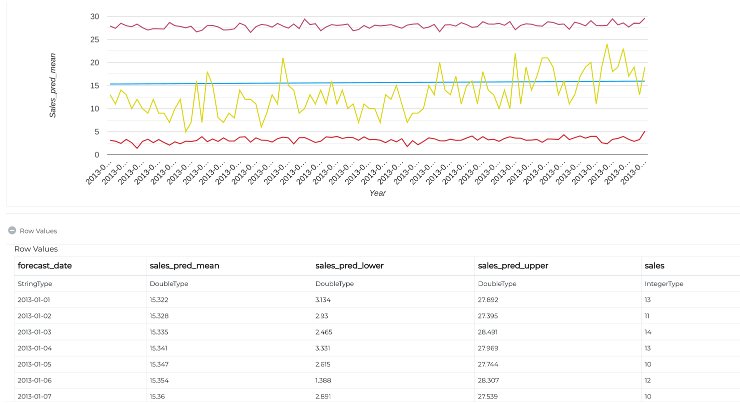

**Graph obtained**

- Sales_pred_mean - Blue line
- Sales_pred_lower - Red line
- Sales_pred_upper - Magenta line
- Sales - Yellow line
- Now have a look into graph

## Air Passengers Forecasting

### Objective

The objective is to develop a time series model to predict future demand of air pas-




Row Values

| forecast_date | sales_pred_mean | sales_pred_lower | sales_pred_upper | sales |
|---|---|---|---|---|
| StringType | DoubleType | DoubleType | DoubleType | IntegerType |
| 2013-01-01 | 15.322 | 3.134 | 27.892 | 13 |
| 2013-01-02 | 15.328 | 2.93 | 27.395 | 11 |
| 2013-01-03 | 15.335 | 2.465 | 28.491 | 14 |
| 2013-01-04 | 15.341 | 3.331 | 27.969 | 13 |
| 2013-01-05 | 15.347 | 2.615 | 27.744 | 10 |
| 2013-01-06 | 15.354 | 1.388 | 28.307 | 12 |
| 2013-01-07 | 15.36 | 2.891 | 27.539 | 10 |

sengers which helps Airline company to take decision on aircraft fleet management.

#### Dataset

Dataset contains 2 columns as follows:-

Month - Month of the year

Passengers - Total number of passengers travelled in that particular month

Air Passengers Occupancy Prediction

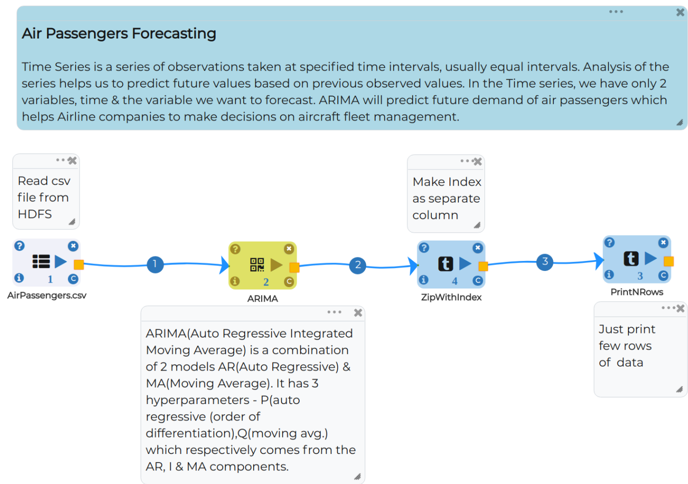
#### Time Series Modelling Workflow on Univariate Data

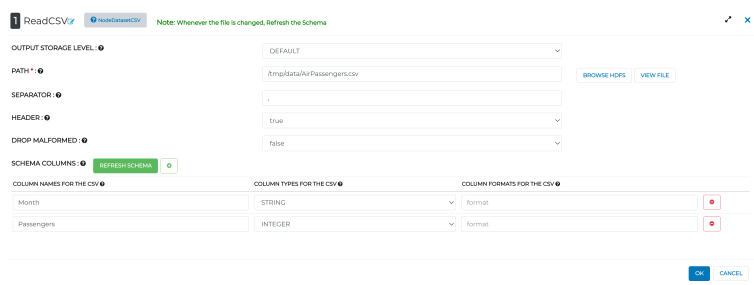
The auto_arima work to fit the best ARIMA(Autoregressive Integrated Moving Average) model to a univariate time arrangement is indicated by either AIC, AICc, BIC or HQIC. The capacity plays out an inquiry (either stepwise or parallelized) over conceivable model requests inside the requirements given.

The auto_arima capacity can be overwhelming. There are a ton of boundaries to tune, and the result is vigorously subject to various themes. In this segment, we spread out a few contemplations you'll need to make when you fit your ARIMA models.

#### Node 1 - ReadCSV

- Reads the given CSV file : AirPassengers.csv

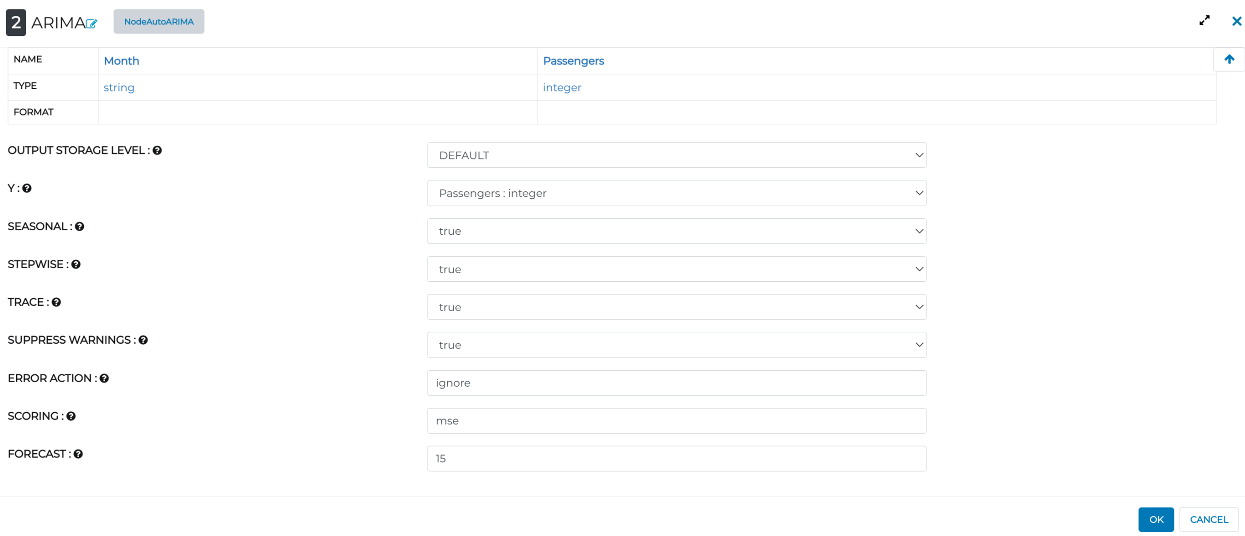



## Node 2 - ARIMA

- p - The number of lag observations included in the model, also called the lag order.
- d - The number of times that the raw observations are different, also called the degree of differencing.
- q - The size of the moving average window, also called the order of moving average.

  Not to worry about p,d,q in this case because we have an interesting model called - AUTO-ARIMA (Able to select automatically optimal value)
- Y - Target Variable (Passengers Per Month)
- SEASONAL - Automatically True but you can change as false if you want as non-seasonal
- SCORING - How do you want to evaluate your model performance like - MSE, MAE
- FORECAST - Number of steps you want to forecast




## Summary

- The model summary reveals a lot of information

## Node 3 - ZipWithIndex

- Creates new column from index of Dataset

## Node 4 - PrintNRows

- Number of rows you want to print to see the final result

## Final Result

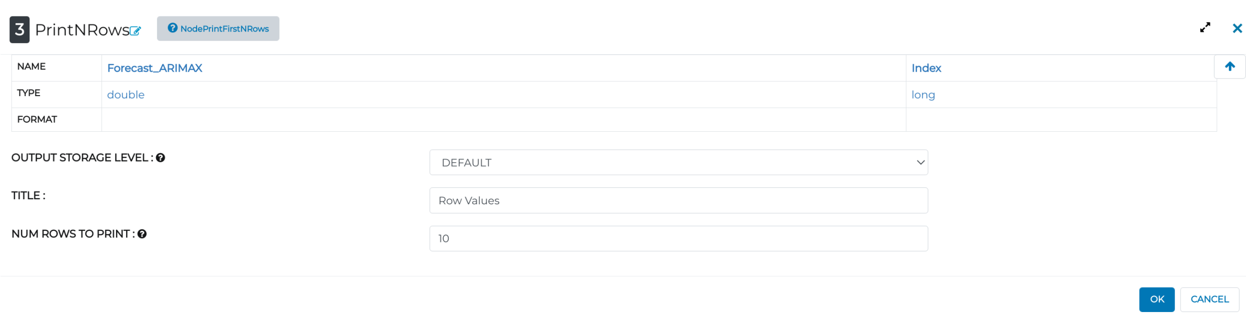
Lets check a few rows of forecasted data by ARIMA Model

## Time Series Feature Engineering

## Objective

It is a process of extracting new features from raw data via data mining techniques. These features can be used to improve the performance of models.

```
Summary:
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  144
Model:               SARIMAX(4, 1, 3)   Log Likelihood                -674.913
Date:                Fri, 30 Oct 2020   AIC                           1365.825
Time:                        12:33:24   BIC                           1389.528
Sample:                             0   HQIC                          1375.457
                                - 144
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.5582      0.117     -4.782      0.000      -0.787      -0.329
ar.L2          0.4935      0.113      4.375      0.000       0.272       0.715
ar.L3          0.1238      0.128      0.970      0.332      -0.126       0.374
ar.L4         -0.5213      0.085     -6.136      0.000      -0.688      -0.355
ma.L1          0.9069      0.094      9.657      0.000       0.723       1.091
ma.L2         -0.5590      0.145     -3.866      0.000      -0.842      -0.276
ma.L3         -0.7385      0.109     -6.778      0.000      -0.952      -0.525
sigma2       724.1724     85.616      8.458      0.000     556.369     891.976
===================================================================================
Ljung-Box (Q):                      256.02   Jarque-Bera (JB):                14.59
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               5.66   Skew:                             0.74
Prob(H) (two-sided):                  0.00   Kurtosis:                         3.52
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```




| Forecast_ARIMAX | Index |
|---|---|
| DoubleType | LongType |
| 467.573805910366 | 0 |
| 490.49456508436225 | 1 |
| 509.1369442565889 | 2 |
| 492.55476914877306 | 3 |
| 495.3059691093772 | 4 |
| 475.94780369392686 | 5 |
| 476.3398372576565 | 6 |
| 475.5521380450703 | 7 |
| 472.35382516685223 | 8 |
| 483.8896762403088 | 9 |

### Dataset

Dataset contains 4 columns as below:

- Date - Date when product was sold
- Store - Store id from where product got sold
- Item - Item id
- Sales - Quantity of product sold

Create new feature from existing table to improve performance of models

### Feature Engineering Workflow

Each column is a feature. But all features may not produce the best results from models, so feature engineering plays an important role in choosing the right features. A model will not entirely improve its prescient force, yet will offer the adaptability to utilize less unpredictable models that are quicker to run and more handily.

### Moving average

**One step moving average**

- Moving average is commonly used to streamline short-period fluctuations in time series data and feature long-term patterns.
- For one step, window size will be from -1 to 1 for sales data

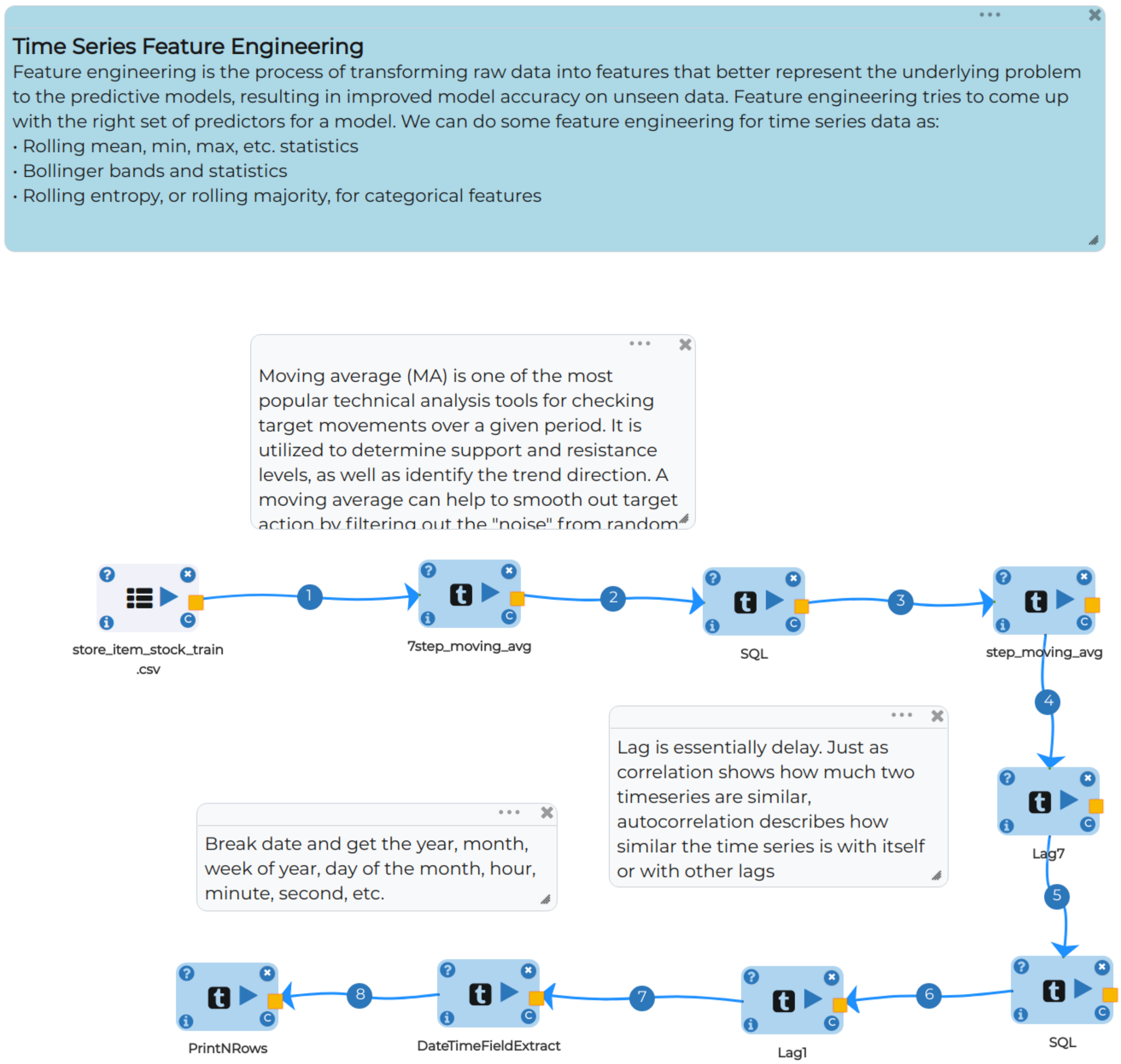
**Seven step moving average**

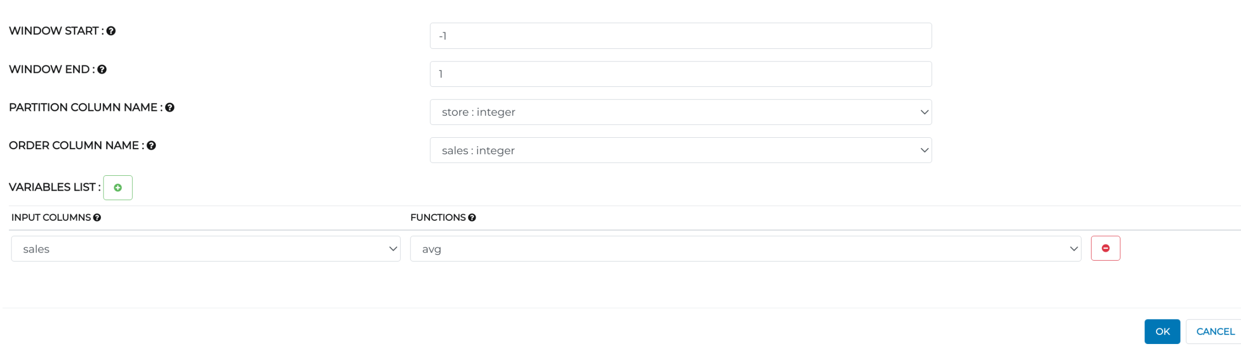
- For seven step, window size will be from -7 to 7 for sales data
- Moving average output

### Extract Date Time Features

- Break date and get the year, month, week of year, day of the month, hour, minute, second, etc.

- Output of Date Time Features


Time Series Feature Engineering
Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data. Feature engineering tries to come up with the right set of predictors for a model. We can do some feature engineering for time series data as:
• Rolling mean, min, max, etc. statistics
• Bollinger bands and statistics
• Rolling entropy, or rolling majority, for categorical features
Moving average (MA) is one of the most popular technical analysis tools for checking target movements over a given period. It is utilized to determine support and resistance levels, as well as identify the trend direction. A moving average can help to smooth out target
store_item_stock_train .csv
7step_moving_avg
SQL
step_moving_avg
Lag is essentially delay. Just as correlation shows how much two timeseries are similar, autocorrelation describes how similar the time series is with itself or with other lags
Lag7
Break date and get the year, month, week of year, day of the month, hour, minute, second, etc.
PrintNRows
DateTimeFieldExtract
Lag1
SQL


WINDOW START :
-1
WINDOW END :
1
PARTITION COLUMN NAME :
store : integer
ORDER COLUMN NAME :
sales : integer
VARIABLES LIST :
INPUT COLUMNS
FUNCTIONS
sales
avg
OK
CANCEL

| date | store | item | sales | mean_sales | 7_mean_sales |
|---|---|---|---|---|---|
| DateType | IntegerType | IntegerType | IntegerType | DoubleType | DoubleType |
| 2013-01-08 | 1 | 1 | 9 | 9.0 | 10.75 |
| 2013-01-10 | 1 | 1 | 9 | 9.333333333333334 | 11.0 |
| 2013-01-05 | 1 | 1 | 10 | 9.666666666666666 | 11.3 |
| 2013-01-07 | 1 | 1 | 10 | 10.333333333333334 | 11.3 |
| 2013-01-02 | 1 | 1 | 11 | 11.0 | 11.3 |
| 2013-01-06 | 1 | 1 | 12 | 11.666666666666666 | 11.3 |
| 2013-01-09 | 1 | 1 | 12 | 12.333333333333334 | 11.3 |
| 2013-01-01 | 1 | 1 | 13 | 12.666666666666666 | 11.3 |
| 2013-01-04 | 1 | 1 | 13 | 13.333333333333334 | 11.555555555555555 |
| 2013-01-03 | 1 | 1 | 14 | 13.5 | 11.875 |

| | |
|---|---|
| COLUMN : | date : date |
| EXTRACT YEAR : | true |
| EXTRACT MONTH : | true |
| EXTRACT DAY OF MONTH : | true |
| EXTRACT HOUR : | true |
| EXTRACT MINUTE : | true |
| EXTRACT SECOND : | true |
| EXTRACT WEEKOFYEAR : | true |

| date | store | item | sales | date_year | date_month | date_dayofmonth | date_hour | date_minute | date_second | date_weekofyear |
|---|---|---|---|---|---|---|---|---|---|---|
| DateType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType |
| 2013-01-01 | 1 | 1 | 13 | 2013 | 1 | 1 | 0 | 0 | 0 | 1 |
| 2013-01-02 | 1 | 1 | 11 | 2013 | 1 | 2 | 0 | 0 | 0 | 1 |
| 2013-01-03 | 1 | 1 | 14 | 2013 | 1 | 3 | 0 | 0 | 0 | 1 |
| 2013-01-04 | 1 | 1 | 13 | 2013 | 1 | 4 | 0 | 0 | 0 | 1 |
| 2013-01-05 | 1 | 1 | 10 | 2013 | 1 | 5 | 0 | 0 | 0 | 1 |
| 2013-01-06 | 1 | 1 | 12 | 2013 | 1 | 6 | 0 | 0 | 0 | 1 |
| 2013-01-07 | 1 | 1 | 10 | 2013 | 1 | 7 | 0 | 0 | 0 | 2 |
| 2013-01-08 | 1 | 1 | 9 | 2013 | 1 | 8 | 0 | 0 | 0 | 2 |
| 2013-01-09 | 1 | 1 | 12 | 2013 | 1 | 9 | 0 | 0 | 0 | 2 |
| 2013-01-10 | 1 | 1 | 9 | 2013 | 1 | 10 | 0 | 0 | 0 | 2 |

### Lags Feature

- Lag is used to make non-stationary data into stationary data
- Outliers are easily discernible on a lag plot
- acf and pacf plot is used to calclulate best lags

**Lag one**

- The most commonly used lag is 1, called a first-order lag
- Window shift is one

| | |
|---|---|
| PARTITIONBY : | store |
| ORDERBY : | date |
| WINDOW FUNCTION : | lag |
| ANALYTICS COLUMN : | sales : integer |
| WINDOW OFFSET : | 1 |

**Lag seven**

- Window shift is seven

| date | store | item | sales | lag | 7_lag |
|---|---|---|---|---|---|
| DateType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType |
| 2013-01-01 | 1 | 1 | 13 | None | None |
| 2013-01-02 | 1 | 1 | 11 | 13 | None |
| 2013-01-03 | 1 | 1 | 14 | 11 | None |
| 2013-01-04 | 1 | 1 | 13 | 14 | None |
| 2013-01-05 | 1 | 1 | 10 | 13 | None |
| 2013-01-06 | 1 | 1 | 12 | 10 | None |
| 2013-01-07 | 1 | 1 | 10 | 12 | None |
| 2013-01-08 | 1 | 1 | 9 | 10 | 13 |
| 2013-01-09 | 1 | 1 | 12 | 9 | 11 |
| 2013-01-10 | 1 | 1 | 9 | 12 | 14 |

### New feature data

## Anamoly Detection for IOT Devices

### Objective

Anomaly detection issue for time arrangement can be planned as discovering exception information guides relative toward some

| date | store | item | sales | date_year | date_month | date_dayofmonth | date_hour | date_minute | date_second | date_weekofyear | lag | 7_lag | mean_sales | 7_mean_sales |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DateType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | IntegerType | DoubleType | DoubleType |
| 2013-01-01 | 1 | 1 | 13 | 2013 | 1 | 1 | 0 | 0 | 0 | 1 | None | None | 12.666666666666666 | 11.3 |
| 2013-01-02 | 1 | 1 | 11 | 2013 | 1 | 2 | 0 | 0 | 0 | 1 | 13 | None | 11.0 | 11.3 |
| 2013-01-03 | 1 | 1 | 14 | 2013 | 1 | 3 | 0 | 0 | 0 | 1 | 11 | None | 13.5 | 11.875 |
| 2013-01-04 | 1 | 1 | 13 | 2013 | 1 | 4 | 0 | 0 | 0 | 1 | 14 | None | 13.333333333333334 | 11.555555555555555 |
| 2013-01-05 | 1 | 1 | 10 | 2013 | 1 | 5 | 0 | 0 | 0 | 1 | 13 | None | 9.666666666666666 | 11.3 |
| 2013-01-06 | 1 | 1 | 12 | 2013 | 1 | 6 | 0 | 0 | 0 | 1 | 10 | None | 11.666666666666666 | 11.3 |
| 2013-01-07 | 1 | 1 | 10 | 2013 | 1 | 7 | 0 | 0 | 0 | 2 | 12 | None | 10.333333333333334 | 11.3 |
| 2013-01-08 | 1 | 1 | 9 | 2013 | 1 | 8 | 0 | 0 | 0 | 2 | 10 | 13 | 9.0 | 10.75 |
| 2013-01-09 | 1 | 1 | 12 | 2013 | 1 | 9 | 0 | 0 | 0 | 2 | 9 | 11 | 12.333333333333334 | 11.3 |
| 2013-01-10 | 1 | 1 | 9 | 2013 | 1 | 10 | 0 | 0 | 0 | 2 | 12 | 14 | 9.333333333333334 | 11.0 |

norm or common sign. Our center will be from a machine persopective, for example, surprising spikes, level move highlighting disintegrating soundness of a machine.

### Dataset

Dataset contains 4 columns as follows:-

- Datetime - 10 mins time interval of accelerometer data
- 4-Bearings - Contains reading of devices

### Anamoly Detection using Prophet Time Series Model Workflow

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends fit with yearly, weekly, daily, seasonality and holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Time-series anomaly detection is a feature used to identify unusual patterns that do not conform to expected behavior, called outliers.

### Data Preprocessing

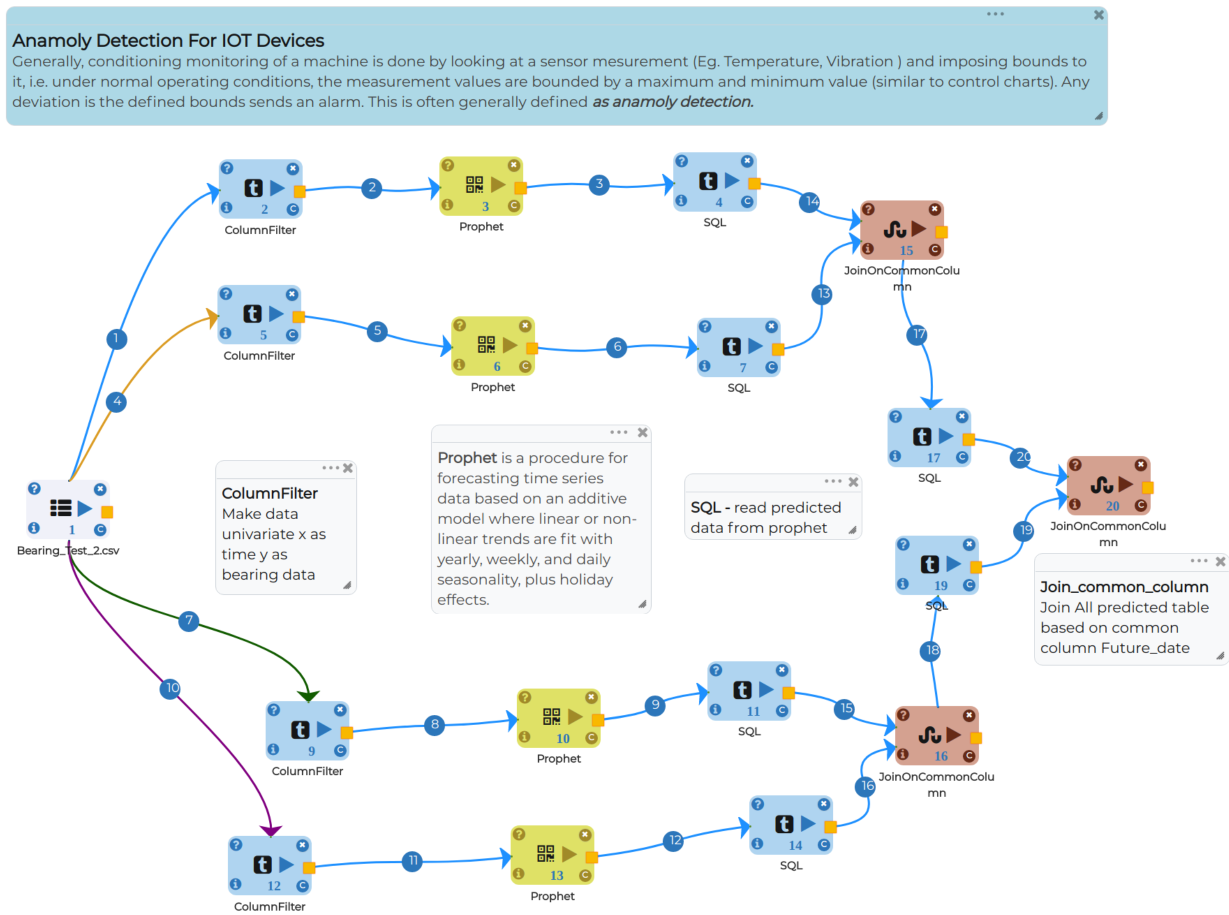
- **Column Filter** convert multivariate data into univariate for prophet model
- **Output** Univariate data

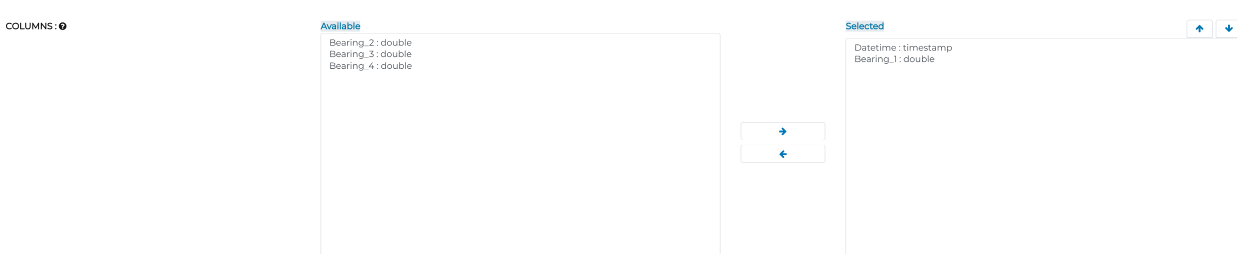
### Data Modeling

- **Prophet** Model for anomaly detication using mean as threshold value

**General Section of Prophet Model**

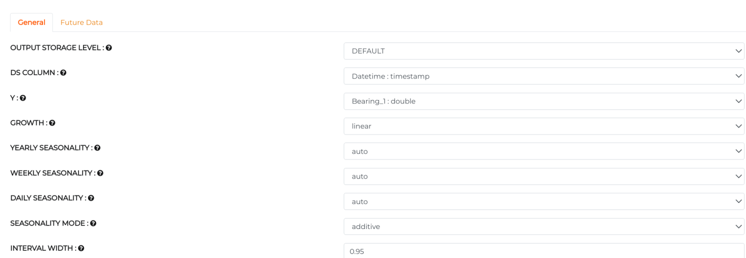
- Set Datetime column in DS column field


Anamoly Detection For IOT Devices
Generally, conditioning monitoring of a machine is done by looking at a sensor mesurement (Eg. Temperature, Vibration ) and imposing bounds to it, i.e. under normal operating conditions, the measurement values are bounded by a maximum and minimum value (similar to control charts). Any deviation is the defined bounds sends an alarm. This is often generally defined *as anamoly detection.*
ColumnFilter
Prophet
SQL
JoinOnCommonColumn
Bearing_Test_2.csv
ColumnFilter
Make data univariate x as time y as bearing data
Prophet is a procedure for forecasting time series data based on an additive model where linear or non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.
SQL - read predicted data from prophet
Join_common_column
Join All predicted table based on common column Future_date


COLUMNS :
Available
Bearing_2 : double
Bearing_3 : double
Bearing_4 : double
Selected
Datetime : timestamp
Bearing_1 : double

| Datetime | Bearing_1 |
|---|---|
| TimestampType | DoubleType |
| 2004-02-12 10:32:39 | 0.058332877581913 |
| 2004-02-12 10:42:39 | 0.058995214610088 |
| 2004-02-12 10:52:39 | 0.060236437326041 |
| 2004-02-12 11:02:39 | 0.061455442160261 |
| 2004-02-12 11:12:39 | 0.061360759802725 |
| 2004-02-12 11:22:39 | 0.061664827384149 |
| 2004-02-12 11:32:39 | 0.061943893744811 |
| 2004-02-12 11:42:39 | 0.061230528834415 |
| 2004-02-12 11:52:39 | 0.062279749987792 |
| 2004-02-12 12:02:39 | 0.059890277845597 |

- Y is the target variable. Set it to the reading of bearings
- Set Growth as linear or logistic
- We are using prophet model so that it is self-sufficient to select seasonality in auto mode
- Set mode of seasonality as additive or multiplicative
- Set confidence Interval (0 to 1) which gives a range of plausible values for the parameter of interest.




**Future Data section of Prophet model**

- FUTURE PERIOD block gives the number of steps we want to predict


- **SQL** set mean column to set threshold

## Model prediction

- **Threshold** to compare anomaly

| ds | trend | trend_lower | trend_upper | yhat | yhat_lower | yhat_upper |
|---|---|---|---|---|---|---|
| TimestampType | DoubleType | DoubleType | DoubleType | DoubleType | DoubleType | DoubleType |
| 2004-02-13 12:02:39 | 0.006557040306542335 | -0.4314989436380987 | 0.443870248355367 | 0.006557040306542335 | -0.4314334302089222 | 0.44348778316922943 |
| 2004-02-14 12:02:39 | -0.04765893255461476 | -1.2551897482733552 | 1.235643745446955 | -0.04765893255461476 | -1.2552038753818562 | 1.2358456648999048 |
| 2004-02-15 12:02:39 | -0.10187490541578374 | -2.4618087335060457 | 2.188138763682763 | -0.10187490541578374 | -2.4616337516841176 | 2.187445509713839 |
| 2004-02-16 12:02:39 | -0.15609087827695273 | -3.7689904345524265 | 3.35756280566189 | -0.15609087827695273 | -3.7689098347803065 | 3.357621979300829 |
| 2004-02-17 12:02:39 | -0.2103068511381217 | -5.3647822962412555 | 4.605712801394639 | -0.2103068511381217 | -5.3638858389282635 | 4.605428933121821 |
| 2004-02-18 12:02:39 | -0.2645228239992907 | -6.981627822573064 | 6.1714043515849175 | -0.2645228239992907 | -6.98272353061758 | 6.171796066104398 |
| 2004-02-19 12:02:39 | -0.3187387968604597 | -8.94688944368949 | 7.832083459560579 | -0.3187387968604597 | -8.947444893896066 | 7.832509699896468 |
| 2004-02-20 12:02:39 | -0.37295476972162867 | -10.606858671880655 | 10.032112103876145 | -0.37295476972162867 | -10.606690639862599 | 10.032533196920488 |
| 2004-02-21 12:02:39 | -0.42717074258279764 | -12.637549731463602 | 11.838283845322355 | -0.42717074258279764 | -12.636874299933673 | 11.838282437376805 |
| 2004-02-22 12:02:39 | -0.4813867154439667 | -14.90960174003428 | 14.331184407396112 | -0.4813867154439667 | -14.909209104192312 | 14.331133324256857 |

| Future_time | Bearing_3_pred | Bearing_4_pred | Bearing_2_pred | Bearing_1_pred |
|---|---|---|---|---|
| TimestampType | DoubleType | DoubleType | DoubleType | DoubleType |
| 2004-02-13 12:02:39 | 0.12892150390232607 | 0.018985275825913313 | 0.10739056982158202 | 0.006557040306542335 |
| 2004-02-14 12:02:39 | 0.17377822690500572 | -0.00595058722291632 | 0.1394695544580692 | -0.04765893255461476 |
| 2004-02-15 12:02:39 | 0.21863494990768537 | -0.030886450271745948 | 0.17154853909455636 | -0.10187490541578374 |
| 2004-02-16 12:02:39 | 0.263491672910365 | -0.05582231332057558 | 0.2036275237310435 | -0.15609087827695273 |
| 2004-02-17 12:02:39 | 0.3083483959130447 | -0.08075817636940523 | 0.2357065083675307 | -0.2103068511381217 |
| 2004-02-18 12:02:39 | 0.3532051189157243 | -0.10569403941823484 | 0.2677854930040179 | -0.2645228239992907 |
| 2004-02-19 12:02:39 | 0.39806184191840394 | -0.13062990246706446 | 0.299864477640505 | -0.3187387968604597 |
| 2004-02-20 12:02:39 | 0.4429185649210836 | -0.15556576551589413 | 0.33194346227699223 | -0.37295476972162867 |
| 2004-02-21 12:02:39 | 0.48777528792376323 | -0.18050162856472377 | 0.3640224469134794 | -0.42717074258279764 |
| 2004-02-22 12:02:39 | 0.532632010926443 | -0.2054374916135534 | 0.3961014315499666 | -0.4813867154439667 |

# CHAPTER 13

# Troubleshooting

## 13.1 Troubleshooting

### 13.1.1 Installation

#### Installation Pre-requisites

Below are the Pre-requisites before installing Fire:

```
JDK 1.8+ installed on the machine

java and jar have to be in the PATH

If running on an Apache Spark cluster, Apache Spark 1.6+ is needed on the cluster.

3GB+ of RAM available on the machine.
```

#### With which user should Fire be installed

If Fire needs to be connected with an Apache Spark cluster the below is needed:

- Fire needs to be installed as a user which can `impersonate other users`. Impersonation for this user has to be set up in `HDFS configs`.
- If you disable impersonation in Fire, then the user with which Fire is installed needs to be able to submit jobs to the cluster.

More Details are available here : https://www.sparkflows.io/connecting-sparkflows-with-spark-cl

#### I do not see anything in my browser after I start Fire

Do check in the logs for exceptions and the root cause. On Linux and Mac, the log files are in nohup.out.

Possible causes are:

- The H2 database was not created and it is failing to the find the table.
- The server did not start properly because some other Application is running on the configured port. The default configured port for Fire is `:8080`

The http and https ports for Fire can be updated in `conf/application.properties`.

#### Fire UI does not get displayed when I go to :8080. Some other UI is displayed

Fire by default runs on `port 8080`. It is possible that you have `some other application running on port 8080`, and you are seeing its output. In this case, the solution is to `run the Fire server on some other port` which is not being used by any other application. Details for running Fire on another port is here : https://www.sparkflows.io/run-fire-on-different-port

### 13.1.2 LDAP

Fire can be configured to authenticate the user with LDAP. Below are some ways to troubleshoot the LDAP configurations.

#### Testing LDAP connection with ldapsearch

It is a good idea to test the ldap environment setup using ldapsearch. This ensures that the machine is setup correctly for LDAP - it can connect to the LDAP server, the LDAP username and passwords are correct, the SSL certificates are good if using LDAPS.

#### Testing Getting User Details from LDAP

- cd to your installation directory
- Create a properties file called `ldaptestconfig.properties`

Below is an example:

```
ldap_attributeUserName=myLdapUsername
ldap_Order="DB_LDAP";
ldap_URL="ldap://localhost:10389";
ldap_base="dc=example,dc=com";
ldap_userDn="uid=john,ou=bindusers,dc=example,dc=com";
ldap_password="johnspassword";
ldap_userSearchBase="ou=sparkflow";
ldap_userSearchFilter="(uid={0})";
ldap_groupSearchBase="ou=groups";
ldap_groupSearchFilter="member={0}";
```

Fetch the user details for the user `xyz` with the following command:

```
java -cp app/fire-ui-3.1.0.jar -Dloader.main=fireui.ldap.LDAPTest org.springframework.
↪boot.loader.PropertiesLauncher xyz
```

#### What if I get locked out

`ldap.Order` determines the order in which Fire tries to log in the user. In case you are locked out of Fire and are not able to log in, you can do the following:

- Add the below line to conf/configuration.properties

ldap.Order=DB

- Then restart the fire server. Now you should be able to log in with your admin account.

Once things are back to normal, you can remove the line you added to `configuration.properties` and restart the fire server.

### 13.1.3 Upgrade

#### Missing column: application_id in FIREDB.PUBLIC.ANALYSIS_FLOW_EXECUTION

After I upgrade to the latest Fire Release I get the error : `Missing column: application_id in FIREDB.PUBLIC.ANALYSIS_FLOW_EXECUTION` or something similar.

After upgrading the Fire Server, it is important to `upgrade the Database Schema`.

- Upgrade it by running `create-h2-db.sh` or `create-mysql-db.sh` from the Fire install directory.
- This would upgrade your DB schema to the latest.

Otherwise you can run into an error like below, when you start the Fire Server:

```
Exception in thread "main" org.springframework.beans.factory.BeanCreationException:
Error creating bean with name 'entityManagerFactory' defined in class path resource ␣
↪[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaAutoConfiguration.
↪class]:
Invocation of init method failed; nested exception is org.hibernate.
↪HibernateException:

Missing column: application_id in FIREDB.PUBLIC.ANALYSIS_FLOW_EXECUTION
```

### 13.1.4 Dataset

#### I am getting an error when clicking 'Update' button on the Create/Update Dataset page

You may see the error below:

```
Unable to retrieve schema for this path :: Bad header for field, should start with a␣
↪character or _ and can contain only alphanumerics and _ 0:" id 1 "
```

- This is because one of the column names of the header is not in the right format. In this case the column name `id 1` contains a space.
- Only `alphanumerics and _` are permitted in the header and column names.
- If your data does not have a header column, set the `Header` field to `false` when defining the Dataset.

### 13.1.5 Running Workflows

#### Getting Exception : 'User: ec2-user is not allowed to impersonate ec2-user

Sparkflows impersonates the logged in user when submitting the jobs onto the Cluster.

So, the user with which Sparkflows is running has to be configured on HDFS as a proxy user.

Details for allowing the sparkflows user to impersonate other users is available at:

- ../installation-upgrading/connecting-spark-cluster

#### When running the workflows on my Spark Cluster, results are not showing up in the Browser

This is probably because there is some configuration error. Sparkflows uses spark-submit to submit the jobs to the cluster. The driver of the spark job posts back results to the Fire server.

- Check out the log for spark-submit for the workflow in `/tmp/fire/workflowlogs` to find the root cause. Maybe the spark job is just failing.
- It is also useful to ensure Spark jobs can be submitted to the Cluster from the machine on which Sparkflows is running with spark-submit. Submit the `SparkPi` job from `spark-examples.jar` to test it.
  - `SparkPi` can be run with a command like : `spark-submit --class org.apache.spark.examples.SparkPi --master yarn --deploy-mode client spark-examples.jar 10`
  - `spark-examples.jar` is in your Apache Spark install direction on the machine.
- If the Spark job is running successfully (according to the logs), but the results are still not showing up in the Browser, it could be because the fire spark job is unable to post results back to the Fire web server. You should see these failures in the logs.
  - Under Administration/Configuration, there is the config `app.postMessageURL`. It determines the Fire URL to which the results from the spark driver are posted back to the fire server. Ensure that it is set up correctly.

#### Getting Exception: org.apache.hadoop.security.AccessControlException: Permission:denied : user=admin

When running on the Cluster, you are running into the exception below:

```
org.apache.hadoop.security.AccessControlException: Permission denied: user=admin,␣
↪access=WRITE, inode="/user":hdfs:supergroup:drwxr-xr-x
```

- If the above exception is coming up when running the workflow, then it means that the logged in user does not exist on HDFS.
- In the above case, the user is logged into Fire as `admin`. So the jobs submitted by Fire on the cluster is as the user `admin`. But the user 'admin' does not exist on HDFS.
- Please make sure to `log into Fire as a user which exists on HDFS`.

#### When running the example workflows on the Spark Cluster it is not able to find the input files

The example workflows read in input files.

- They have to be on HDFS in the home directory of the logged in user.

- The data directory which comes with Sparkflows has to be uploaded onto HDFS.
- For example, if the logged in user is `john`, then the data directory would be on HDFS in the directory `/user/john`

### Getting Exception : Server returned HTTP response code: 405 for URL: http://10.125.221.72:8080/messageFromSparkJob

When submitting jobs to the cluster from Fire, you are running into the exception below:

```
Sending 'POST' request to URL : http://10.125.221.72:8080/messageFromSparkJob

Response Code : 405

java.io.IOException: Server returned HTTP response code: 405 for URL: http://10.125.
↪221.72:8080/messageFromSparkJob

at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)

at sun.reflect.NativeConstructorAccessorImpl.
↪newInstance(NativeConstructorAccessorImpl.java:62)

at sun.reflect.DelegatingConstructorAccessorImpl.
↪newInstance(DelegatingConstructorAccessorImpl.java:45)

at java.lang.reflect.Constructor.newInstance(Constructor.java:423)

at sun.net.www.protocol.http.HttpURLConnection$10.run(HttpURLConnection.java:1944)

at sun.net.www.protocol.http.HttpURLConnection$10.run(HttpURLConnection.java:1939)
```

Fire submits Spark jobs to the cluster. The spark driver, posts certain results back to the Fire server to be displayed to the user.

The cause of this error is that the postback-url has not been set correctly - `http://10.125.221.72:8080/messageFromSparkJob`

There could be following issues with the URL:

```
The machine name/IP is wrong. It has to be the machine on which Fire is running.

The port number is wrong. Fire server is running on another port on the machine.
```

### Getting Exception : java.lang.ClassNotFoundException: fire.execute.WorkflowExecuteFromFile

When running the jobs on the cluster, you are running into the exception below.

- The reason for it is that the `app.sparkSubmitJar` is not set up correctly. Fire comes with a jar file which gets submitted to the cluster with spark-submit. app.sparkSubmitJar has to correctly point to this jar file.
- You can go under `Administration/Configuration` to set it up correctly.

Exception:

```
Warning: Local jar /home/ec2-user/fire-2.1.0/fire-lib/fire-spark_1_6-core-2.1.0-jar-
↪with-dependencies.jar does not exist, skipping.
java.lang.ClassNotFoundException: fire.execute.WorkflowExecuteFromFile at java.net.
↪URLClassLoader.findClass(URLClassLoader.java:381) at
```

(continues on next page)

(continued from previous page)

```
java.lang.ClassLoader.loadClass(ClassLoader.java:424) at java.lang.ClassLoader.
↪loadClass(ClassLoader.java:357) at
java.lang.Class.forName0(Native Method) at java.lang.Class.forName(Class.java:348) at
org.apache.spark.util.Utils$.classForName(Utils.scala:177) at
org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$
↪$runMain(SparkSubmit.scala:688) at
org.apache.spark.deploy.SparkSubmit$$anon$1.run(SparkSubmit.scala:163) at
org.apache.spark.deploy.SparkSubmit$$anon$1.run(SparkSubmit.scala:161) at java.
↪security.AccessController.doPrivileged(Native Method) at
javax.security.auth.Subject.doAs(Subject.java:422) at
org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1917)␣
↪at
org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:161) at
org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:206) at org.apache.
↪spark.deploy.SparkSubmit$.main(SparkSubmit.scala:121) at
org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

#### Getting Exception on HDInsight : No FileSystem for scheme: wasbs

When running the jobs on the cluster, you are running into the exception below.

- The reason for it is that it is not understanding the scheme `wasb`. In order to fix it, run `./run-fire-spark-submit.sh start` instead of `./run-fire.sh start`.
- This enables getting the distribution libraries into the executable.

Exception:

```
Error : java.io.IOException: No FileSystem for scheme: wasbs at
org.apache.hadoop.fs.FileSystem.getFileSystemClass(FileSystem.java:2586) at
org.apache.hadoop.fs.FileSystem.createFileSystem(FileSystem.java:2593) at
org.apache.hadoop.fs.FileSystem.access$200(FileSystem.java:91) at
org.apache.hadoop.fs.FileSystem$Cache.getInternal(FileSystem.java:2632)
```

### 13.1.6 Fire Server & Workflow Execution Logs

#### Where do I find the logs of the Fire Server

When running on linux or mac the logs of the Fire Process are in the file `fire.log`. Logs of the Fire Web Server are in the file `fireserver.log` under the directory where Fire has been installed. It would be something like `...../fire-2.1.0`

#### Where do I find the logs of the workflows when running on my Cluster

The logs are in the directory `/tmp/fire/workflowlogs` on the machine on which the Fire server is running:

```
Each workflow execution has its own log file.
```

The json representation of the workflow is in `/tmp/fire/workflows` when running in `YARN client mode`. They are in `.fireStaging` directory under the `users home directory on HDFS` when running in `YARN cluster mode`.

### 13.1.7 Dashboards

#### When viewing the Dashboard the cells are showing up empty

Dashboards show output of Workflows.

If the corresponding workflow has not executed, the content in the Dashboard would show up as empty.

### 13.1.8 Kerberos

#### My cluster is Kerberised. How do I setup Sparkflows for it

The steps to setup Sparkflows on a Kerberised cluster are at:

- ../installation-upgrading/configuration/configuring-kerberos

### 13.1.9 Python Installation

Python installations from source with version 3.6.5

#### showing warning message with missing package while restarting pyspark server

showing warning message with missing package while restarting pyspark server:

```
UserWarning: Could not import the lzma module. Your installed Python is incomplete
```

#### Possible Solution

For centos: Install development tool:

```
sudo yum install -y xz-devel
```

Recompile python from source code:

```
cd Python-3.6.5
sudo ./configure --enable-optimizations
sudo make altinstall
```

CHAPTER 14

# Frequently Asked Questions

## 14.1 FAQ

### 14.1.1 Scheduling Workflows

#### How can I schedule the workflows I create ?

Fire Insights saves workflow definitions as JSON files. These workflows are executed through spark-submit.

Fire Insights has a scheduler which allows Workflows to be scheduled at regular intervals.

Since the workflows are submitted with spark-submit, they can also be easily scheduled with Oozie, crontab etc.

### 14.1.2 Custom Nodes

#### Does Fire Insights allow me to create my own custom nodes?

Yes, new Nodes can be easily to added to Fire Insights. Develop nodes in Java or in Scala and dop the definition JSON for the node on the server. The newly added nodes will become visible in the Fire Insights User Interface.

### 14.1.3 Distributions Supported

#### What distributions or platforms are supported with Sparkflows?

Sparkflows Fire has been tested with CDH, Hortonworks, MapR, AWS EMR, Apache Spark distributions.

Note: Any cluster with Apache Spark 1.6+ will work fine with Sparkflows.

#### Can I run Sparkflows on my Amazon AWS cluster or Microsoft Azure or Google Cloud?

Yes, all Sparkflows needs for successful deployment is a Apache Spark cluster. Sparkflows is deployed on the edge node of the cluster.

### 14.1.4 Workflow Export - Import

#### How does one export/import workflows between instances?

Sparkflows allows workflows to be exported and imported. Workflows are represented as JSON files and hence can also be checked into github etc. for versioning.

Sparkflows also maintains the version history of the workflows.

### 14.1.5 Submit Apache Spark Jobs

#### When running on a Apache Spark cluster how does Sparkflows submit the spark jobs?

Fire Insights uses spark-submit to submit the Apache Spark jobs to the cluster. Hence it is important that spark-submit work from the machine on which Fire Insights is installed.

### 14.1.6 Multi User Support

#### How does the Sparkflows platform handle multi-user support (i.e. Can user 1 see or edit user 2's data sources, pipelines, etc)

Sparkflows supports various user types and enables users to easily share datasets and workflows with each other to foster collaboration.

### 14.1.7 Data Sources

#### How does one define a new data source and establish a connection?

Sparkflows platform has various OOTB connectors to HIVE, Flume, Kafka, HBase, Solr. For all other structured or unstructured datasets on HDFS or CloudBricks, Sparkflows platform can identify the schema on the fly when a new dataset is created in Sparkflows pointing to a data source. The schema can be updated right there as well. Sparkflows workflow execution writes a summary of its output to MySQL/Oracle/H2 which is accessible by the users of the system.

### 14.1.8 Hadoop Installation Pre-Requisites

Below are the pre-requisites for installing Hadoop:

- Linux
- JDK 1.8 installed
- IPV6 disabled
- Selinux disabled

### Linux

Minimum machine configuration:

- vCPU : 8 vcores
- RAM: 32 GB

### JDK

JDK 8 is needed on the Linux Machine. Below are the steps for installing oracle java:

- Install java 8 as the root user
- http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- wget –no-cookies –no-check-certificate –header "Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie" "https://download.oracle.com/otn-pub/java/jdk/8u201-b09/42970487e3af4f5aa5bca3f542482c60/jdk-8u201-linux-x64.rpm"
- yum localinstall jdk-8u201-linux-x64.rpm

Ensure that java 8 is installed properly:

- java -version


Set the below in .bash_profile

- export JAVA_HOME=/usr/java/jdk1.8.0_201-amd64/

### Disable IPV6

- Edit file /etc/sysctl.conf - vi /etc/sysctl.conf

Add the following lines:

- net.ipv6.conf.all.disable_ipv6 = 1
- net.ipv6.conf.default.disable_ipv6 = 1

Execute the following command to reflect the changes.

- sysctl -p

### Selinux

Just ensure that selinux should be disabled so that it cant impact Hadoop performance.

- sudo setenforce 0

To disable it permanently

- edit /etc/selinux/config

SELINUX=disabled

- reboot

### Steps Involved in Installing Hadoop

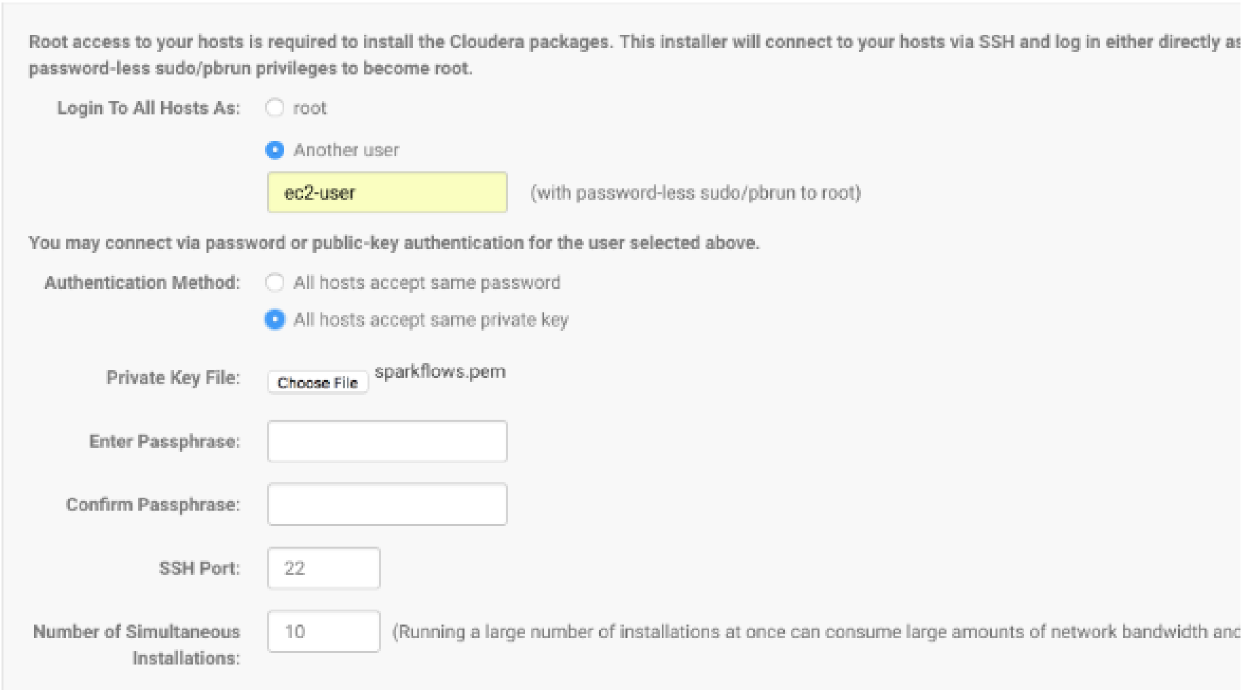
- Install bind-utils : Otherwise Cloudera Manager gives **host not found**
  - yum install bind-utils
- Install Cloudera Manager
  - cd
  - wget https://archive.cloudera.com/cm5/installer/latest/cloudera-manager-installer.bin
  - chmod u+x cloudera-manager-installer.bin
  - ./cloudera-manager-installer.bin
  - Accept Licenses
- Open ports on Linux Machine
  - Open the ports 7180 and 8080

### After Installation of Cloudera Manager

- go to http://host-ip:7180/
  - Log in with admin/admin
  - Select Cloudera Express Installation
  - For host, give the hostname IP (private IP)
  - Install using Parcels
  - Include the Kafka parcels
  - User : sparkflows ( As per as updated on machine while creating Linux Machine)
  - Supply the private key
  - Install Core with Spark
  - Update default Configurations in it.

### Add proxy user in HDFS

- Add sparkflows as proxy user in HDFS
  - https://www.sparkflows.io/connecting-sparkflows-with-spark-cl
  - Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml
    - hadoop.proxyuser.sparkflows.hosts
    - hadoop.proxyuser.sparkflows.groups
- Restart Cluster services




### Create HDFS directory

Create HDFS directory for sparkflows user (we can create as per as requirements)

- sudo su
- su hdfs
- hadoop fs -mkdir /user/sparkflows
- hadoop fs -chown sparkflows:sparkflows /user/sparkflows

### Install Spark2

spark2 is installed using CSD or Parcels

- https://www.cloudera.com/documentation/spark2/latest/topics/spark2_installing.html
  - cd /opt/cloudera/csd
  - sudo su
  - wget http://archive.cloudera.com/spark2/csd/SPARK2_ON_YARN-2.1.0.cloudera2.jar
  - chown cloudera-scm:cloudera-scm SPARK2_ON_YARN-2.1.0.cloudera2.jar
  - chmod 644 SPARK2_ON_YARN-2.1.0.cloudera2.jar
  - service cloudera-scm-server restart

### Login Again into Cloudera Manager

- In Cloudera Manager:
  - Go to Hosts/Parcels

  - Download Spark2
  - Distribute Spark2
  - Activate Spark2
- Add Spark2 service in Cloudera Manager
  - Go to Cluster/Add Service
  - Add Spark2 Service
  - For dependency select one with HIVE etc.
  - Select the host

### In YARN increase Container memory to 8GB

- yarn.scheduler.maximum-allocation-mb
- yarn.nodemanager.resource.memory-mb

### AFTER INSTALLATION GET CDH TO USE JAVA 8

- In Spark configuration in Cloudera Manager set the below for spark-defaults.conf
  - spark.executorEnv.JAVA_HOME=/usr/java/jdk1.8.0_201-amd64/
  - then redeploy the client configurations
  - Restart the cluster service

### Install Sparkflows

- ssh to the machine
- wget https://s3.amazonaws.com/sparkflows-release/fire/rel-x.y.z/2/fire-x.y.z.tgz
- tar xvf fire-x.y.z.tgz
- cd fire-x.y.z
- ./create-h2-db.sh
- ./run-fire.sh start
- ./run-fire-server.sh start

### Upload the Fire Insights example data directory onto HDFS

- As sparkflows user
- cd fire-x.y.z
- hadoop fs -put data

### Log into Fire Insights

- http://host-ip:8080/#/dashboard
  - Log in with admin/admin
  - Create user sparkflows in Sparkflows. Give it admin rights. Add to group default, save it.
  - Again Login with sparkflows user.
  - Go to Configurations under administration and click on infer hadoop cluster config and save it.
  - open spark and update spark2-submit under "spark.spark-submit" and save it.
  - Create a workflow and execute it.

CHAPTER 15

# Administration

## 15.1 Administration Guide

### 15.1.1 User Administration

Fire allows you to create and manage

- Users
- Groups
- Roles

These are accessible under the Administration Menu.

#### Users

- Fire allows you to create and edit users
- Users belong to groups and have roles
- A user can be a designated as a superuser
- The user should exist on HDFS (when running against a Hadoop Cluster). Fire can run independent of a Hadoop Cluster.

#### Groups

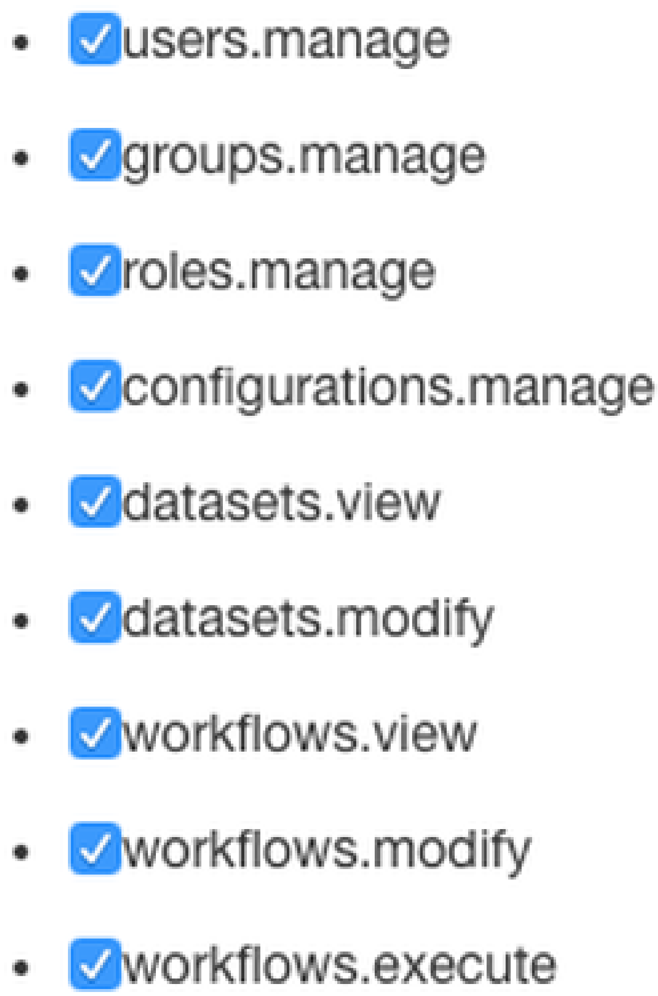
- Fire allows you to create and edit groups
- Groups allow users to share Datasets, Workflows and Dashboards with other groups

### Roles

- Fire allows you to create and edit roles.
- A role has various permissions associated with it.

### Permissions

Fire has the following permissions defined.



CHAPTER 16

# Databricks Integration

## 16.1 Databricks Guide

### 16.1.1 Databricks Prerequisites

Below are the Prerequisites for installing Fire Insights on a Databricks Cluster:

Table 1: Below are the Needed Package

| Package | Description | Value |
|---|---|---|
| Python version | python version on Databricks Cluster | 3.6.0 or above |
| pip version | pip version on Databricks Cluster | 20.0 or above |
| Spark version | Spark Version on Databricks Cluster | 2.4 |
| Fire Running Port | Port on Which Fire is Running | Accessible from databricks Cluster |

### 16.1.2 Databricks Integration Steps

Fire Insights integrates with Databricks. It submits jobs to the Databricks clusters using the REST API of Databricks and have the results displayed back in Fire Insights.

Fire also fetches the list of Databases and Tables from Databricks, making it easier for the user to build their workflows and execute them. In addition fire displays the list of Databricks clusters running for the user.

Databricks can be running on Azure or on AWS.

- Running Databricks on Azure : https://docs.microsoft.com/en-us/azure/azure-databricks/quickstart-create-databricks-workspace-portal
- Running Databricks on AWS : https://databricks.com/aws

Below are the steps for Integrating Fire Insights with your Databricks Clusters.

## Install Fire Insights

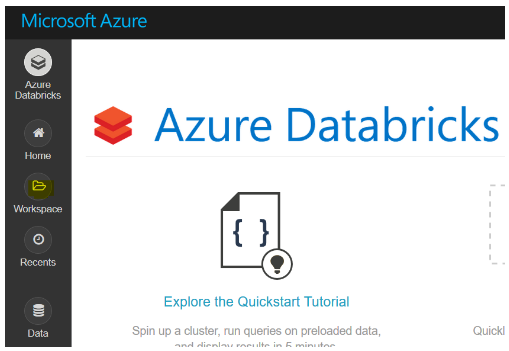
Install Fire Insights on any machine. The machine has to be reachable from the Databricks cluster.

## Upload Fire Core Jar to Databricks

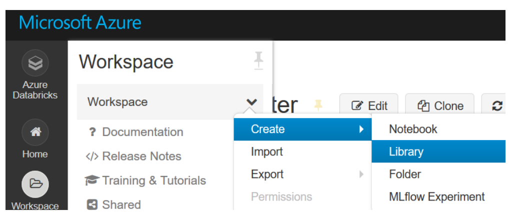
Fire Insights jar has to be uploaded to Databricks. Fire Insights jobs running on Databricks make use of this jar file.

Upload `fire-x.y.z/fire-core-lib/fire-spark_2_3-core-3.1.0-jar-with-dependencies.jar` to Databricks. Upload it under Workspace as a Library on to Databricks.

1. Login to `Databricks Cluster`
2. Click on `workspace` in the left side pane




3. Create a new **Library**



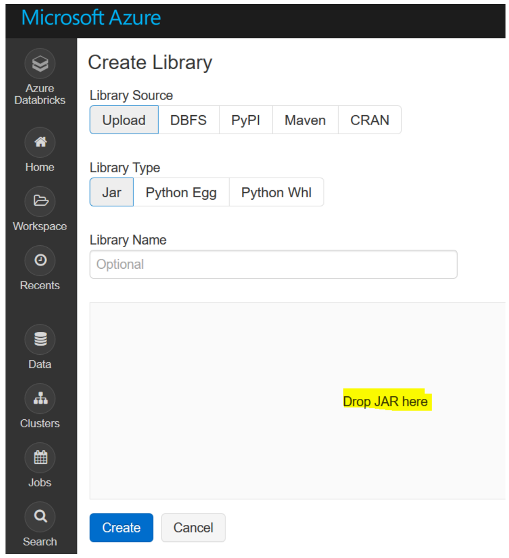

4. Upload `fire-spark_2_4-core-3.1.0-jar-with-dependencies.jar` from your machine by Clicking on `Drop JAR here`
5. Once `fire-spark_2_4-core-3.1.0-jar-with-dependencies.jar` is uploaded, click on `Create`

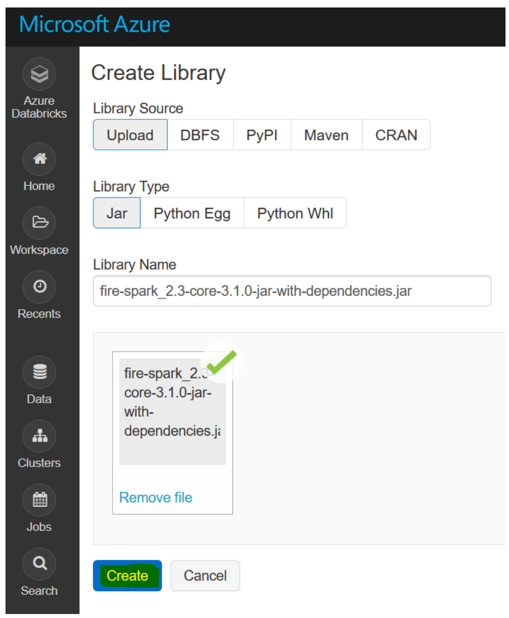
- Check the box with `Install automatically on all clusters`, in order to avoid installing it manually to every cluster.

## Configure the Uploaded Library in Fire Insights

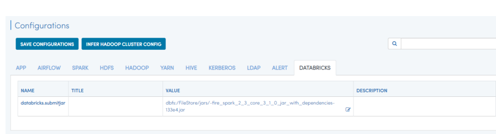
Configure the path of the uploaded fire core jar library in Databricks in Fire Insights.

This has to be done under Administration/Configuration.


Microsoft Azure
Azure Databricks
Home
Workspace
Recents
Data
Clusters
Jobs
Search
Create Library
Library Source
Upload
DBFS
PyPI
Maven
CRAN
Library Type
Jar
Python Egg
Python Whl
Library Name
Optional
Drop JAR here
Create
Cancel

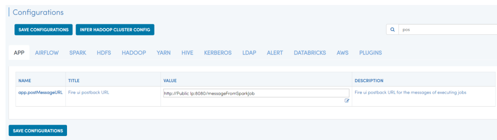

Microsoft Azure
Azure Databricks
Home
Workspace
Recents
Data
Clusters
Jobs
Search
Create Library
Library Source
Upload
DBFS
PyPI
Maven
CRAN
Library Type
Jar
Python Egg
Python Whl
Library Name
fire-spark_2.3-core-3.1.0-jar-with-dependencies.jar
Remove file
Create
Cancel


Microsoft Azure
fire-spark_2.3-core-3.1.0-jar-with-dependencies.jar
fire-spark_2.3-core-3.1.0-jar-with-dependencies.jar
Files
fire_spark_2_3_core_3_1_0_jar_with_dependencies.jar
Source
dbfs:/FileStore/jars/ccbd9fc8_e256_4b10_9ab3_6562fdf4b9ce-fire_spark_...
Copy
Status on running clusters
Install automatically on all clusters
Uninstall
Install


Configurations

### Configure app.postMessageURL in Fire Insights

Configure `app.postMessageURL` to be the IP of the machine on which Fire Insights is installed. Jobs running on Databricks would post back results to Fire Insights using this URL.




### Install Databricks JDBC Driver

Fire needs the Databricks JDBC Driver to be installed. Install it in the `fire-user-lib` and `fire-server-lib` folder of the Fire installation.

You can download the Databricks JDBC Driver from the Databricks site :

- https://docs.databricks.com/bi/jdbc-odbc-bi.html
- https://databricks.com/spark/odbc-driver-download

The driver is available as a zip file. eg: `SimbaSparkJDBC-2.6.3.1003.zip`

- Unzip the downloaded file. It will create a directory like `SimbaSparkJDBC-2.6.3.1003`
- Copy the jdbc jar file named `SparkJDBC4.jar` into `fire-x.y.z/fire-user-lib` and `fire-x.y.z/fire-server-lib`

### Create your REST API token in Databricks

Create your token in Databricks. It would be used in making REST API calls to Databricks from Fire Insights.

1. Login to your Databricks Account
2. Click on `Account` icon in right corner top
3. Click on `User Settings`
4. Click on `Generate New Token`
5. Add `comment` & `Lifetime(days)` for token expiry & Click on `Generate`
6. Copy the token generated. Click on `DONE`

### Create Databricks Connection in Fire Insights

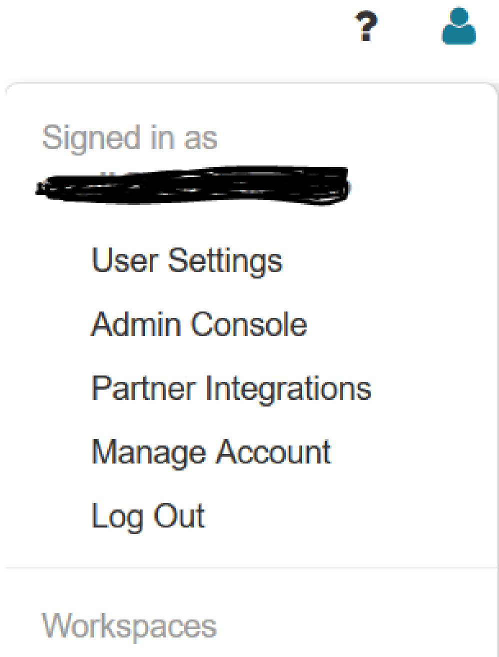
Create a connection in Fire Insights to Databricks.

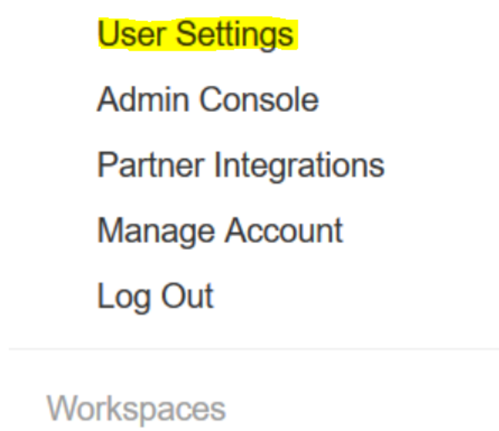
It can be created by the Administrator under Administration/Global Connections. These connections are available for everyone to use.

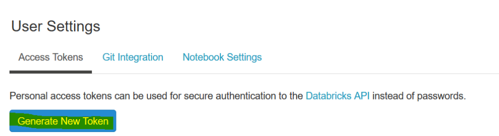
It can also be created by any user with their Application. In this case, it is only available to the Application and its users.

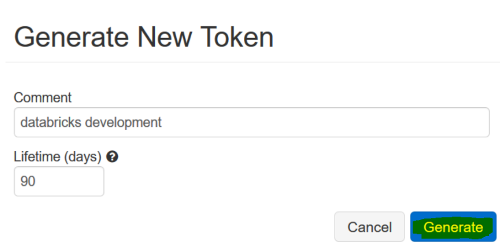
- Specify your Databricks Token.
- Specify the Databricks JDBC URL of your cluster in Databricks.

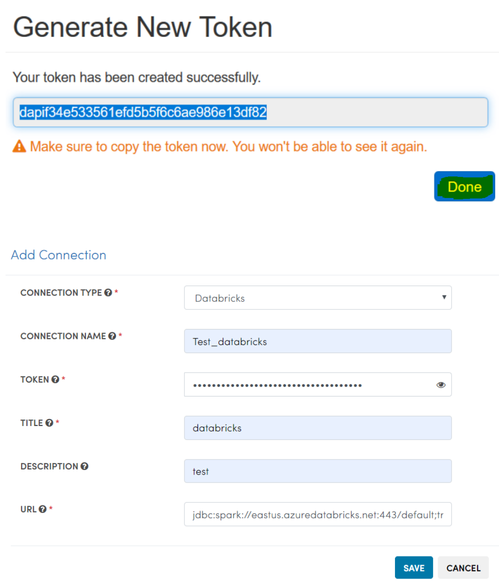
Now we are ready to start using the Databricks Connection in Fire Insights to:


?
Signed in as
[redacted]
User Settings
Admin Console
Partner Integrations
Manage Account
Log Out
Workspaces


User Settings
Admin Console
Partner Integrations
Manage Account
Log Out
Workspaces


User Settings
Access Tokens
Git Integration
Notebook Settings
Personal access tokens can be used for secure authentication to the Databricks API instead of passwords.
Generate New Token


Generate New Token
Comment
databricks development
Lifetime (days)
90
Cancel
Generate




- Browse DBFS
- View your Databricks Clusters
- Browse your Databricks Databases & Tables
- Create Workflows which Read from and Write to Databricks

### 16.1.3 Databricks Python Integration Steps

Fire Insights integrates with Databricks and can submit Python jobs. It submits jobs to the Databricks clusters using the REST API of Databricks and have the results displayed back in Fire Insights.

Below are the steps for Integrating Fire Insights with your Databricks Clusters for running Python jobs.

---

**Note:** The Machine on which Fire Insights is installed should have Python 3.7.0 or above.

---

Python Installation Steps:

- https://docs.sparkflows.io/en/latest/installation/python-install-linux.html

#### Install Fire Insights

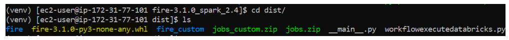
Install Fire Insights on your machines. The machine has to be reachable from the Databricks cluster.

#### Upload Fire wheel file to Databricks

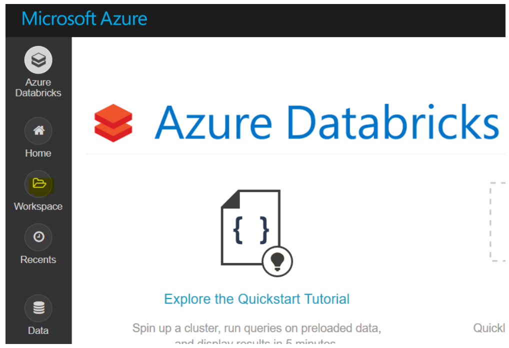
Fire Insights wheel file has to be uploaded to Databricks. Fire Insights jobs running on Databricks make use of this wheel file.

Upload `fire-x.y.z/dist/fire-3.1.0-py3-none-any.whl` to Databricks. Upload it under Workspace as a Library on to Databricks under DBFS or even in S3 Bucket which is accessible from the Databricks Cluster.

1. Login to `Databricks Cluster`

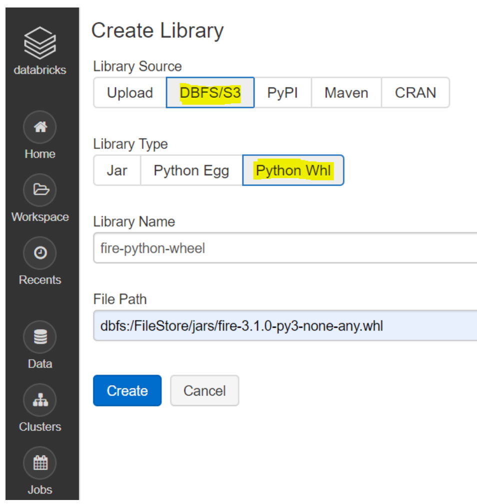



2. Click on `workspace` in the left side pane




3. Create a new Library

You can select Library Source as `DBFS`, Library Type as `Python Whl`, provide any `Library Name` field, & add File Path of `fire-3.1.0-py3-none-any.whl` located in DBFS.




On Clicking on `Create` button it will ask to install on specific databricks Cluster, select cluster on which you want to install.

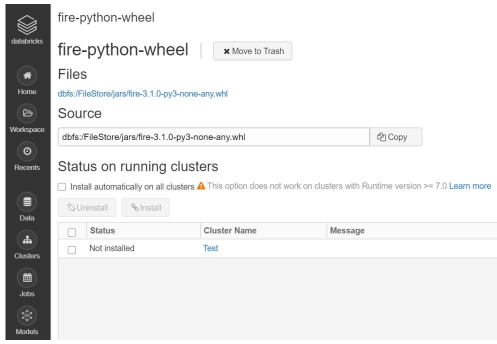
On Successfull installation of wheel file on Databricks Cluster, it would be displayed under `Libraries`.

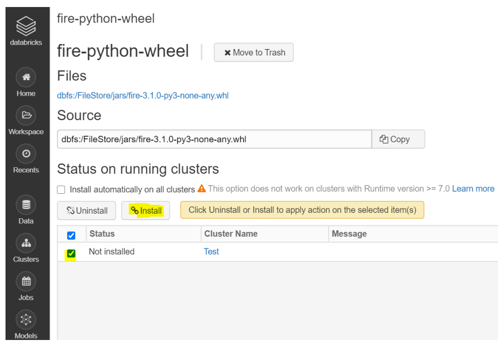
Another option is to upload `fire-3.1.0-py3-none-any.whl` file to s3 Bucket which is accessible from Databricks Cluster.

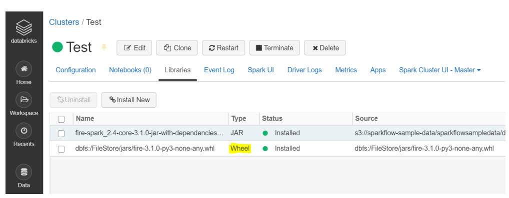
Once you upload `fire-3.1.0-py3-none-any.whl` file to s3 Bucket, login to Databricks Cluster & inside Libraries tab.

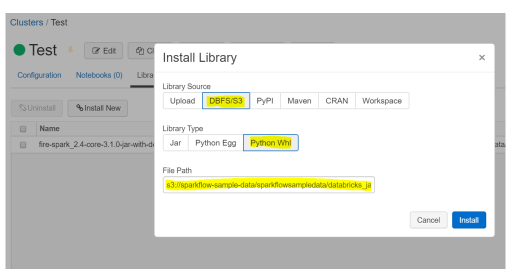
Install New Library & select `DBFS/S3` in Library Source, `Python Whl` in Library Type and copy paste the location of python wheel file available in s3 in File Path & Click on Install.

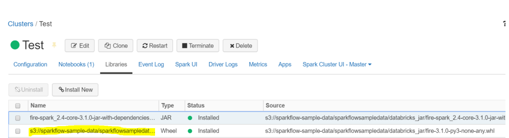
Once it is installed successfully, you can see the python wheel inside Library is up.


fire-python-wheel
fire-python-wheel
Move to Trash
Files
dbfs:/FileStore/jars/fire-3.1.0-py3-none-any.whl
Source
dbfs:/FileStore/jars/fire-3.1.0-py3-none-any.whl
Copy
Status on running clusters
Install automatically on all clusters
This option does not work on clusters with Runtime version >= 7.0
Learn more
Uninstall
Install
Status
Cluster Name
Message
Not installed
Test


fire-python-wheel
fire-python-wheel
Move to Trash
Files
dbfs:/FileStore/jars/fire-3.1.0-py3-none-any.whl
Source
dbfs:/FileStore/jars/fire-3.1.0-py3-none-any.whl
Copy
Status on running clusters
Install automatically on all clusters
This option does not work on clusters with Runtime version >= 7.0
Learn more
Uninstall
Install
Click Uninstall or Install to apply action on the selected item(s)
Status
Cluster Name
Message
Not installed
Test


Clusters / Test
Test
Edit
Clone
Restart
Terminate
Delete
Configuration
Notebooks (0)
Libraries
Event Log
Spark UI
Driver Logs
Metrics
Apps
Spark Cluster UI - Master
Uninstall
Install New
Name
Type
Status
Source
JAR
Installed
Wheel
Installed
dbfs:/FileStore/jars/fire-3.1.0-py3-none-any.whl

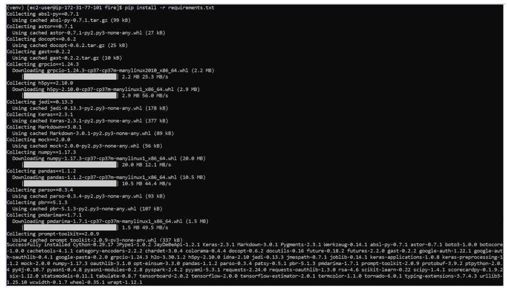

Install Library
Library Source
Upload
DBFS/S3
PyPI
Maven
CRAN
Workspace
Library Type
Jar
Python Egg
Python Whl
File Path
s3://sparkflow-sample-data/sparkflowsampledata/databricks_ja
Cancel
Install



### Install Python dependencies

You need to install the python dependencies required by Fire Insights on the machine by running below Command from `fire-x.y.z/dist/fire/` directory:

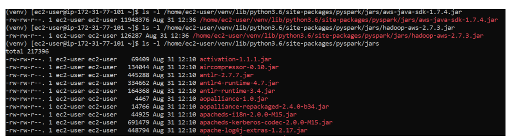
```
pip install -r requirements.txt
```



Note: Make sure that pip etc. is already installed on that machine

### Install dependency for AWS

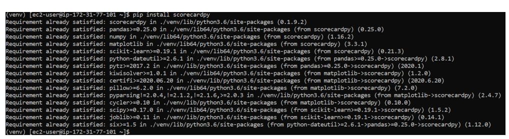
Copy the jars `hadoop-aws` and `aws-java-sdk` to pyspark jar path.



Install any specific package of python, if Need to use in Custom Processors on databricks Cluster aswellas Fire Insights Machine.

Use the command below to install it on the Fire Insights machine:

```
pip install scorecardpy
```



Install it on your Databricks cluster with the below:

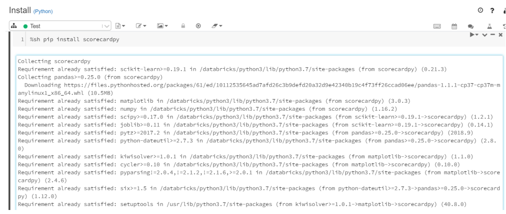
```
* Open a Notebook and attach to Databricks Cluster.
* %sh pip install scorecardpy
```

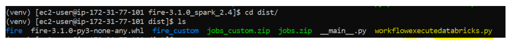
### Upload Fire workflowexecutedatabricks.py file to DBFS

For Python Job submission to Databricks Cluster.

Upload `fire-x.y.z/dist/workflowexecutedatabricks.py`, file to DBFS or even S3 Bucket too.

You can `UPLOAD` it, using DBFS Browser too.





### Configure the Uploaded Library in Fire Insights

Configure the path of the uploaded fire python wheel package file & workflowexecutedatabricks.py under `databricks.pythonFile` & `databricks.pythonPackages` respectively in Fire Insights.

It can be two source either `DBFS` or `S3` path.

If you have Uploaded in `DBFS` path.

If you have Uploaded in `S3` path.

### Job Submission using Pyspark Engine

Now You can submit pyspark jobs to Databricks Cluster from Fire Insights.

## 16.1.4 Databricks User Guide

### Browsing Databricks Tables

Fire Insights allows you to Browse your Databricks Databases & Tables.

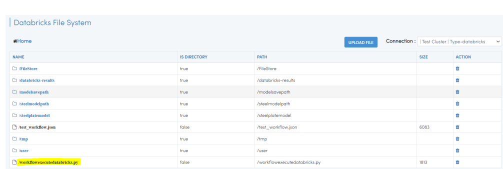
### Go to Data/Databricks DB

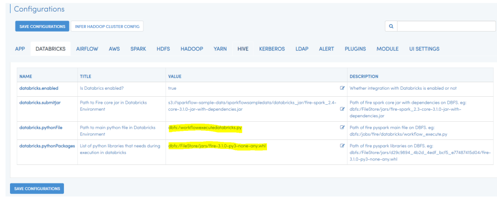
It will display the Databricks DB page.

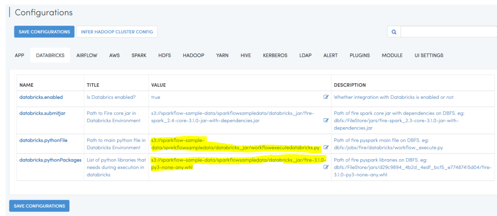
### Select the Tables

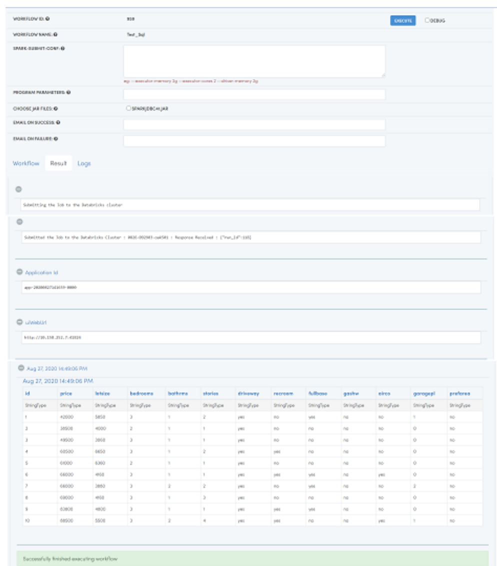
Once you select the `Tables`, right click on it to get the query to view the first few records from the table.

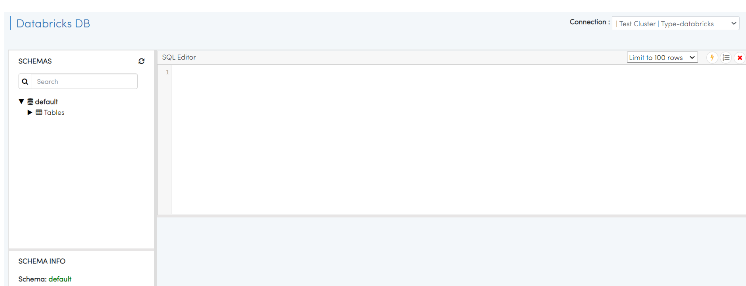
Execute the sql query to view records from the table selected.

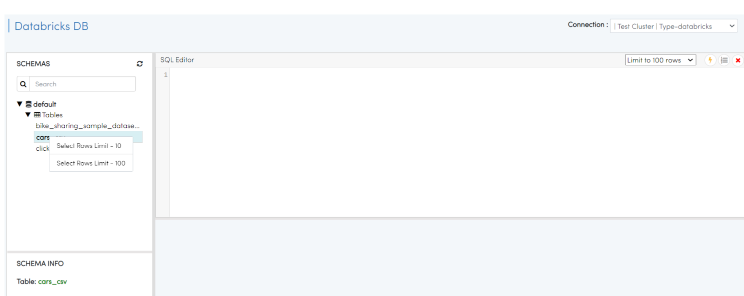


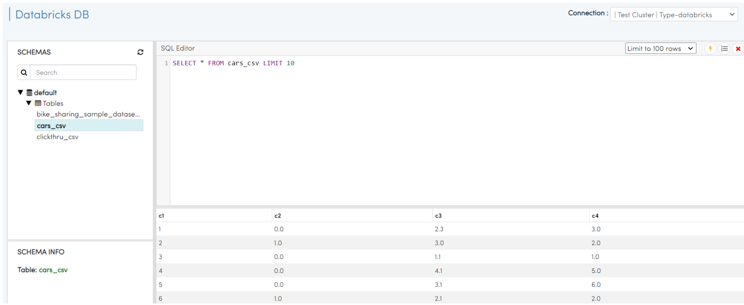

Configurations
SAVE CONFIGURATIONS
APP DATABRICKS AIRFLOW AWS SPARK HDFS HADOOP YARN HIVE KERBEROS LDAP ALERT PLUGINS MODULE UI SETTINGS


Configurations
SAVE CONFIGURATIONS
INFER HADOOP CLUSTER CONFIG
APP DATABRICKS AIRFLOW AWS SPARK HDFS HADOOP YARN HIVE KERBEROS LDAP ALERT PLUGINS MODULE UI SETTINGS
NAME TITLE VALUE DESCRIPTION
databricks.enabled
databricks.submitJar
databricks.pythonFile
databricks.pythonPackages


Workflow Result Logs
Successfully finished executing workflow


Databricks DB
Connection :
Test Cluster | Type-databricks
SCHEMAS
SQL Editor
Limit to 100 rows
Search
default
Tables
SCHEMA INFO
Schema: default


Databricks DB
Connection :
Test Cluster | Type-databricks
SCHEMAS
SQL Editor
Limit to 100 rows
Search
default
Tables
bike_sharing_sample_datase...
Select Rows Limit - 10
Select Rows Limit - 100
SCHEMA INFO
Table: cars_csv



## Running DDL Commands

Fire Insights allows you to run DDL commands on Databricks.

With this one can:

- Create New Databases
- Create New Tables
- View the schema of the tables
- And many more

Go to DATABROWSERS/Databricks DB. Then click on DDL.

Databricks has a good page on Creating New Tables:

https://docs.databricks.com/spark/latest/spark-sql/language-manual/create-table.html

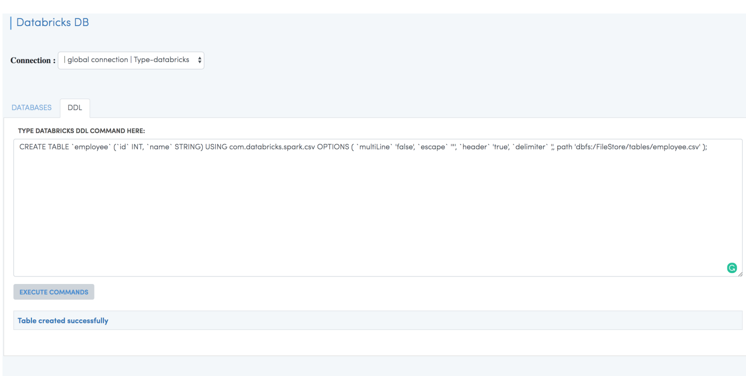
## Below are example of running DDL

## Creating Table

- DDL Statement:

```
CREATE TABLE `employee` (`id` INT, `name` STRING) USING com.databricks.spark.csv␣
↪OPTIONS ( `multiLine` 'false', `escape` '"', `header` 'true', `delimiter` ',',␣
↪path 'dbfs:/FileStore/tables/employee.csv' );
```

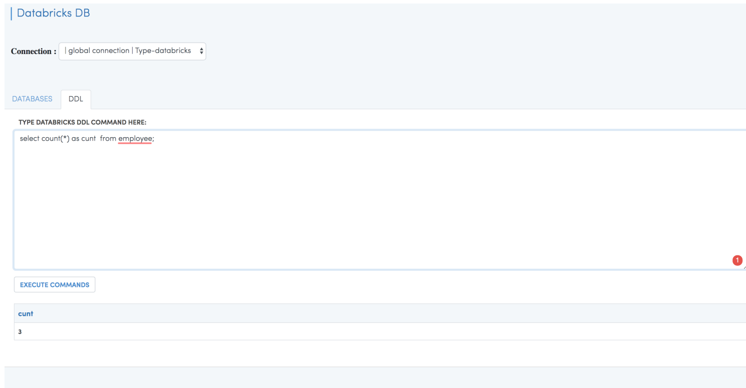
Location of the data could be changed to S3 location.



### Running SQL

- Select SQL Statement:

```
select count(*) as count  from employee;
```

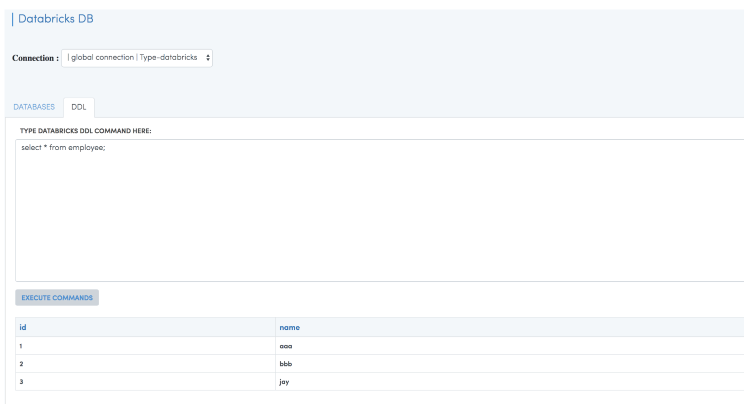



### Sample Data:

- Select SQL Statement:

```
select * from employee;
```

By default first 100 rows of data is displayed.



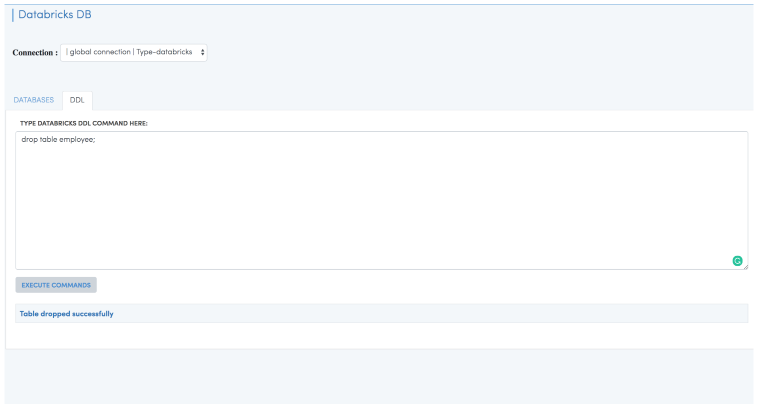

### Drop Table

- Drop Statement:

```
drop table employee;
```

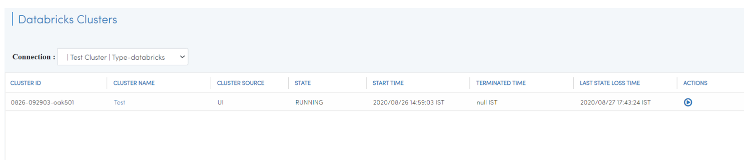
### Viewing Databricks Clusters

Fire Insights enables you to view your Databricks Clusters. You can also Start and Stop the Databricks clusters from Fire Insights.

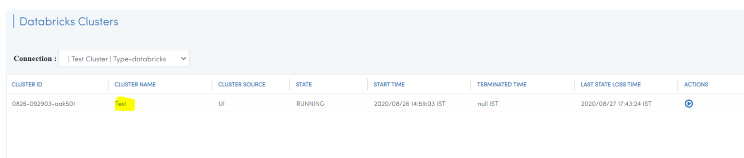



### Go to Data Browsers/Databricks Clusters

It will display the various Databricks Clusters available.




If you want to see Cluster Details, Click on `CLUSTER NAME`, it will display all informations.




You can also Start and Stop the Databricks clusters from Fire Insights, using `ACTIONS` button.

### Browse DBFS

Fire Insights enables you to browse your DBFS & UPLOAD FILE & Delete file and directory in DBFS.

### Go to DATA BROWSERS/DBFS

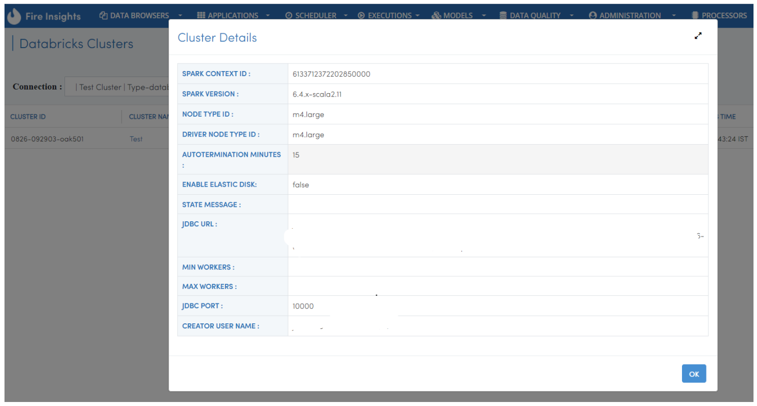
It will display the Databricks File System list page.

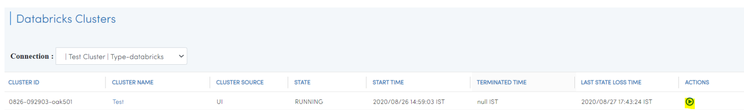
### UPLOAD FILE in DBFS

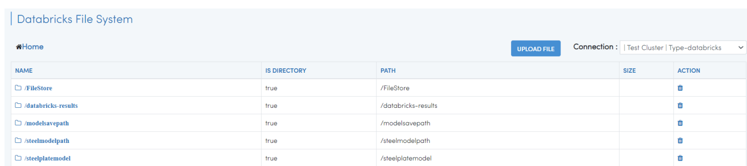
You can upload file in `DBFS` from local pc.

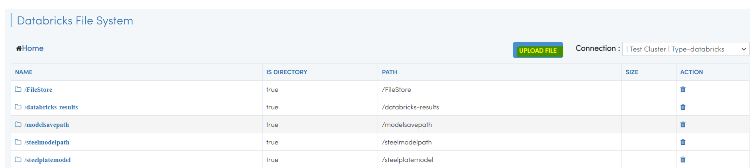
On clicking on `UPLOAD FILE` button, it will ask you to select file from local pc and UPLOAD.

On successful `UPLOAD`, it will show successful informations and file can be viewed inside the folder in `DBFS`.

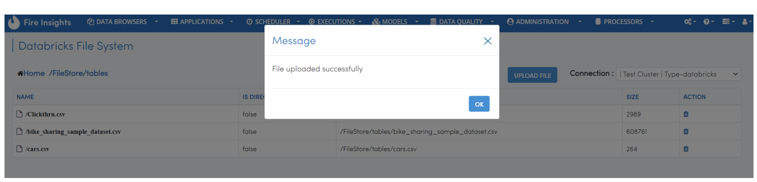

Fire Insights
Databricks Clusters
Cluster Details
SPARK CONTEXT ID :
6133712372202850000
SPARK VERSION :
6.4.x-scala2.11
NODE TYPE ID :
m4.large
DRIVER NODE TYPE ID :
m4.large
AUTOTERMINATION MINUTES :
15
ENABLE ELASTIC DISK:
false
STATE MESSAGE :
JDBC URL :
MIN WORKERS :
MAX WORKERS :
JDBC PORT :
10000
CREATOR USER NAME :
OK

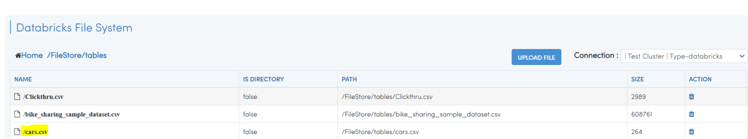

Databricks Clusters
Connection :
| Test Cluster | Type-databricks
CLUSTER ID
CLUSTER NAME
CLUSTER SOURCE
STATE
START TIME
TERMINATED TIME
LAST STATE LOSS TIME
ACTIONS
0826-092903-oak501
Test
UI
RUNNING
2020/08/26 14:59:03 IST
null IST
2020/08/27 17:43:24 IST


Databricks File System
Home
UPLOAD FILE
Connection :
| Test Cluster | Type-databricks
NAME
IS DIRECTORY
PATH
SIZE
ACTION
/FileStore
true
/FileStore
/databricks-results
true
/databricks-results
/modelsavepath
true
/modelsavepath
/steelmodelpath
true
/steelmodelpath
/steelplatemodel
true
/steelplatemodel


Databricks File System
Home
UPLOAD FILE
Connection :
| Test Cluster | Type-databricks
NAME
IS DIRECTORY
PATH
SIZE
ACTION
/FileStore
true
/FileStore
/databricks-results
true
/databricks-results
/modelsavepath
true
/modelsavepath
/steelmodelpath
true
/steelmodelpath
/steelplatemodel
true
/steelplatemodel

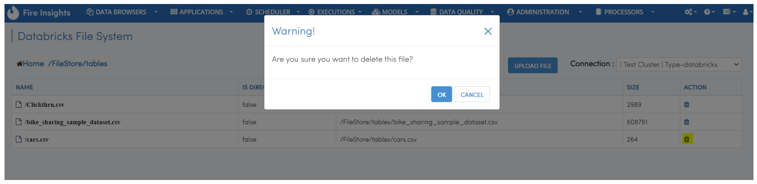

Fire Insights
Databricks File System
Home /FileStore/tables
Upload File
Choose File
cars.csv
UPLOAD
CLOSE
Connection :
| Test Cluster | Type-databricks
/Clickthru.csv
/bike_sharing_sample_dataset.csv
false
/FileStore/tables/bike_sharing_sample_dataset.csv
2989
608761


Fire Insights
Databricks File System
Home /FileStore/tables
Message
File uploaded successfully
OK
UPLOAD FILE
Connection :
| Test Cluster | Type-databricks
/Clickthru.csv
/bike_sharing_sample_dataset.csv
/cars.csv
/FileStore/tables/bike_sharing_sample_dataset.csv
/FileStore/tables/cars.csv
2989
608761
264

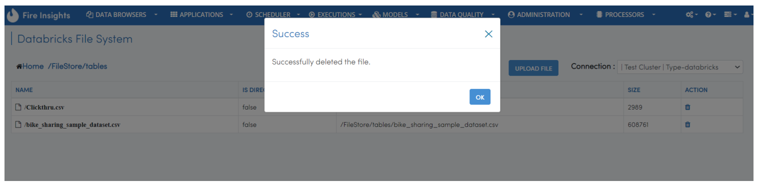

Databricks File System
Home /FileStore/tables
UPLOAD FILE
Connection :
| Test Cluster | Type-databricks
NAME
IS DIRECTORY
PATH
SIZE
ACTION
/Clickthru.csv
false
/FileStore/tables/Clickthru.csv
2989
/bike_sharing_sample_dataset.csv
false
/FileStore/tables/bike_sharing_sample_dataset.csv
608761
/cars.csv
false
/FileStore/tables/cars.csv
264

### Delete file and directory in DBFS

You can delete file and directory in DBFS using delete `ACTION` button.




On successful `deletion`, it will show successful informations and file can be viewed inside the folder in `DBFS`.




## Reading Databricks Tables

Fire Insights enables you to read from and write to Databricks tables.

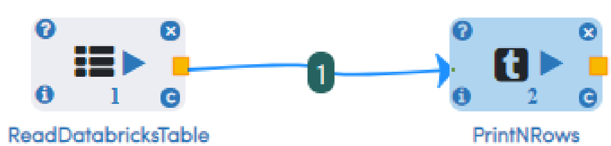
Below is a workflow which reads data from the Databricks table `xyz`. It then processes the data and finally writes out the result to the Databricks table abc.

### Read Databricks table in Workflow

In the workflow use the processor 'ReadDatabricksTable'. It will allow you to read tables from Databricks.

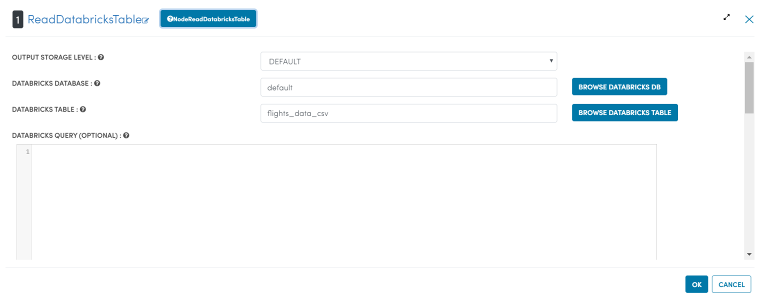
Then use the other processors in Fire for processing the data read from the Databricks Table.

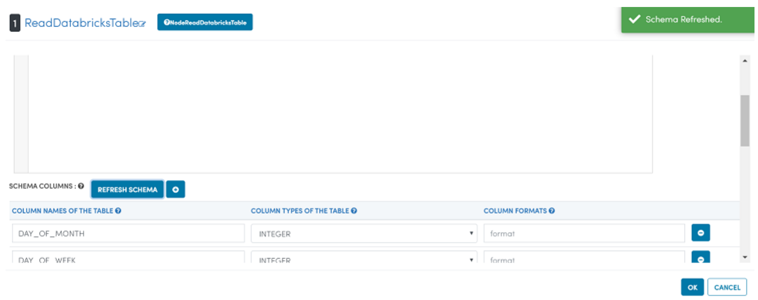
### Workflow



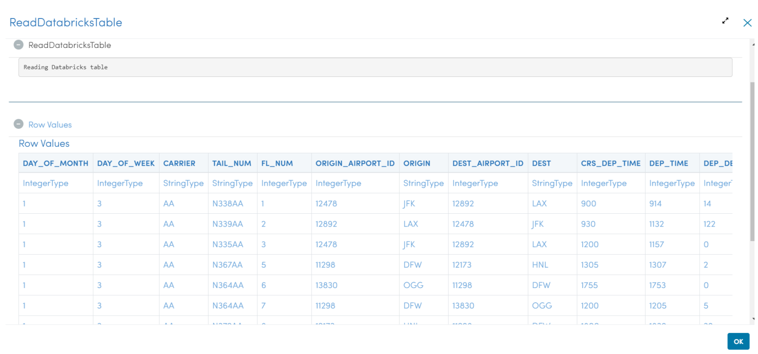

### Processor Configurations for ReadDatabricksTable

### Refresh schema for processor ReadDatabricksTable




### Processor executions for ReadDatabricksTable




### Databricks Workflow execution

Below is the output of executing the above workflow which reads data from a Databricks table.

### Writing to Databricks Tables

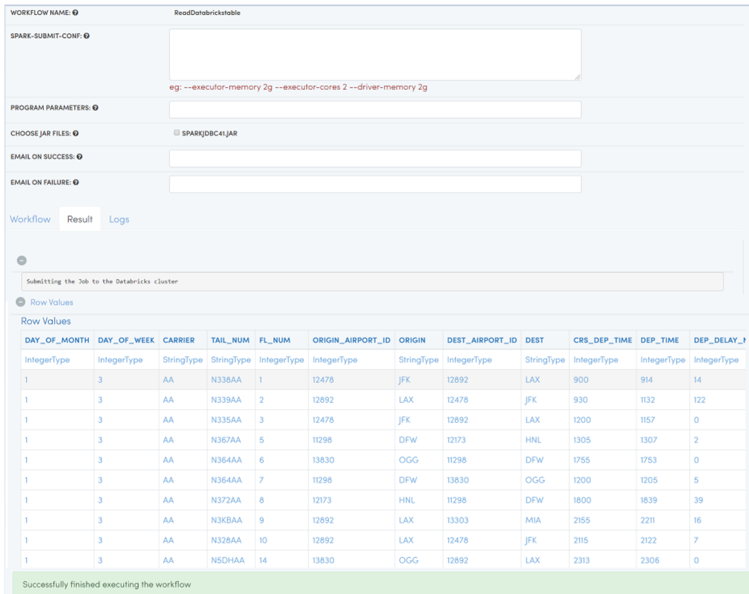
Fire Insights enables you to write to Databricks tables.

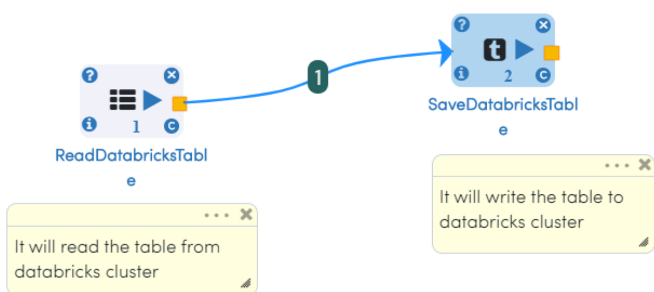
In the workflow use the processor 'SaveDatabricksTable'. It will allow you to save data to tables to Databricks.

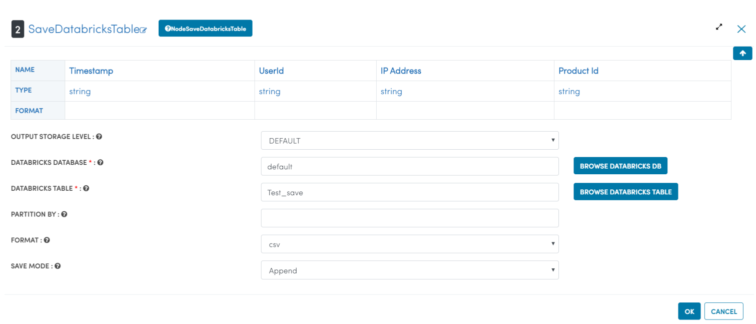
Below is a workflow which writes data to the Databricks table `test_save`.

### Workflow

### Processor Configurations for SaveDatabricksTable

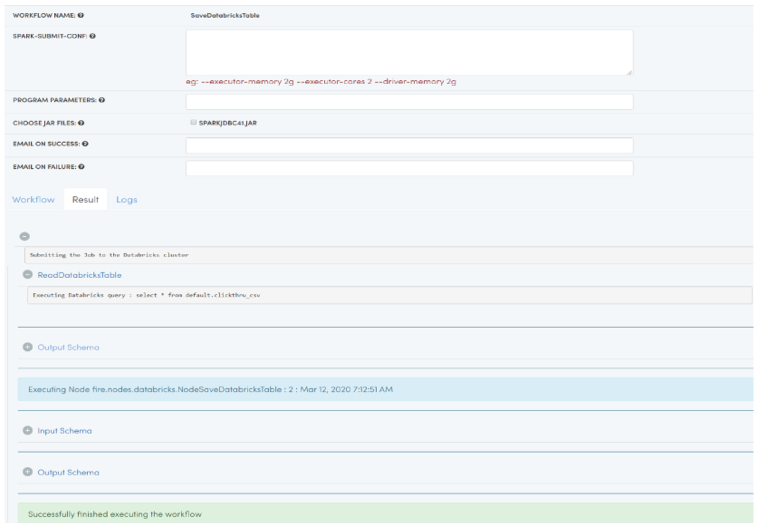

WORKFLOW NAME:
ReadDatabrickstable
SPARK-SUBMIT-CONF:
eg: --executor-memory 2g --executor-cores 2 --driver-memory 2g
PROGRAM PARAMETERS:
CHOOSE JAR FILES:
SPARKJDBC41.JAR
EMAIL ON SUCCESS:
EMAIL ON FAILURE:
Workflow
Result
Logs
Row Values
Successfully finished executing the workflow


ReadDatabricksTabl
e
SaveDatabricksTabl
e
It will read the table from databricks cluster
It will write the table to databricks cluster

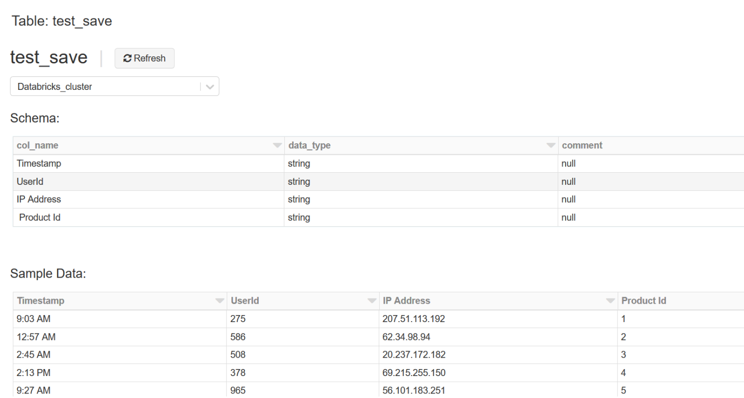

SaveDatabricksTable
Timestamp
UserId
IP Address
Product Id
OUTPUT STORAGE LEVEL :
DEFAULT
DATABRICKS DATABASE * :
default
BROWSE DATABRICKS DB
DATABRICKS TABLE * :
Test_save
BROWSE DATABRICKS TABLE
PARTITION BY :
FORMAT :
csv
SAVE MODE :
Append
OK
CANCEL

### Databricks Workflow execution

Below is the output of executing the above workflow which saves the data to Databricks table.




- Verify the Table




| col_name | data_type | comment |
|---|---|---|
| Timestamp | string | null |
| UserId | string | null |
| IP Address | string | null |
| Product Id | string | null |

| Timestamp | UserId | IP Address | Product Id |
|---|---|---|---|
| 9:03 AM | 275 | 207.51.113.192 | 1 |
| 12:57 AM | 586 | 62.34.98.94 | 2 |
| 2:45 AM | 508 | 20.237.172.182 | 3 |
| 2:13 PM | 378 | 69.215.255.150 | 4 |
| 9:27 AM | 965 | 56.101.183.251 | 5 |

### File Formats

The tables can be saved into CSV, JSON, Parquet and ORC file formats.

If the file format is not specified, the data in tables is stored in Parquet format.

### Reading S3 files

https://docs.databricks.com/_static/notebooks/data-import/s3.html

There are two ways in Databricks to read from S3. You can either read data using an IAM Role or read data using Access Keys.

Databricks recommends leveraging IAM Roles in Databricks.

Fire Insights allows you to browse your Data in S3 and create workflows using them. When the job is submitted to Databricks, the job reads data from the S3 location and processes them.

You can also create external tables in Databricks over data in S3. Fire Insights can process data from Databricks tables.

### Accessing S3 buckets from Databricks

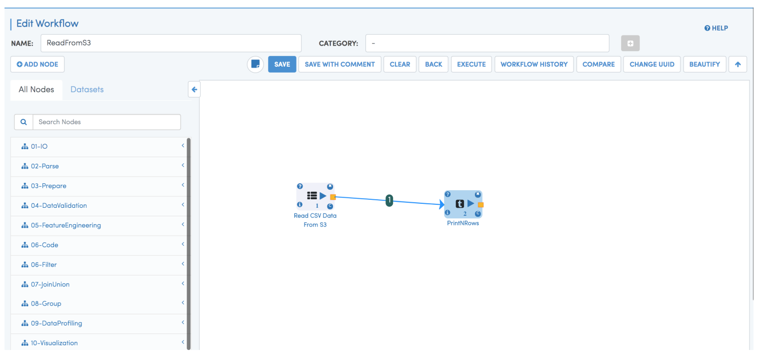
This document from Databricks has very good information on the setup for accessing S3 buckets from Databricks.

https://docs.databricks.com/security/credential-passthrough/iam-passthrough.html

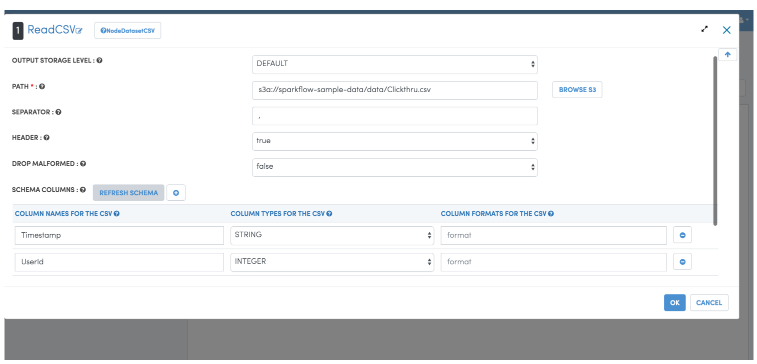
### Read the data from S3 in Workflow

In Sparkflows, user can read the data from S3 location using processors like ReadCSV, ReadParquet, ReadJson etc.

### Workflow




### Browse S3 Path and Refresh schema for processor ReadCSV

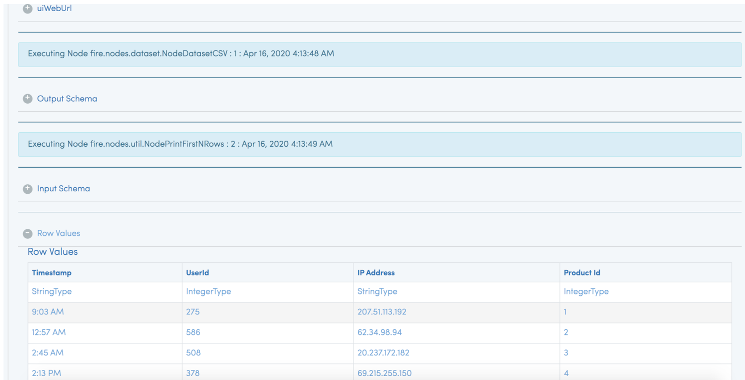



### Workflow executions Results

### Writing to S3 files

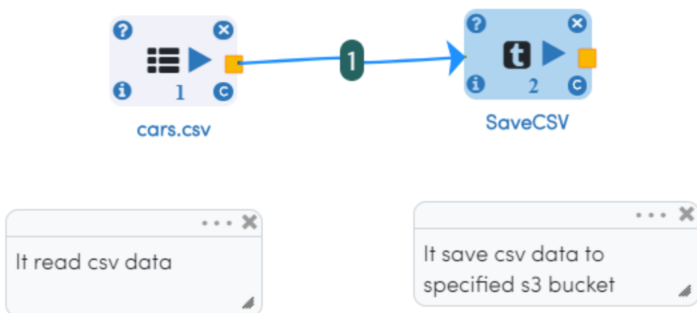
https://docs.databricks.com/_static/notebooks/data-import/s3.html

Fire Insighs workflows can write data to S3 locations.

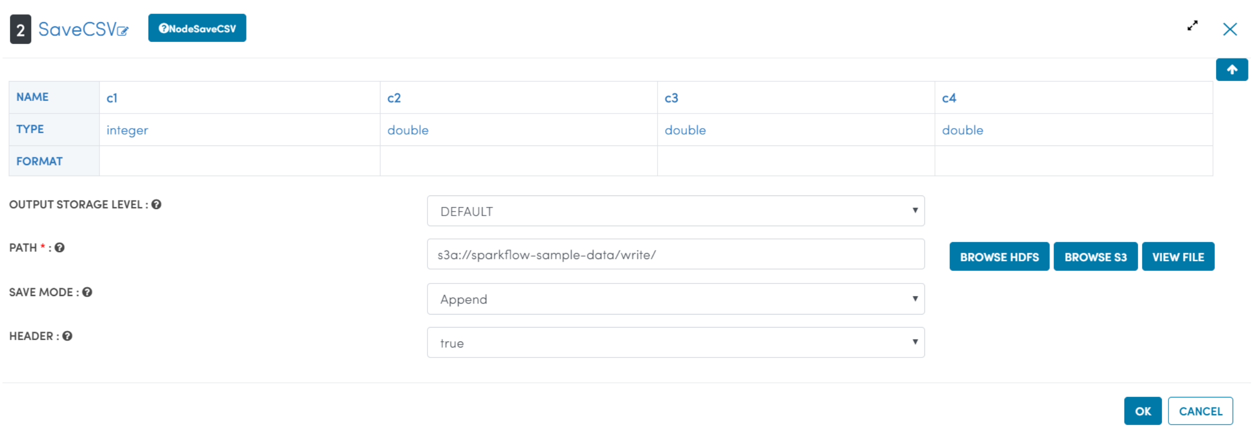



Below is an example workflow which writes data to S3. When the workflow is executed, the Dataframe is saved to the S3 location.

In the dailog box of the save CSV processor the path is specified as `s3a://sparkflow-sample-data/write/`



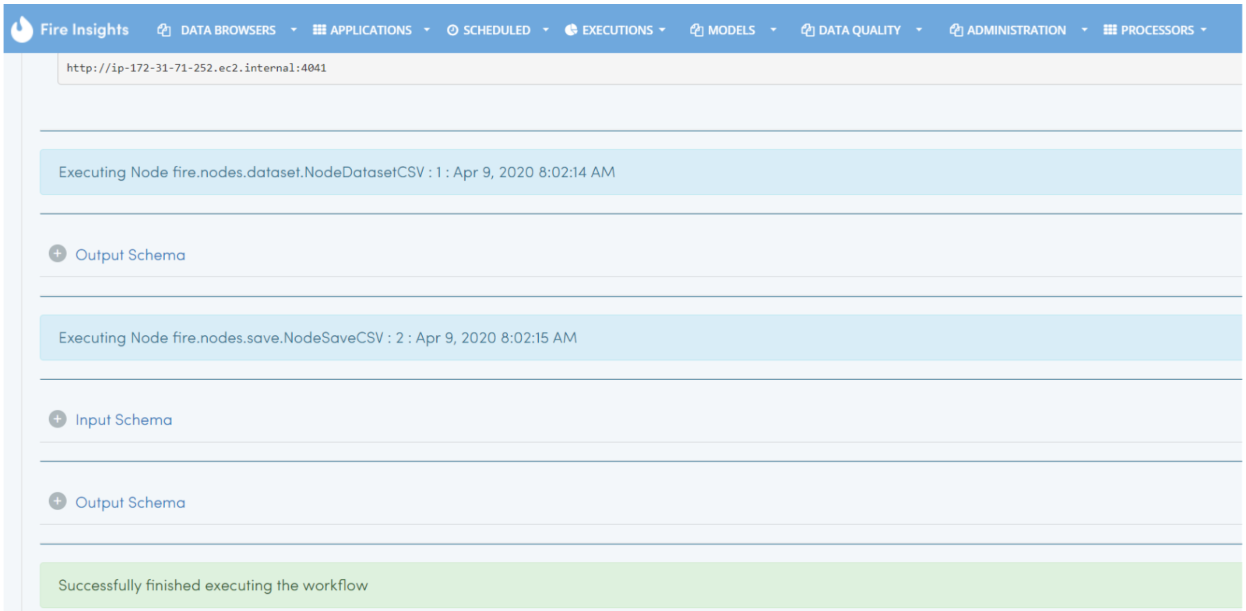

Browse S3 specified Path & other parameter for processor SaveCSV

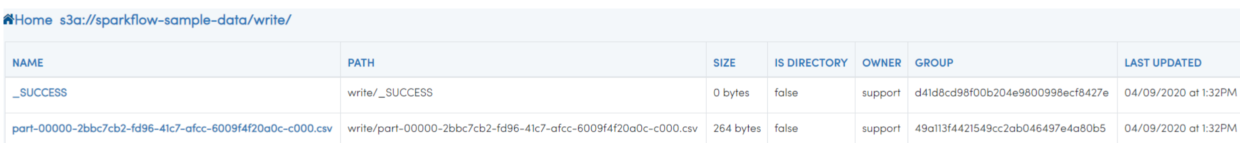



Execution Result

Once the above workflow successfully completed, the save data can be viewed using `DATABROWSERS/AWS S3` Location with specified path

### 16.1.5 Troubleshooting Fire/Databricks Integration



Home s3a://sparkflow-sample-data/write/

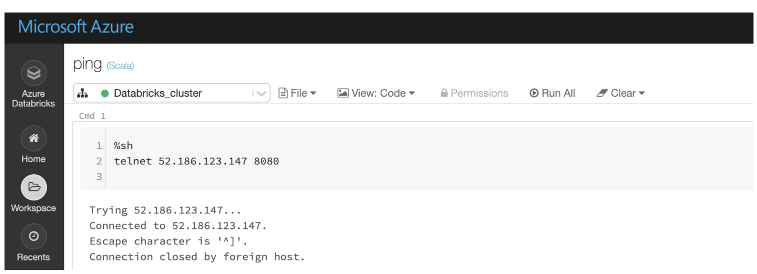
| NAME | PATH | SIZE | IS DIRECTORY | OWNER | GROUP | LAST UPDATED |
|---|---|---|---|---|---|---|
| _SUCCESS | write/_SUCCESS | 0 bytes | false | support | d41d8cd98f00b204e9800998ecf8427e | 04/09/2020 at 1:32PM |
| part-00000-2bbc7cb2-fd96-41c7-afcc-6009f4f20a0c-c000.csv | write/part-00000-2bbc7cb2-fd96-41c7-afcc-6009f4f20a0c-c000.csv | 264 bytes | false | support | 49a113f4421549cc2ab046497e4a80b5 | 04/09/2020 at 1:32PM |

### When the workflow is executed, nothing shows up in Fire

One problem might be that the `postbackURL` is not configured right in Fire Insights under Administration/Configuration.

The other problem can be that the machine running Fire Insights is not accessible from the Databricks Cluster. Test connectivity to the Fire Insights machine from Databricks.

Connectiving from Databricks to Fire postbackURL can be done in Databricks via Notebooks using the telnet command.




### When the workflow is executed, nothing shows up in Fire

Another reason might be that you are using the Databricks `High Concurrency` cluster. Ensure that you are connecting Fire to Databricks `Standard` cluster.

### When accessing most Databricks pages in Fire, it gives Simba JDBC error

The reason for it is that the Databricks Simba JDBC jar file is not deployed in Fire.

https://docs.sparkflows.io/en/latest/databricks/databricks-installation.html#install-databricks-jdbc-driver

### In the workflow editor, it shows 'Cannot connect to Fire'

Ensure that under `Administration/Configuration`, app.runOnCluster is set to `false`.

### Checking the cluster logs in Databricks

There are times when it is helpful to look at the Cluster logs in Databricks when running Fire with Databricks.

The following logs under `Driver Logs` are useful:

- log4j-active.log

Search for `WorkflowExecuteDatabricks` in the logs to view if the Fire Insights Job is running in Databricks.

**java.lang.Exception: An error occurred while initializing the REPL. Please check whether there are conflicting Scala libraries o**
at com.databricks.backend.daemon.driver.DatabricksILoop$class.initSpark(DatabricksILoop.scala:98)

This error can happen when running spark 2.3 version of Fire with spark 2.4 cluster on Databricks. Either upgrade Fire to spark 2.4 version, or create another Databricks cluster which supports spark 2.3.

### Databricks Cluster Versions Support

Databricks Runtime Version Spark Version Scala Version

6.2 2.4.4 2.11

6.3 2.4.4 2.11

6.4 2.4.5 2.11

6.5 2.4.5 2.11

CHAPTER 17

# AWS Integration

## 17.1 AWS Guide

### 17.1.1 Introduction

Fire Insights is the flagship product from Sparkflows. It is seamlessly integrated with AWS. With Fire Insights you can perform self-serve data processing, analytics and machine learning on AWS.

Fire Insights integrates with EMR, S3, Redshift, SageMaker, HIVE and Kinesis.

Fire Insights comes with a number of components including:

- **Workflow Editor** : To create workflows for data processing, analytics and machine learning.
- **260+ Processors** : These include reading data from various stores, data processing, machine learning and visualizations.
- **Execution Engine** : For executing the workflow on EMR
- **Scheduler** : For scheduling running the workflows at certain time intervals

Sparkflows Fire Insights can be deployed to an existing Amazon EMR cluster, or you can use one of our CloudFormation templates to set up a new Amazon EMR Cluster. If you use our provided CloudFormation templates we'll create an EMR cluster for you or even an EMR cluster and MySQL instance running in RDS, depending on which template you choose.

#### Pre-requisites and Requirements

Fire Insights needs EMR for running the workflows. So, you need a running EMR cluster for using Fire Insights.

You also need ssh access to one of the machines of the EMR cluster for installing Fire Insights. This machine is typically an edge node or a master node of the EMR cluster.

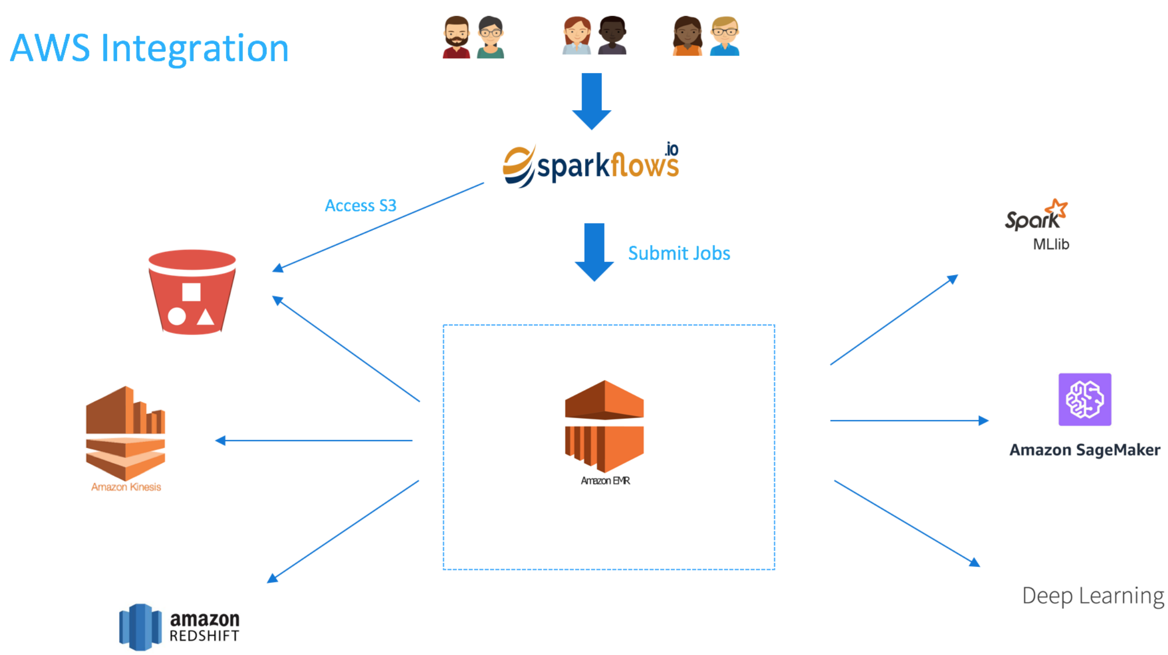
- Getting started with EMR - https://aws.amazon.com/emr/getting-started/

- Opening SSH access to the EMR master node - https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-connect-master-node-ssh.html

### Architecture

Fire Insights runs on the edge node or one of the master nodes of the EMR cluster. It submits the processing jobs onto the cluster. By default it runs on port 8080. This port needs to be changed to some port which is available on the machine as it is in use by default. Lets assume we will use port 8085.

When the jobs are fired onto the EMR cluster, it can read/write data from S3/HDFS/Redshift/Kinesis. It can also fire Machine Learning modeling jobs to SageMaker.




## 17.1.2 Planning Guide

This document describes details to help you plan on deploying and using Fire Insights on AWS.

### Security

Fire Insights is installed onto the edge node or master node of the EMR cluster. The jobs fired by the users would be able to access and process data on S3, HDFS, Redshift, Kinesis.

### Costs

The main costs involved when using Fire Insights are around the EMR cluster. EMR cluster has master nodes and workflow nodes.

Pricing for EMR can be found here : https://aws.amazon.com/emr/pricing/

The more processing capacity needed, the larger should be the size of the EMR cluster.

Fire Insights can also run Machine Learning Modeling jobs onto SageMaker. If this is used, there would be cost associated with using AWS SageMaker. Amazon SageMaker Pricing details are here : https://aws.amazon.com/sagemaker/pricing/

#### Sizing

EMR cluster normally starts with a mimumum of 1 master node and 2 worker nodes.

We recommend using at least 16GB machines for the master and worker nodes.

As your data volume and the number of concurrent users increases, we recommend increasing the size of the EMR cluster. Memory for the worker nodes can be increased to 32GB to 64GB to 512GB. Since Apache Spark has the ability to use as much memory you provide, its a good idea to give it more memory.

Same goes for the number of disks and vcores.

### 17.1.3 Deployment Guide

Fire can be easily installed on an AWS EMR Cluster. Fire can be installed on the master node of an EMR cluster. It would then submit the jobs to the EMR cluster.

Below are the overall steps for installing Fire Insights on EMR.

- *ssh into the Master node*
- *Download Fire Insights* from https://www.sparkflows.io/download
- *Unzip it*
- *Create H2 Database*
- *Start Fire*

#### Steps

- Start your EMR cluster on AWS:

```
Start your EMR cluster on AWS if you do not already have it running.
```

- Update the inbound rules for the Master Node:

```
- We would have Fire listening on ports 8085 and 8086
- Fire by default listens on 8080 and 8443. But EMR clusters have other processes␣
↪listening on these ports.
- So we will later change it to listen on ports 8085 and 8086
- Update the inbound rules for the Master Node to allow ports 8085 and 8086
```

- ssh into the Master EMR node as the `hadoop` user:

```
ssh -i my.pem hadoop@ec2-xx-yyy-zz-aaa.compute-1.amazonaws.com
```

- Download the fire tgz file by one of the following options:
  - **https://www.sparkflows.io/download** OR
  - **https://www.sparkflows.io/archives** OR
  - wget https://s3.amazonaws.com/sparkflows-release/fire/rel-x.y.z/2/fire-x.y.z.tgz

- Unpack it:

```
tar xvf fire-x.y.z.tgz
```

- Copy hadoop-lzo.jar:

```
cp /usr/lib/hadoop-lzo/lib/hadoop-lzo.jar /home/hadoop/fire-3.1.0/fire-user-lib
```

- Configure Fire to listen on ports 8085 and 8086:

```
- cd <fire install_dir>
- Edit conf/application.properties
- Update the last two lines to below:
    http.port=8085
    https.port=8086
```

- Create H2 DB:

```
Fire stores its metadata into the embedded H2 database. You can also connect it␣
↪to an external MySQL database.

  cd <fire install_dir>
  ./create-h2-db.sh
```

- Launch Fire Server:

```
cd <fire install_dir>
./run-fire-server.sh start
```

- Open your web browser and navigate to:

```
<machine_name>:8085/index.html
```

- Login with the following default username and password:

```
username : admin
password : admin
```

- Connect Fire with the EMR Cluster:

```
- Go to Administration/Configuration
- Click on 'Infer Hadoop Configs'
- Save

- If your EMR cluster is not running HIVE, update 'spark.sql-context = SQLContext'
```

- Create the `hadoop` user in Fire:

```
- Under Administration/Users, add the 'hadoop' user
```

## Loading Example Workflows

- From the home page of Fire Insights, click on ***Load Example Applications***
- Upload the Fire examples data onto HDFS:

```
cd <fire install_dir>
hadoop fs -put data /tmp
```

### Install and Running Example Workflows

- Start off with executing the example workflows:

```
- Fire comes pre-packaged with a number of example workflows
- You can install them by clicking on the 'Install example workflows' link in the␣
↪landing page when logged in as the `admin` user.
```

- Logout from the current session and login again with the 'hadoop' user
  - Execute the workflows

### Adding a new user

Create the home directory on HDFS for the new user.

For example, for user 'test':

- hadoop fs -mkdir /user/test
- hadoop fs -chown test:test /user/test

Create the user in Fire Insights if not already created.

### Extra configuration for running PySpark

EMR needs extra configurations when running PySpark. In the below the python 3.6 virtual environment is installed in the directory /home/hadoop/venv

- export SPARK_HOME=/usr/lib/spark/
- export PYSPARK_PYTHON=/home/hadoop/venv/bin/python
- export YARN_CONF_DIR=/etc/hadoop/conf

## 17.1.4 S3 Integration

Fire Insights allows you to access your files on S3. This page describes S3 integration of Fire.

We recommend controlling access to S3 using IAM Roles.

- Run Fire Insights on an EC2 machine with the appropriate S3 IAM Role.
- Run the EMR cluster with the appropriate S3 IAM Role.

If you are running Fire Insights on a independent machine, you can also use `aws configure` to set the AWS Access Key and Secret Access Key on the machine.

AWS CLI S3 Reference : https://docs.aws.amazon.com/cli/latest/reference/s3/ls.html

### Installing aws cli

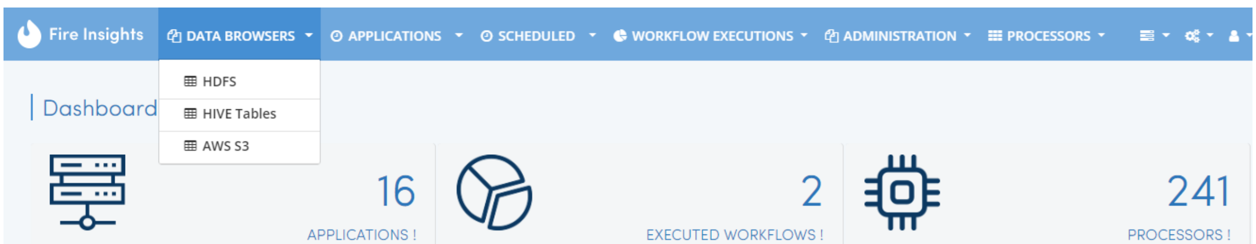
- http://docs.aws.amazon.com/cli/latest/userguide/installing.html
- pip install awscli –upgrade –user

### Configuring AWS access key and password

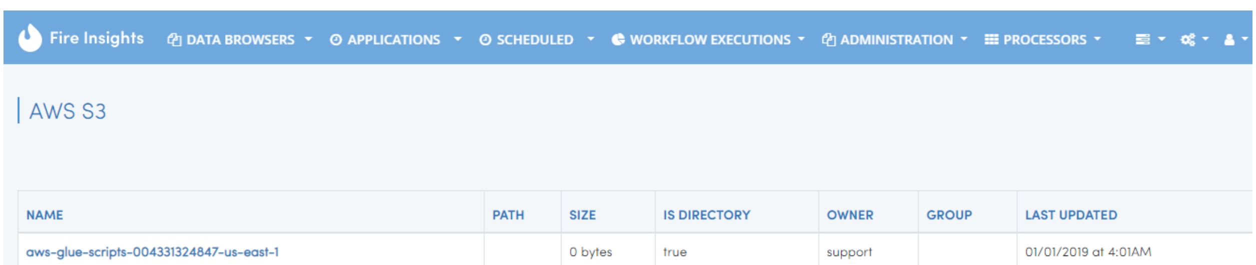
Run `aws configure` to configure your credentials on the machine on which Fire Insights is running.

### Access S3 in fire-ui

In Fire Insights, you can browse S3 under the menu Browser/AWS S3.




- Click on AWS S3 to view the files on S3.




### Protecting Data Using Server Side Encryption

Data encryption settings on S3 buckets: https://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html

### REFERENCE : Creating Access Key & Secret Key

1. You'll need create a user with programmatic access by following the steps here (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html).
2. Next, you'll create an IAM policy that defines what this user has access to in your AWS account. It's important to only grant this user minimal access within your account. See this documentation for how to create IAM policies (https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html).
3. Finally, you'll create an access key and secret key for this user (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_CreateAccessKey).

*Note* It's important to regularly rotate your access and secret keys. See this documentation for more information (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#rotating_access_keys_console)

### 17.1.5 Testing Fire Insights on AWS

After you have deployed Fire Insights on AWS, it is a good idea to test the things.

Below are a few good steps for achieving it:

- Ensure you can log into the sytem
- View the Sample Applications
- Execute a workflow on EMR

#### Log into the System

- Log into the system as the `hadoop` user which you had created during the Deployment process. * http://docs.sparkflows.io/en/latest/aws/running-on-emr.html

#### View the Sample Applications

- Go to the ‘‘Applications/List’ page.
- If you loaded the Sample Applications during the deployment process you would see a number of Sample Applications listed.
- Click on any of them to view their Datasets/Workflows etc.

#### Execute a workflow on EMR

- From the Applications/Workflows page.
- Click on the Execute icon next to any workflow
- This will open up the Execute page.
- Click on `Execute` to execute the workflow on the EMR cluster
- The results of execution would get displayed on the page.

### 17.1.6 Operational Guide

This document describes details for operating Sparkflows when running on AWS.

#### Onboarding New Users

New users can be created in Sparklows by logging into it. Then go to Administration/Users.

#### Health Check

The main server process which handles the web requests is fire-ui. This is a long running process and very stable. This process can be checked for responsiveness for any health checks.

### Backup and Recovery

Fire Insights stores the metadata into a Relational Database.

It comes with an embedded H2 database. It scales well for pretty heavy loads and upto 50 users. Sparkflows can be easily configured to run with an MySQL database.

When running with H2 database, Sparkflows by default stores the db files in the user home directory which is running Sparkflows. There are 2 files:

- firedb.mv.db
- firedb.trace.db

For backup, just copying these files to a backup location is enough. There is no need to stop Sparkflows. It is a good idea to copy it to another maching.

When running with MySQL running on the same or different machine, the MySQL database named `fire` needs to be backed up.

### Routing Maintenance

Apart from backups of the database, Fire does not need much of routine maintenance.

Fire stores the details of the job executions in the relational database. Over time, you may have too many jobs executed. Deleting old jobs from the Workflow Executions page is a good idea so as not to fill up the database too much. But it has the ability to handle millions of jobs, so you do not have to worry too much about it.

### Support

For support, you can contact Sparkflows at support@sparkflows.io. We will guide you through the process.

Sparkflows can also support you though Zendesk tickets. Get in touch with us for guidance and setup.

## 17.1.7 Copying files to S3 with aws-cli

There would be times when you want to upload multiple files from your laptop to S3. This document describes the process for it.

### Installing aws-cli on mac

brew install awscli

### Configure AWS Credentials

aws configure:

```
- Enter your awsAccessKeyId
- Entery your awsSecretAccessKey
```

### View S3 Buckets

- aws s3 ls

#### View S3 Directory

- aws s3 ls s3://bucket_name/dir1/

#### Copy files to S3

Copy all files from local_direcory to s3://bucket-name/dir1:

```
aws s3 cp local_directory s3://bucket-name/dir1 --recursive
```

#### Delete All Files in Directory

- aws s3 rm s3://bucket_name/dir1/ –recursive

#### Setting Roles and Policies for EMR

In order to be able to access S3 files from the EMR cluster, attach the AmazonS3FullAccess Policy to the EMRDefaultRole.

Now the EMR cluster would have access to the S3 buckets.

#### REFERENCE : Creating Access Key & Secret Key

1. You'll need create a user with programmatic access by following the steps here (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html).
2. Next, you'll create an IAM policy that defines what this user has access to in your AWS account. It's important to only grant this user minimal access within your account. See this documentation for how to create IAM policies (https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html).
3. Finally, you'll create an access key and secret key for this user (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_CreateAccessKey).

*Note* It's important to regularly rotate your access and secret keys. See this documentation for more information (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#rotating_access_keys_console)

### 17.1.8 Reading/Writing from S3

Fire is fully integrated with AWS S3. The Dataset Processors of Fire, can directly read data from S3 if the policies allow them to.

#### Dataset Processors

Dataset Processors include:

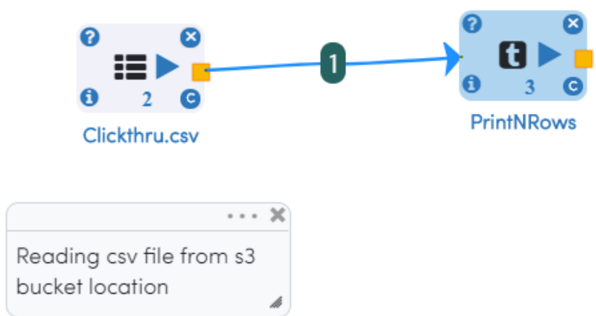
- Read CSV
- Read Parquet
- Read JSON
- Read XML

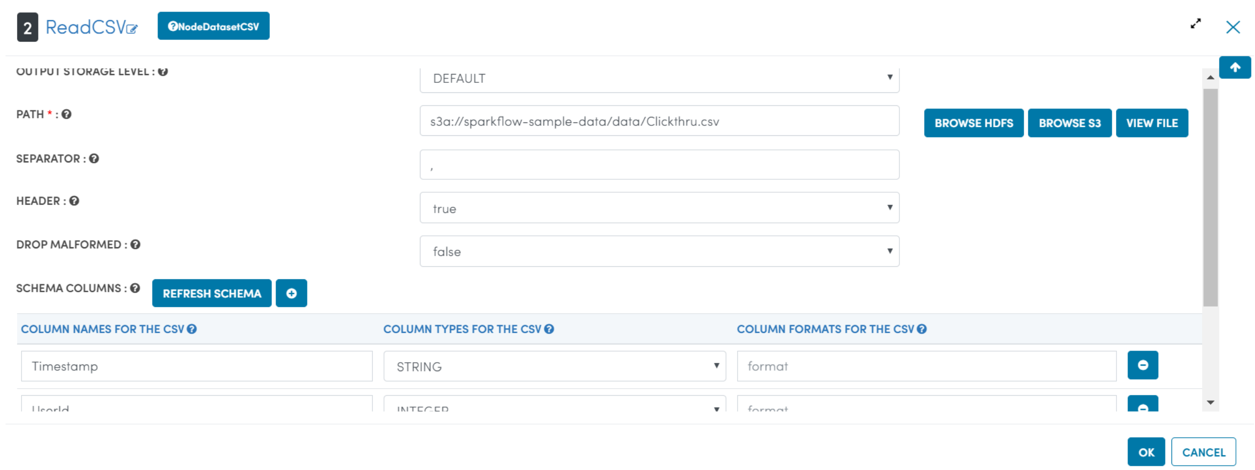
The path specified for reading from S3 would be s3://. . .

### Reading from S3

Below is an example Workflow. It reads a CSV file from S3, parses it and prints out the first 10 records.

In the dialog box of the Read CSV processor the path is specified as `s3a://sparkflow-sample-data/data/Clickthru.csv`




### Writing to S3

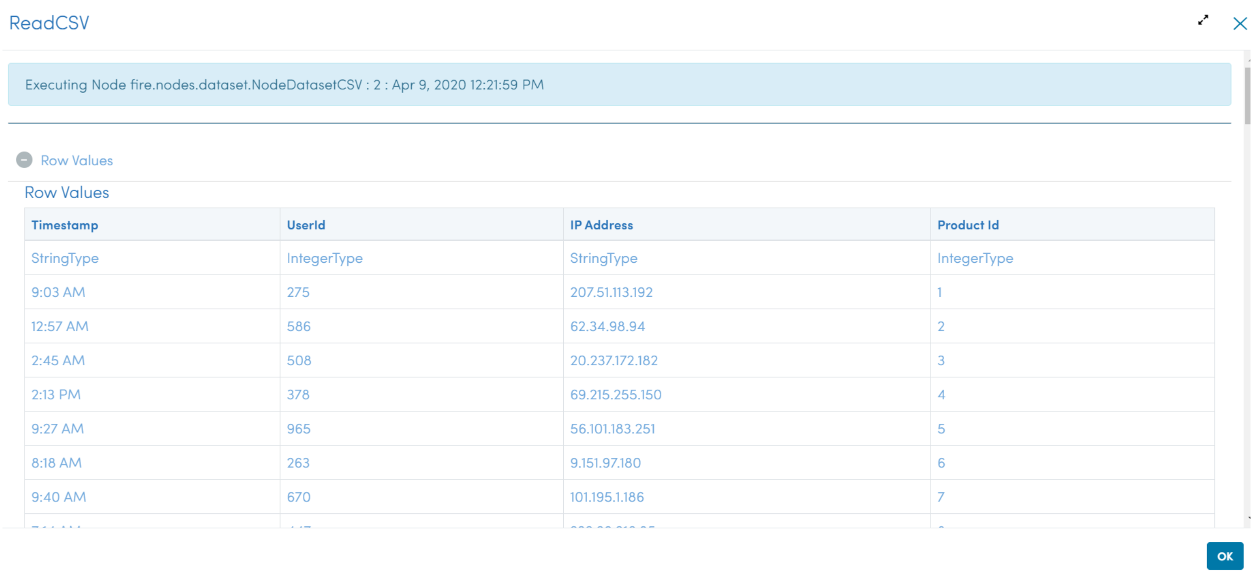
Below is an example Workflow. It reads a CSV file and save it to S3 path specified.

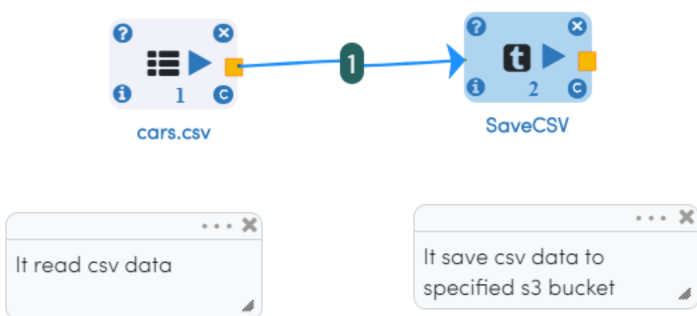
In the dailog box of the save CSV processor the path is specified as `s3a://sparkflow-sample-data/write/`

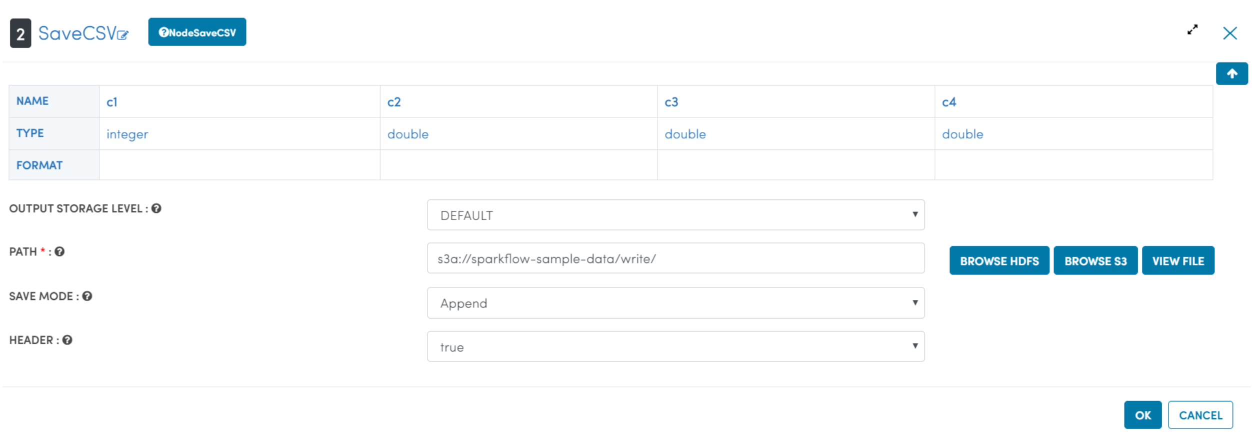
Execution Result

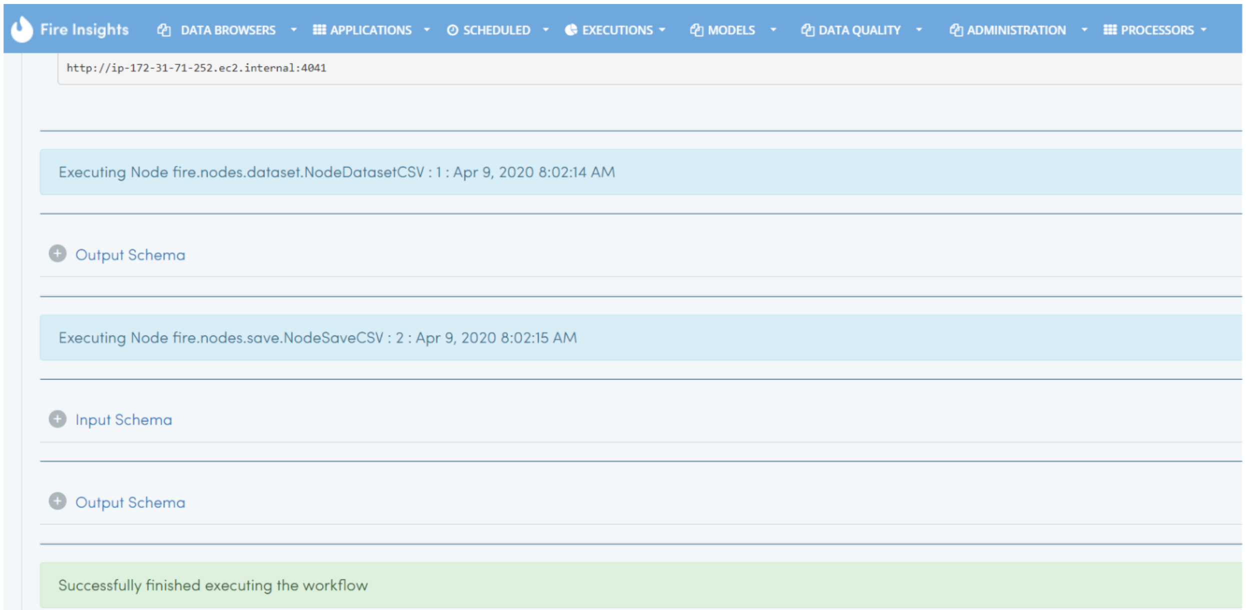
Once the above workflow successfully completed, the save data can be viewed using `DATABROWSERS/AWS S3` Location with specified path

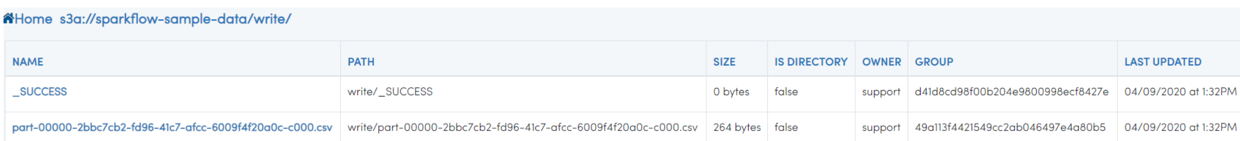
## 17.1.9 Saving ML Model to S3


ReadCSV
Executing Node fire.nodes.dataset.NodeDatasetCSV : 2 : Apr 9, 2020 12:21:59 PM
Row Values
Row Values
Timestamp | UserId | IP Address | Product Id
StringType | IntegerType | StringType | IntegerType
9:03 AM | 275 | 207.51.113.192 | 1
12:57 AM | 586 | 62.34.98.94 | 2
2:45 AM | 508 | 20.237.172.182 | 3
2:13 PM | 378 | 69.215.255.150 | 4
9:27 AM | 965 | 56.101.183.251 | 5
8:18 AM | 263 | 9.151.97.180 | 6
9:40 AM | 670 | 101.195.1.186 | 7
OK


1
1
2
cars.csv
SaveCSV
It read csv data
It save csv data to specified s3 bucket


2 SaveCSV
NodeSaveCSV
NAME | c1 | c2 | c3 | c4
TYPE | integer | double | double | double
FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
PATH * :
s3a://sparkflow-sample-data/write/
BROWSE HDFS
BROWSE S3
VIEW FILE
SAVE MODE :
Append
HEADER :
true
OK
CANCEL





Home s3a://sparkflow-sample-data/write/

| NAME | PATH | SIZE | IS DIRECTORY | OWNER | GROUP | LAST UPDATED |
|---|---|---|---|---|---|---|
| _SUCCESS | write/_SUCCESS | 0 bytes | false | support | d41d8cd98f00b204e9800998ecf8427e | 04/09/2020 at 1:32PM |
| part-00000-2bbc7cb2-fd96-41c7-afcc-6009f4f20a0c-c000.csv | write/part-00000-2bbc7cb2-fd96-41c7-afcc-6009f4f20a0c-c000.csv | 264 bytes | false | support | 49a113f4421549cc2ab046497e4a80b5 | 04/09/2020 at 1:32PM |

### Saving Spark ML Model

Below is an example workflow in sparkflows, where data is read from S3 and the final Spark ML model is saved to S3 location.

Workflow:

Configure ReadCSV

Configure SaveMlModel

Execution Result:

### Saving H20 ML Model

Below is an example workflow in sparkflows, where final H20 ML model is saved to S3 location.

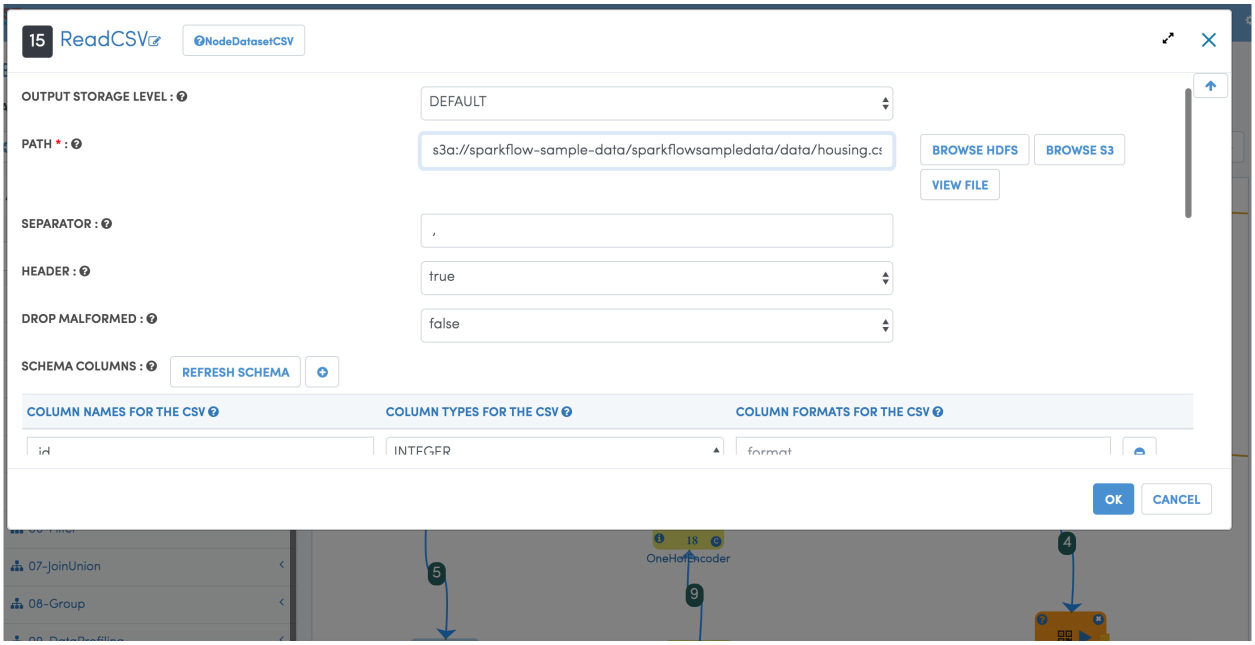
Workflow:

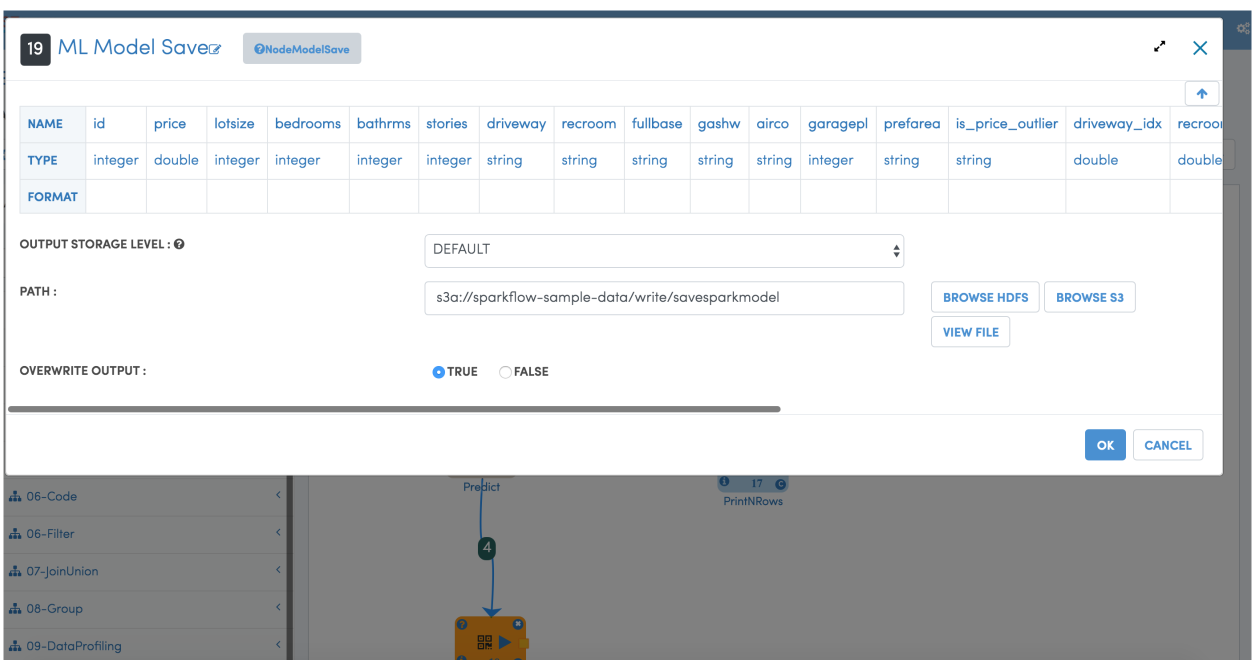
Configure Save H20 ML Model

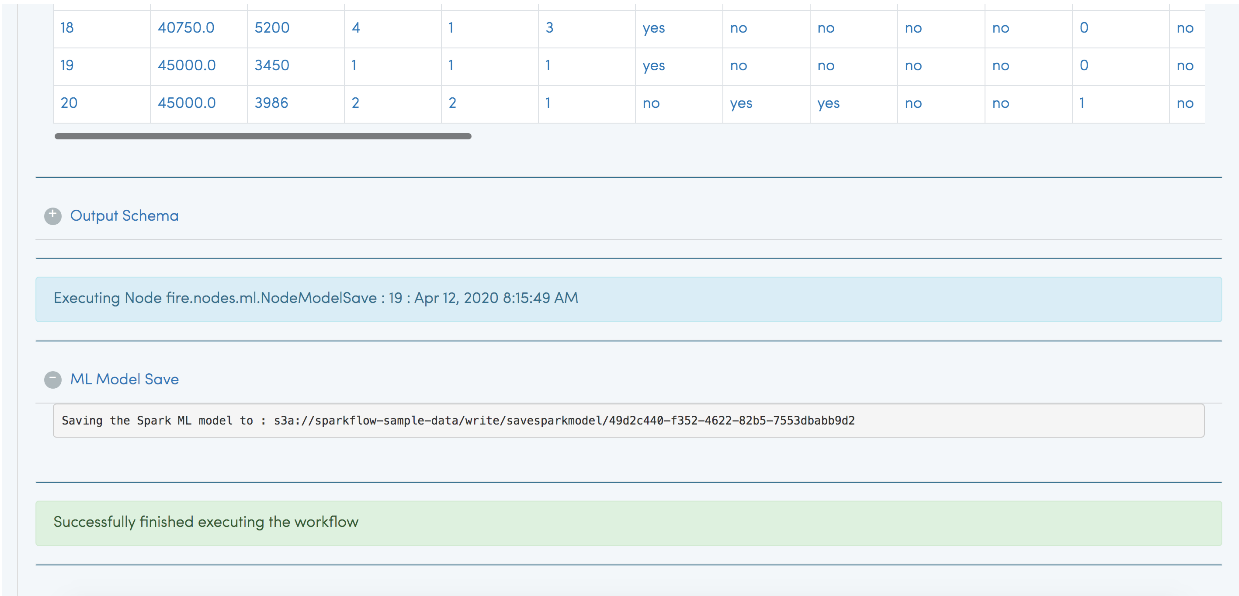
Execution Result:

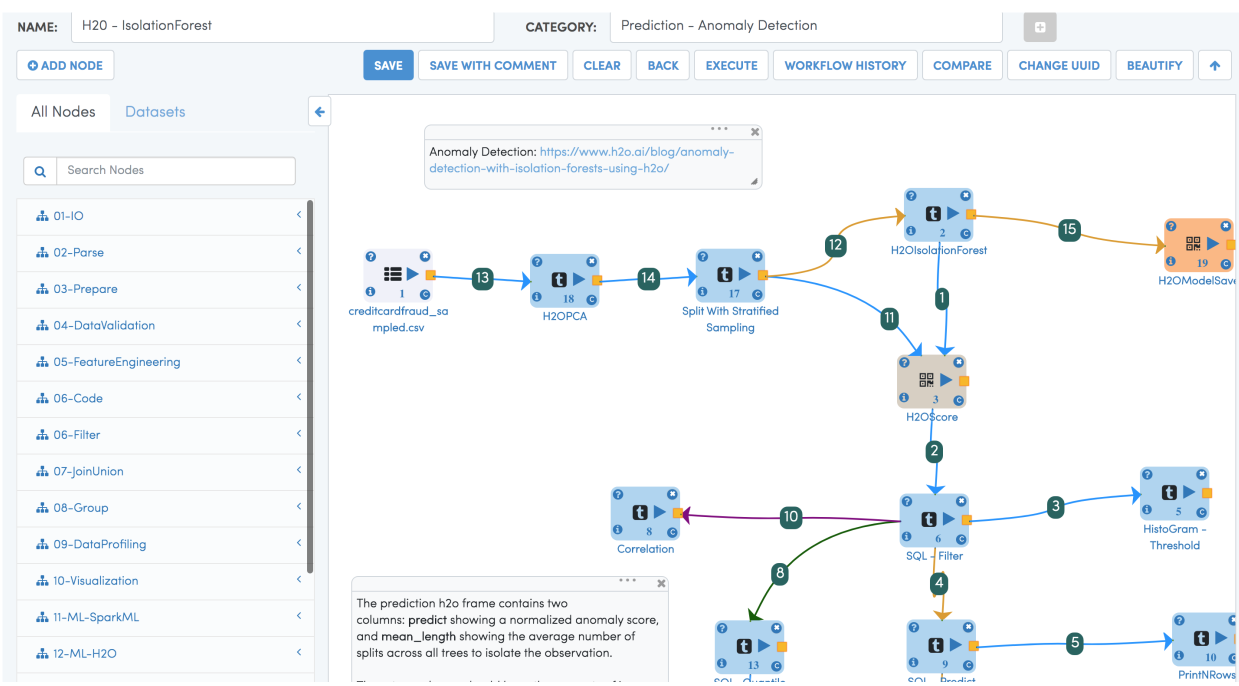
## 17.1.10 Fire Integration with HIVE

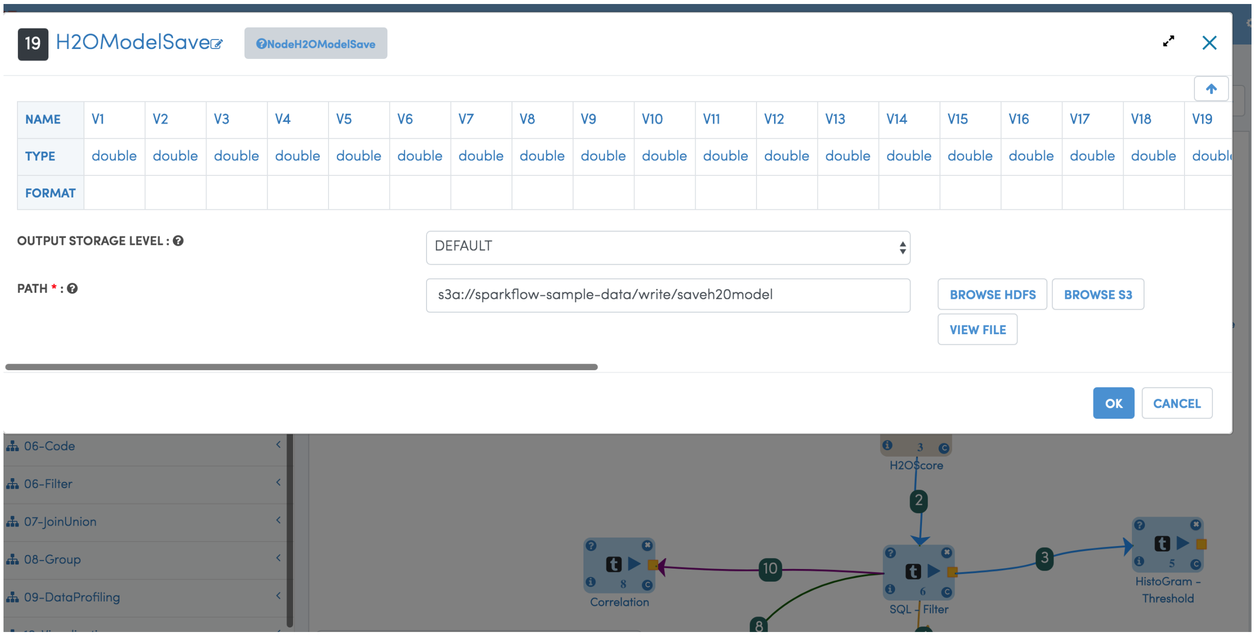
Fire seamlessly integrates with HIVE when running on AWS.

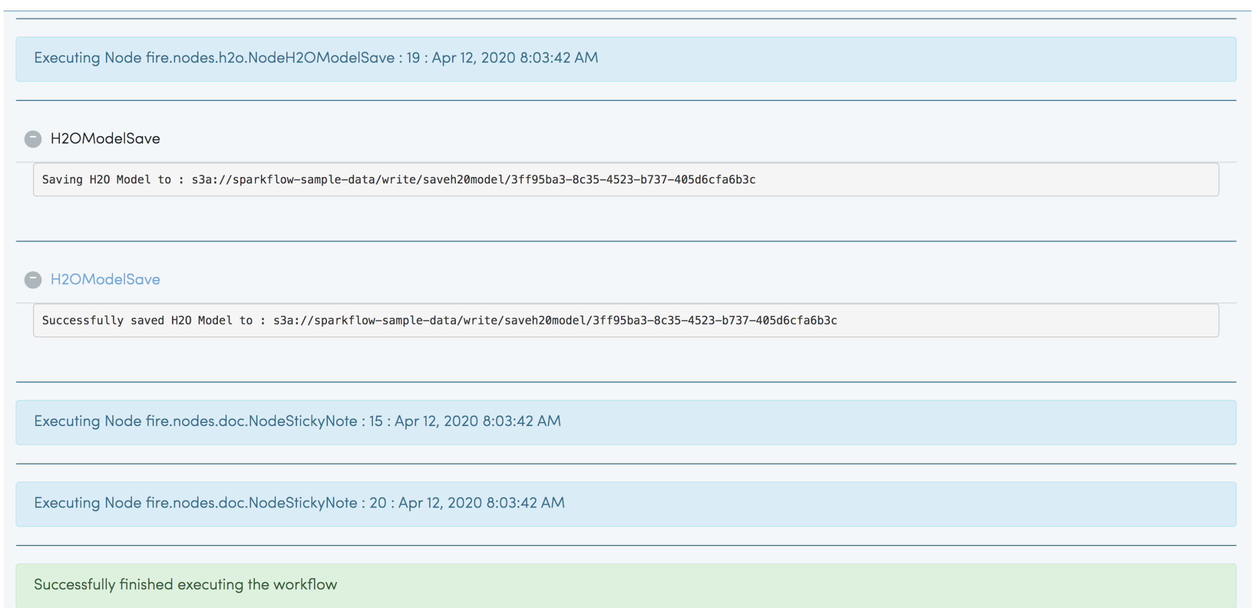

15
ReadCSV
NodeDatasetCSV
OUTPUT STORAGE LEVEL :
DEFAULT
PATH * :
s3a://sparkflow-sample-data/sparkflowsampledata/data/housing.cs
BROWSE HDFS
BROWSE S3
VIEW FILE
SEPARATOR :
,
HEADER :
true
DROP MALFORMED :
false
SCHEMA COLUMNS :
REFRESH SCHEMA
COLUMN NAMES FOR THE CSV
COLUMN TYPES FOR THE CSV
COLUMN FORMATS FOR THE CSV
id
INTEGER
format
OK
CANCEL
07-JoinUnion
08-Group
OneHotEncoder


19
ML Model Save
NodeModelSave
NAME id price lotsize bedrooms bathrms stories driveway recroom fullbase gashw airco garagepl prefarea is_price_outlier driveway_idx recroo
TYPE integer double integer integer integer integer string string string string string integer string string double double
FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
PATH :
s3a://sparkflow-sample-data/write/savesparkmodel
BROWSE HDFS
BROWSE S3
VIEW FILE
OVERWRITE OUTPUT :
TRUE
FALSE
OK
CANCEL
06-Code
06-Filter
07-JoinUnion
08-Group
09-DataProfiling
Predict
PrintNRows


18 40750.0 5200 4 1 3 yes no no no no 0 no
19 45000.0 3450 1 1 1 yes no no no no 0 no
20 45000.0 3986 2 2 1 no yes yes no no 1 no
Output Schema
Executing Node fire.nodes.ml.NodeModelSave : 19 : Apr 12, 2020 8:15:49 AM
ML Model Save
Saving the Spark ML model to : s3a://sparkflow-sample-data/write/savesparkmodel/49d2c440-f352-4622-82b5-7553dbabb9d2
Successfully finished executing the workflow


NAME: H20 - IsolationForest
CATEGORY: Prediction - Anomaly Detection
ADD NODE
SAVE
SAVE WITH COMMENT
CLEAR
BACK
EXECUTE
WORKFLOW HISTORY
COMPARE
CHANGE UUID
BEAUTIFY
All Nodes
Datasets
Search Nodes
01-IO
02-Parse
03-Prepare
04-DataValidation
05-FeatureEngineering
06-Code
06-Filter
07-JoinUnion
08-Group
09-DataProfiling
10-Visualization
11-ML-SparkML
12-ML-H2O
Anomaly Detection: https://www.h2o.ai/blog/anomaly-detection-with-isolation-forests-using-h2o/
creditcardfraud_sampled.csv
H2OPCA
Split With Stratified Sampling
H2OIsolationForest
H2OModelSave
H2OScore
Correlation
SQL - Filter
HistoGram - Threshold
PrintNRows
The prediction h2o frame contains two columns: predict showing a normalized anomaly score, and mean_length showing the average number of splits across all trees to isolate the observation.


19 H2OModelSave
NodeH2OModelSave
NAME V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19
TYPE double double double double double double double double double double double double double double double double double double
FORMAT
OUTPUT STORAGE LEVEL :
DEFAULT
PATH * :
s3a://sparkflow-sample-data/write/saveh20model
BROWSE HDFS
BROWSE S3
VIEW FILE
OK
CANCEL
06-Code
06-Filter
07-JoinUnion
08-Group
09-DataProfiling
H2OScore
Correlation
SQL - Filter
HistoGram - Threshold


Executing Node fire.nodes.h2o.NodeH2OModelSave : 19 : Apr 12, 2020 8:03:42 AM
H2OModelSave
Saving H2O Model to : s3a://sparkflow-sample-data/write/saveh20model/3ff95ba3-8c35-4523-b737-405d6cfa6b3c
H2OModelSave
Successfully saved H2O Model to : s3a://sparkflow-sample-data/write/saveh20model/3ff95ba3-8c35-4523-b737-405d6cfa6b3c
Executing Node fire.nodes.doc.NodeStickyNote : 15 : Apr 12, 2020 8:03:42 AM
Executing Node fire.nodes.doc.NodeStickyNote : 20 : Apr 12, 2020 8:03:42 AM
Successfully finished executing the workflow

#### Overview

On AWS, the data normally resides in S3 buckets. HIVE tables are created pointing to data in the S3 buckets.

#### Details

- Fire would run on the master node of the EMR cluster, or on an Edge node with the cluster contiguration files.
- HIVE can be running on the same EMR cluster on running on another EMR cluster.
- Make sure to have the correct hive-site.xml on the cluster where the Spark jobs are running.
- Fire will automatically pick it up and be able to process it.

#### Writing to HIVE

Below is a workflow for writing to HIVE.

It reads housing.csv, creates a DataFrame and writes it out to a HIVE table.

### 17.1.11 Fire Integration with Redshift

Fire is fully integrated with Redshift. Fire has a number of Processors specifically for Redshift.

#### Redshift Processors

Fire has processors for reading from and writing to Redshift. They include:

- Read Redshift AWS
- Write Redshift AWS

### 17.1.12 Fire Integration with SageMaker

Fire is fully integrated with AWS SageMaker. Fire provides a number of processors for doing model building with SageMaker.

You can do Data Preparation and Feature Engineering with Sparkflows doing compute with Apache Spark. Sparkflows then seamlessly enables you to do your model training and deployment with SageMaker.

The above forms a very powerful combinations for end to end Machine Learning.

#### Spark Sagemaker Examples

There are a number of SageMaker-Spark examples by AWS here :

- https://github.com/aws/sagemaker-spark
- https://docs.aws.amazon.com/sagemaker/latest/dg/apache-spark-example1.html

### Fire SageMaker Processors

SageMaker Processors include:

- KMeansSageMakerEstimator
- XGBoostSageMakerEstimator
- LDASageMakerEstimator
- LinearLearnerBinaryClassifier
- LinearLearnerRegressor
- PCASageMakerEstimator
- SaveSageMaker

### AWS Provided Policies

AWS provides managed policies for SageMaker. Example : AmazonSageMakerFullAccess

### Launching EMR

When launching the EMR Cluster make sure that the Role (eg: **EMR_EC2_DefaultRole**) used has the AmazonSageMakerFullAccess policy.

Now that the Roles and Policies are in place, start up your EMR cluser with the **EMR_DefaultRole** and **EMR_EC2_DefaultRole** Roles.

### Create New Role

Create a new Role called **aws-sagmaker-full-access** with the below Policy. It would be used in the Apache Spark job when accessing SageMaker.

- AmazonSageMakerFullAccess

### Use ARN of the new Role in the Workflow

We now use the ARN of the new Role when we use the SageMaker KMeans Estimator Node in the Workflow.

**arn:aws:iam::account_id:role/aws-sagemaker-full-access**

### AWS Instance Types

AWS has various instance types:

- p : GPU Instances
- c : Compute Instances
- r : Memory Optimized Instances
- m : General Instances

Amazon SageMaker Instance Types details are here : https://aws.amazon.com/sagemaker/pricing/instance-types/

### Dataset Column Names for Training with Sagemaker

Sagemaker needs the following columns to exist in the Dataset.

- label : label column
- features : features column, this column can also be set

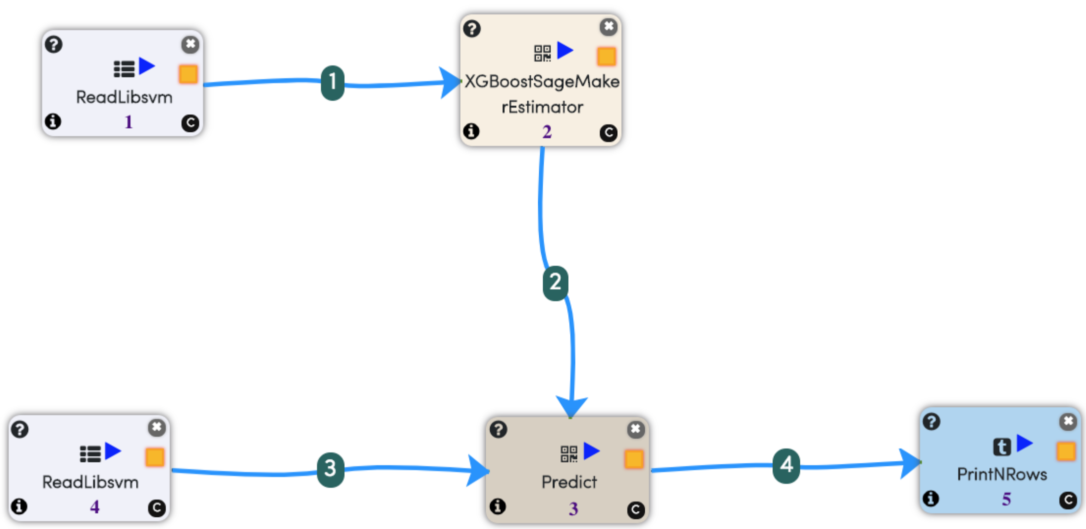
### Flow with Sparkflows and AWS

- We do the Data Preparation and Feature Generation in EMR with Sparkflows.
- When Sparkflows invokes the SageMakerEstimator, it calls SageMaker for Training and Deployment.
- Once the model is deployed on SageMaker, the endpoint can be used for realtime predictions.

### XGBoost Sagemaker Workflow

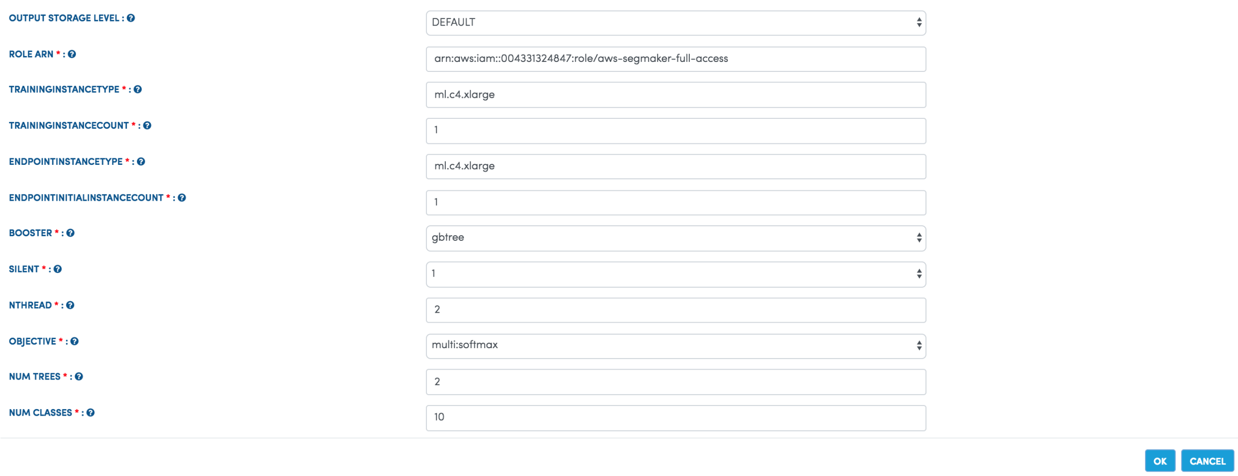
Below is a workflow which:

- Reads in a libsvm file as input
- Performs XGBoost Modeling
- Reads in another libsvm file
- Performs predictions with the model built in the previous step
- Prints out the result



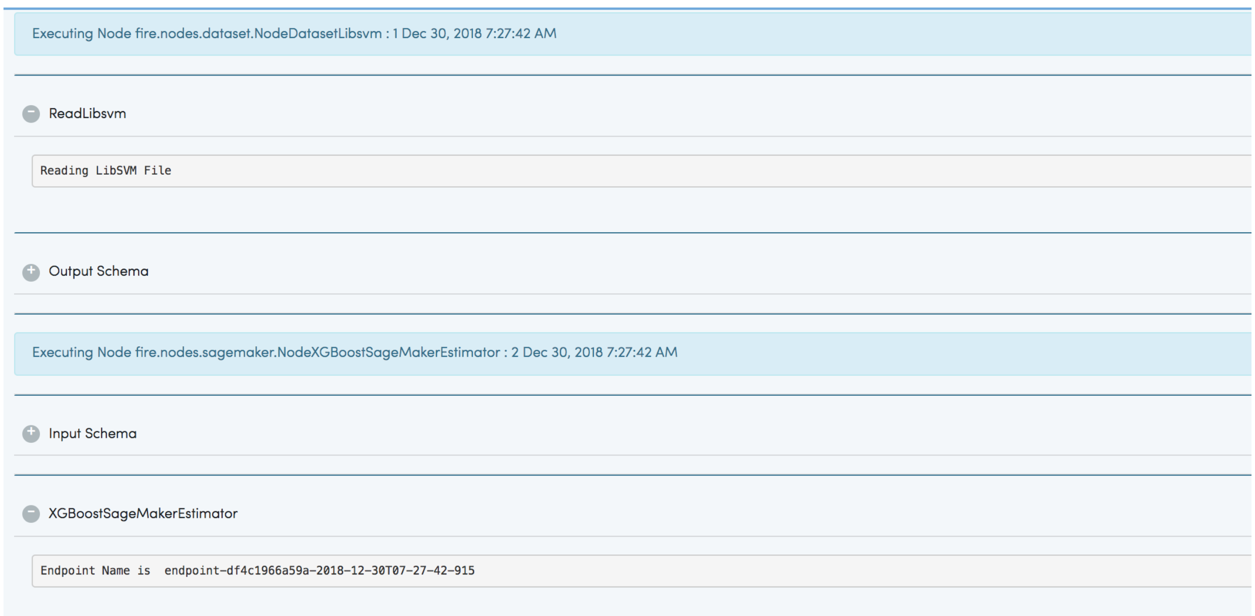

### XGBoost Configuration

Below are the configuration setup details of the XGBoost Processor.




### Executing the Workflow

Below are the results of executing the workflow.




## 17.1.13 Fire Integration with Kinesis

This document described Fire integration with Kinesis. Fire uses Apache Spark Structured Streaming Connector from Qubole.

https://github.com/qubole/kinesis-sql

### Install AWS CLI

Install AWS CLI:

```
https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
```

### Create an access key and secret key

Create an access key and secret key for the user (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_CreateAccessKey).

*Note* It's important to regularly rotate your access and secret keys. See this documentation for more information (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#rotating_access_keys_console)

### Configure AWS CLI

Configure AWS CLI:

```
https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html
aws configure region: us-east-1 aws_access_key_id = accesskeyid aws_secret_access_key␣
↪= awssecretaccesskey
```

### Create AWS Kinesis Stream

Create AWS Kinesis Stream:

```
aws kinesis create-stream --stream-name sparkflows_kinesis_test --shard-count 1
```

### Send message to AWS Kinesis from AWS CLI

Sending message to Kinesis:

```
aws kinesis put-record --stream-name sparkflows_kinesis_test --data file://data.json -
↪-partition-key uuidgen
```

### Update EMR_EC2_Default_Role

Update **EMR_EC2_DefaultRole** with **AmazonKinesisFullAccess** Policy so that our EMR Cluster would have full access to Kinesis.

### Or Create an IAM policy for accessing Amazon Kinesis

Create an IAM policy that defines what this user has access to in your AWS account. It's important to only grant this user minimal access within your account. See this documentation for how to create IAM policies (https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html).

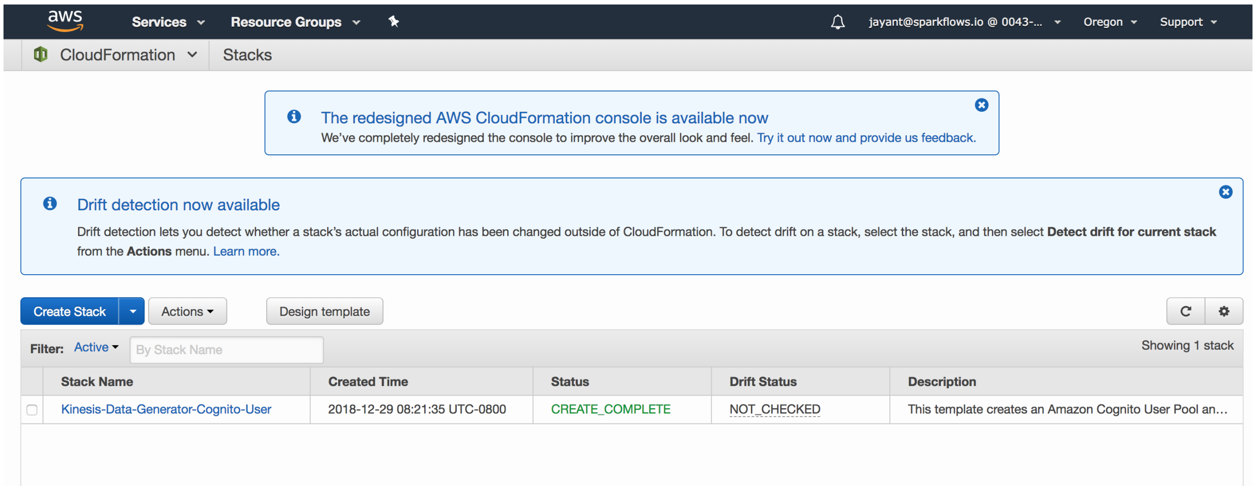
### Create EMR Cluster with the above Role

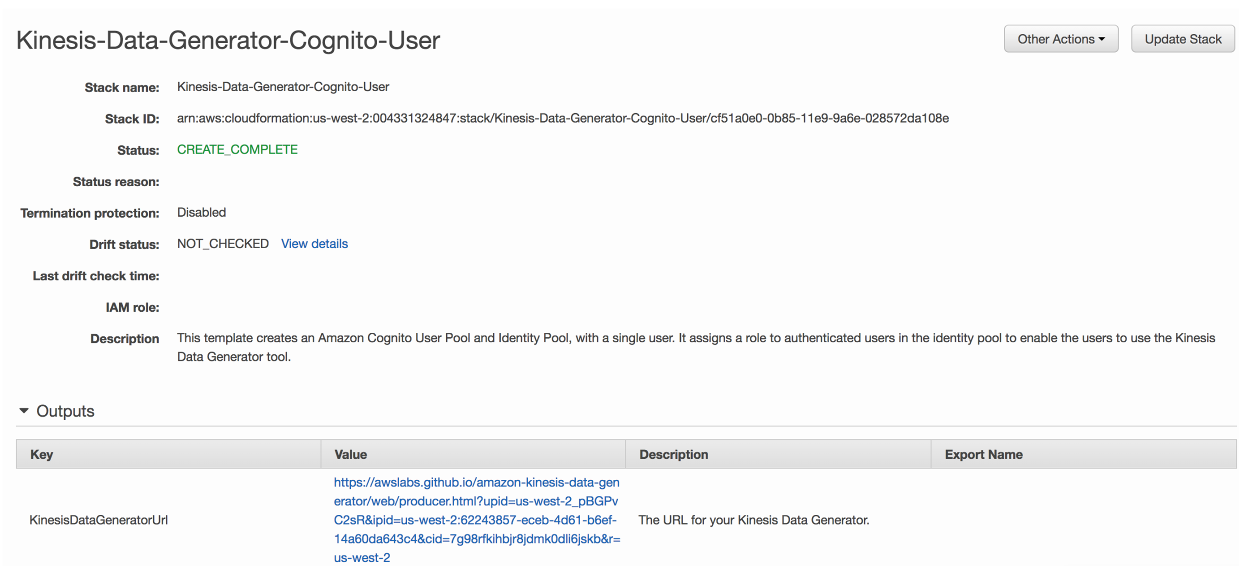
When we create the EMR Cluster with the above Role, it would have full access to Amazon Kinesis.

### Pushing data to Kinesis

AWS provides a Kinesis Data Generator. It can be configured for pushing random data in specified format to Kinesis.

https://awslabs.github.io/amazon-kinesis-data-generator/web/help.html




### Kinesis Workflow in Fire

Workflows can be easily built in Fire which read data from Kinesis, process them and save the results where needed.

### REFERENCE : Creating Access Key & Secret Key

1. You'll need create a user with programmatic access by following the steps here (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html).
2. Next, you'll create an IAM policy that defines what this user has access to in your AWS account. It's important to only grant this user minimal access within your account. See this documentation for how to create IAM policies (https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html).

3. Finally, you'll create an access key and secret key for this user (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_CreateAccessKey).

*Note* It's important to regularly rotate your access and secret keys. See this documentation for more information (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#rotating_access_keys_console)

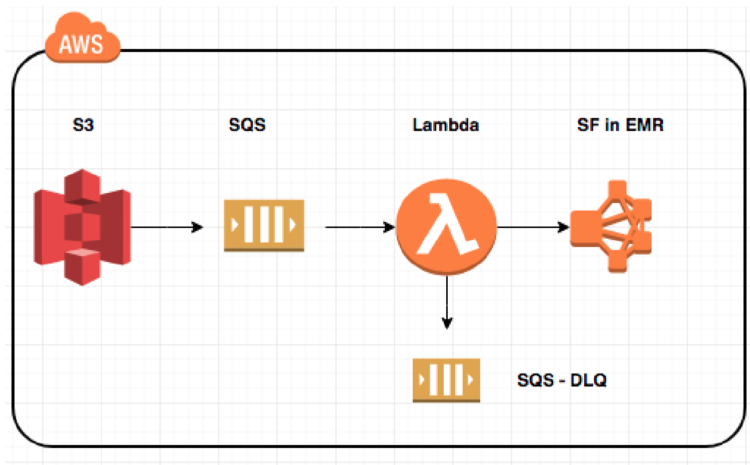
### 17.1.14 File Watcher with AWS & Sparkflows

#### Overview

There are many use cases where we have to process the incoming files on S3. This document describes one way to achieve it with SQS, Lambda and using the REST API of Fire Insights.

#### Design

The below diagram captures the high level design:




Below is the flow of execution:

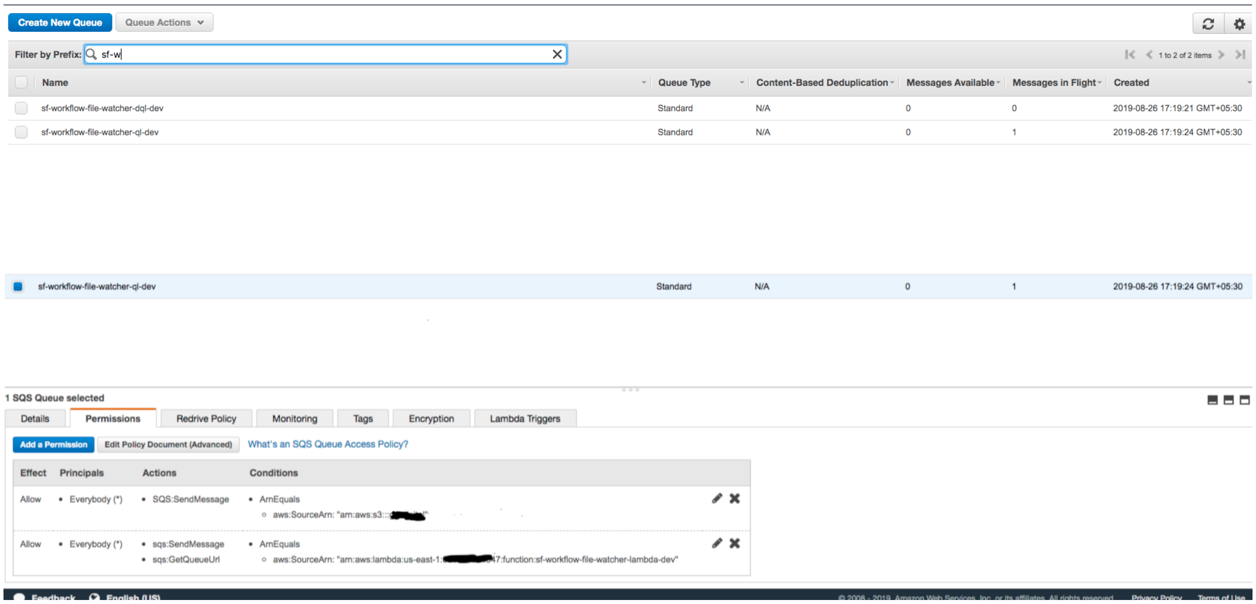
- New files arrives on S3 in the directory location `/sparklows-file-watcher/raw-data/iot/2019-08-2201`
  - In the above design, all the raw data comes into the directory `/sparklows-file-watcher/raw-data`
  - There are various types of raw data which can come.
  - `iot` is one type of raw data coming in. Each day we receive a number of iot files in the folder `/sparklows-file-watcher/raw-data/iot/yyyy-MM-dd`.
  - Once all the files for that date have been written to the appropriate folder, a _SUCCESS files is written into it.
- It triggers an event which is sent to a configured SQS queue.
- Once the event reaches SQS, it triggers an AWS Lambda.
- The AWS Lambda uses the Fire Insights REST API(http://docs.sparkflows.io/en/latest/rest-api-reference/workflow.html#execute) to execute a workflow to process the new incoming files in the AWS S3 bucket.
- If AWS Lambda fails, it sends the event to DLQ (Dead Letter Queue). It can be further handled from there based on the requirements.

### Create an SQS Queue

Create an SQS Queue for receiving the events from S3 and triggering the AWS Lambda function.

Below we see the SQS queue : `sf-workflow-file-watcher-ql-dev`.

It has the below permissions to receive the messages from S3 bucket and invoke the AWS Lambda function.




### Configure AWS S3 bucket to generate events

Configure the AWS S3 bucket to send events for the new files coming in to AWS SQS queue.

Below, it looks for the new files with prefix of `events` and suffix of `_SUCCESS`. It sends these events to `sf-workflow-file-watcher-ql-dev` SQS Queue.

### Create the AWS Lambda function

Create the AWS Lambda function to take the SQL Event and kick off the workflow in Fire Insights. This workflow would process the new files which came in.

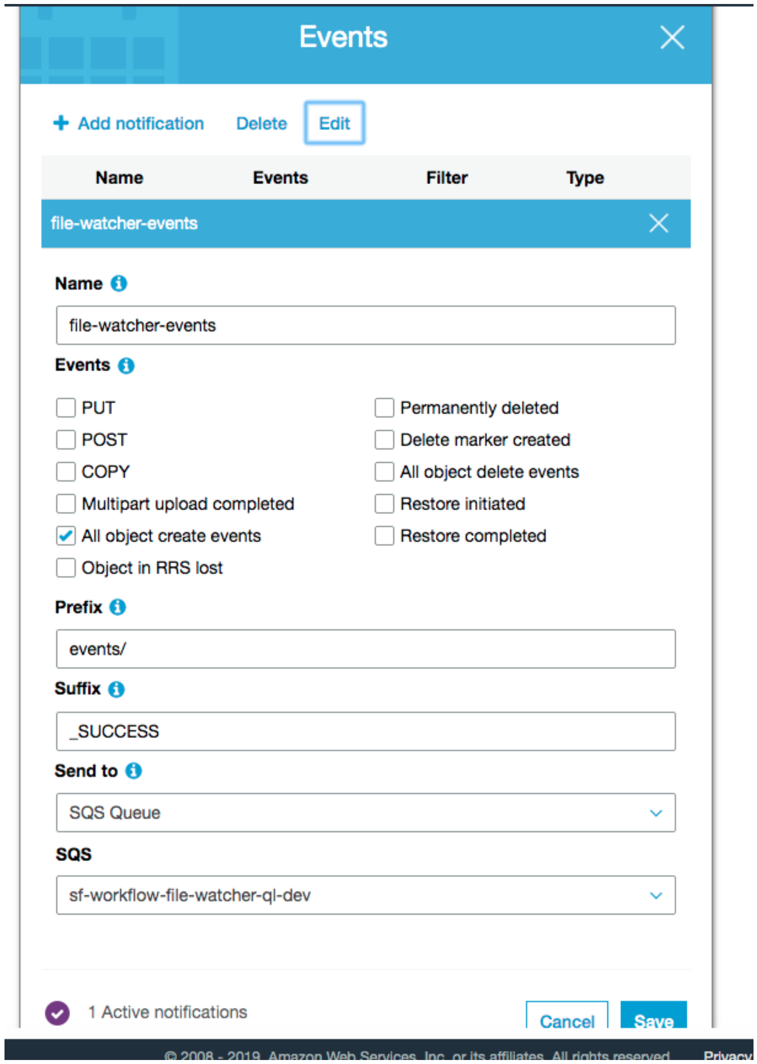
First create an IAM role. An example is shown below.

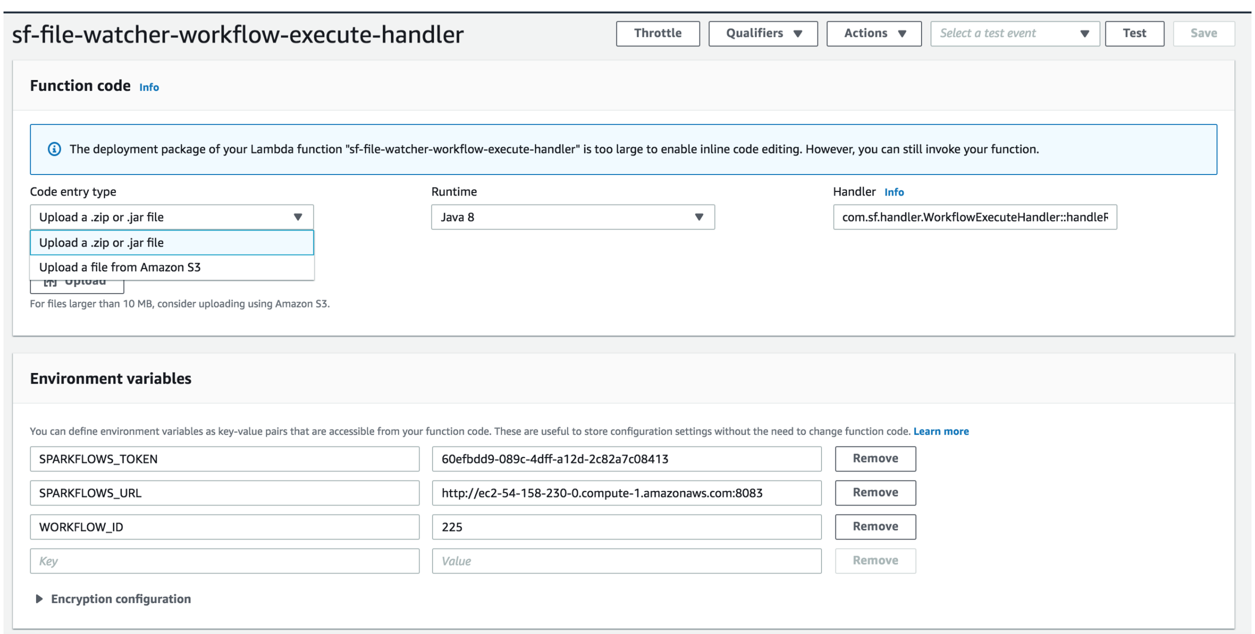
We add 3 Environment variables as shown below. These get used by the Lambda functions in this example.

- SPARKFLOWS_TOKEN or KMS_ARN
- SPARKFLOWS_URL
- WORKFLOW_ID

Instead of the Sparkflows token, users can encrypt the token using KMS and use the kms arn as the Environment variable and decrypt the token using kms inside the Lamdba.

Upload the jar file for the RequestHandler. It can also be placed into S3 location and the Lambda configured for it.


Events
Add notification
Delete
Edit
Name
Events
Filter
Type
file-watcher-events
Name
file-watcher-events
Events
PUT
POST
COPY
Multipart upload completed
All object create events
Object in RRS lost
Permanently deleted
Delete marker created
All object delete events
Restore initiated
Restore completed
Prefix
events/
Suffix
_SUCCESS
Send to
SQS Queue
SQS
sf-workflow-file-watcher-ql-dev
1 Active notifications
Cancel
Save


sf-file-watcher-workflow-execute-handler
Throttle
Qualifiers
Actions
Select a test event
Test
Save
Function code Info
The deployment package of your Lambda function "sf-file-watcher-workflow-execute-handler" is too large to enable inline code editing. However, you can still invoke your function.
Code entry type
Upload a .zip or .jar file
Upload a .zip or .jar file
Upload a file from Amazon S3
Runtime
Java 8
Handler Info
com.sf.handler.WorkflowExecuteHandler::handleF
For files larger than 10 MB, consider uploading using Amazon S3.
Environment variables
You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more
SPARKFLOWS_TOKEN
60efbdd9-089c-4dff-a12d-2c82a7c08413
Remove
SPARKFLOWS_URL
http://ec2-54-158-230-0.compute-1.amazonaws.com:8083
Remove
WORKFLOW_ID
225
Remove
Key
Value
Remove
Encryption configuration

### WorkflowExecuteHandler

```
package com.sf.handler

import com.amazonaws.services.lambda.runtime.events.SQSEvent
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage
import com.amazonaws.services.lambda.runtime.{Context, LambdaLogger, RequestHandler}
import com.amazonaws.services.s3.event.S3EventNotification
import com.amazonaws.services.s3.event.S3EventNotification.S3EventNotificationRecord
import com.sf.WorkflowExecute

import scala.collection.JavaConverters._

class WorkflowExecuteHandler extends  RequestHandler[SQSEvent, Unit] {

  private val token = System.getenv("SPARKFLOWS_TOKEN")
  private val sparkflowsURL = System.getenv("SPARKFLOWS_URL")
  private val workflowId = System.getenv("WORKFLOW_ID")

  def handleRequest(sqsEvent: SQSEvent, context: Context): Unit = {

    implicit val logger: LambdaLogger = context.getLogger

    logger.log(s"sparkflowsURL: $sparkflowsURL")
    logger.log(s"workflowId: $workflowId")

    sqsEvent
      .getRecords
      .asScala.map(sqsMessageToS3Event)
      .foreach(_.getRecords.asScala.foreach(processS3Record))
  }

  private[handler] def sqsMessageToS3Event(sqsMessage: SQSMessage):
↪S3EventNotification = {
    S3EventNotification.parseJson(sqsMessage.getBody)
  }

  private[handler] def processS3Record(s3EventRecord: S3EventNotificationRecord)
                                      (implicit logger: LambdaLogger): Unit = {

    val s3Entity = s3EventRecord.getS3
    val inputBucketName: String = s3Entity.getBucket.getName
    val inputObjectKey: String = s3Entity.getObject.getUrlDecodedKey
    val eventName: String = s3EventRecord.getEventName
    val path = s"s3://$inputBucketName/$inputObjectKey".replace("/_SUCCESS", "")

    logger.log(s"Event record $eventName; path $path")

    val body = s"""
                 |{
                 |   "workflowId": "${workflowId}",
                 |   "parameters": "--var datapath=${path}"
                 |}
       """.stripMargin

     val workflowStatus = WorkflowExecute.executeWorkflow(body, token, sparkflowsURL)
```

(continues on next page)

(continued from previous page)

```
    logger.log(s"Status of workflow $workflowStatus")
  }
}
```

### WorkflowExecute

```
package com.sf

import com.mashape.unirest.http.Unirest

object WorkflowExecute {

  def executeWorkflow(body: String, token: String, sparkflowsHostName: String) = {

    val workflow = Unirest.post(s"$sparkflowsHostName/api/v1/workflow/execute")
      .header("Content-Type", "application/json")
      .header("Cache-Control", "no-cache")
      .header("Authorization", s"Bearer $token")
      .body(body)
      .asString

    workflow match {
      case s if workflow.getStatus >= 200 && workflow.getStatus <= 300 => workflow.
↪getBody
      case f => throw SubmissionFailedException(s"Job submissions failed, status code␣
↪is ${f.getStatus}")
    }
  }
  case class SubmissionFailedException(message:String) extends Exception(message)
}
```

## 17.1.15 CloudFormation Template with Embedded H2 DB

### Overview

Using CloudFormation Templates, Fire can be easily installed on AWS. This CFT works with EMR 5.8 onwards.

The below steps would allow you to start up an EMR Cluster and have Fire setup on it.

The CFT does the following:

- Creates EMR cluster with 1 master node and 2 worker nodes by default.
- Once the cluster is ready it runs the job/script to deploy Fire (takes around 1-1:30 min for deploying app!).

### Relevant Files

Table 1: Below are the Relevant Files

| Title | Description | File |
|---|---|---|
| emr-file-h2.json | CloudFormation Template | https://s3.amazonaws.com/sparkflows-cft/h2-db/emr-fire-h2.json |
| deploy-fire-h2.sh | Script for deploying Fire | https://s3.amazonaws.com/sparkflows-cft/h2-db/deploy-fire-h2.sh |
| script-runner.jar | Script Runner | https://s3.amazonaws.com/sparkflows-cft/h2-db/script-runner.jar |

### Ports

- With this CFT and deploy-fire-h2.sh, when Fire comes up, it would be listening on ports 8085 and 8086.

### Download Files and Upload to your S3 Bucket

- Download CFT **emr-fire-h2.json** from the above link.
- Download **deploy-fire-h2.sh** and **script-runner.jar** from the above links and upload them to your s3 bucket

### Update Cloudformation template based on your environment

Update the CFT **emr-fire-h2.json** according to your requirement and environment in which you are deploying.

- ElasticMapReduce-Master-SecurityGroup under mastersg:

```
From AWS console -> EC2 -> Security Groups -> search for "ElasticMapReduce-master"
```

- ElasticMapReduce-Slave-SecurityGroup under slavesg:

```
From AWS console -> EC2 -> Security Groups -> search for "ElasticMapReduce-slave"
```

- Applications:

```
By default the CFT deploys Hadoop, Hive & Spark. Add any other Applications which␣
↪you need.
```

- EbsRootVolumeSize:

```
If required change the root(/) ebs volume size. By default CFT has 50GB disk␣
↪volume
```

- SizeInGB for Master and Core Instances:

```
If required change the SizeInGB under EbsConfiguration. By default CFT has 50GB␣
↪disk volume (used for hdfs)
```

- VolumesPerInstance for Master and Core Instances:

```
If required change the VolumesPerInstance under EbsConfiguration By default cft␣
↪has 1. It means one additional disk of 50GB added to each instance(for hdfs). e.
↪g. If you change it 2, two 50GB (SizeInGB size) disks will be added to each␣
↪instances.
```

- deploy-fire-h2.sh and script-runner.jar:

```
Change the s3 bucket path for these two files, this s3 bucket  must be same␣
↪bucket as S3Bucket. You'll pass the S3Bucket value while creating the␣
↪cloudformation stack.
```

### Steps to Create EMR Cluster and Deploy Fire

- **AWS web Console -> Management tools -> CloudFormation**
  - Click on **Create Stack**.
- Next page is **Select Template**
  - Select the radio-button **Upload a template to Amazon S3**
  - Select the updated **emr-fire-h2.json** from your system
  - Click Next
- Next page is **Specify Details**
  - Enter CloudFormation stack name

Table 2: Update Parameters where needed

| Name of Parameter | Description |
|---|---|
| AdditionalSecurityGroups | From the list choose the additional secuirty group(sg), it's required because default emr sg's ports are not opened for ssh, fire & etc… |
| AmiId | EMR cluster can be launched using Custom AMI, pass the value if you have a Custom AMI |
| ClusterName | Name for EMR Cluster |
| CoreInstanceType | Provide the required instance type for core nodes, default instance type is m4.xlarge |
| CoreNodes | Choose the required number of core nodes, by default it's 2 |
| EmrVersion | Choose the required EMR version, it's should be above EMR v.5.8.x |
| Environment | By default dev |
| FireVersion | Enter the required version of Fire |
| KeyName | Enter the valid pem key name to connect to emr nodes |
| MasterInstanceType | Provide the required instance type for master nodes, default instance type is m4.xlarge |
| MasterNodes | By default 1 |
| Owner | provide the name of a team or person creating the cluster |
| ReleaseVersion | Enter the required ReleaseVersion, it has to match with fire version |
| S3Bucket | Provide the s3 bucket name, this s3 bucket should be same s3 bucket where deploy-fire-h2.sh and script-runner.jar are uploaded |
| Subnet | Provide the proper subnet name, which has sufficient resources to create emr cluster |
| TaskInstanceType | Optional, required only if you're choosing TaskNodes. Provide the required instance type for task nodes, default instance type is m4.xlarge |
| TaskNodes | Optional, required only if you want to create the cluster with tasknodes.By default zero, enter the required number of nodes |

- Click Next
- Next Page is **Options**
  - If required (not mandatory) enter tag details
  - Click Next
- Next Page is **Review**

- Review all the details provided to create an EMR stack
- Click on Create
- It will start creating the Stack

- Next page is back to **Cloudformation Page**
  - Choose your Stack name
  - Click on **Events** to check the process
  - Click on **Resources** to get the EMR Cluster id
- Once the stack runs successfully, your EMR Cluster and Fire is ready to use. Cluster creation time depends on your EMR cluster configuration
- To **cross check** the Fire installation
  - Go to EMR from AWS web console
  - Choose your EMR Cluster
  - Identify the Master Node Public DNS
  - Go to `http://masternodeip:8085/index.html`

### Connect Fire to the New Cluster

- Go to `User/Administration`
- Click on `Infer Hadoop Configuration`
- Click on the `Save` button

### Load Examples

- In Fire, click on `Load Examples`
- `ssh` to the master node
- `cd /opt/fire/fire-3.1.0`
- Upload the example data files to HDFS
  - `hadoop fs -put data`

### Create hadoop user

- Go to `Administration/User`
- Click on `Add User`
- Create a new user with username `hadoop`
- Log out and log back in as user `hadoop`

### Start running the Examples

- Go to `Applications`
- Start creating/using the Applications

#### Summary

Using the above CFT you have your EMR cluster with Fire running seamlessly.

### 17.1.16 CloudFormation Template with MySQL

#### Overview

Using CloudFormation Templates, Fire can be easily installed on AWS. This CFT works with EMR 5.8 onwards.

The below steps would allow you to start up an EMR Cluster and have Fire setup on it.

The CFT does the following:

- Creates External DB for Fire to be used as the metastore for Fire data
- Creates EMR cluster with 1 master node and 2 worker nodes by default.
- Once the cluster is ready it runs the job/script to deploy Fire (takes around 1-1:30 min for deploying app!).

#### Relevant Files

Table 3: Below are the Relevant Files

| Title | Description | File |
|---|---|---|
| emr-file-mysql.json | CloudFormation Template | https://s3.amazonaws.com/sparkflows-cft/mysql-db/emr-fire-mysql.json |
| deploy-fire-mysql.sh | Script for deploying Fire with MySQL | https://s3.amazonaws.com/sparkflows-cft/mysql-db/deploy-fire-mysql.sh |
| script-runner.jar | Script Runner | https://s3.amazonaws.com/sparkflows-cft/mysql-db/script-runner.jar |

#### Ports

- With this CFT and deploy-fire-mysql.sh, when Fire comes up, it would be listening on ports 8085 and 8086.

#### Download Files and Upload to your S3 Bucket

- Download CFT **emr-fire-mysql.json** from the above link.
- Download **deploy-fire-mysql.sh** and **script-runner.jar** from the above links and upload them to your s3 bucket

#### Update Cloudformation template based on your environment

Update the CFT **emr-fire-mysql.json** according to your requirement and environment in which you are deploying.

- ElasticMapReduce-Master-SecurityGroup under mastersg:

```
From AWS console -> EC2 -> Security Groups -> search for "ElasticMapReduce-master"
```

- ElasticMapReduce-Slave-SecurityGroup under slavesg:

```
From AWS console -> EC2 -> Security Groups -> search for "ElasticMapReduce-slave"
```

- Applications:

```
By default the CFT deploys Hadoop, Hive & Spark. Add any other Applications which␣
↪you need.
```

- EbsRootVolumeSize:

```
If required change the root(/) ebs volume size. By default CFT has 50GB disk␣
↪volume
```

- SizeInGB for Master and Core Instances:

```
If required change the SizeInGB under EbsConfiguration. By default CFT has 50GB␣
↪disk volume (used for hdfs)
```

- VolumesPerInstance for Master and Core Instances:

```
If required change the VolumesPerInstance under EbsConfiguration By default cft␣
↪has 1. It means one additional disk of 50GB added to each instance(for hdfs). e.
↪g. If you change it 2, two 50GB (SizeInGB size) disks will be added to each␣
↪instances.
```

- deploy-fire-mysql.sh and script-runner.jar:

```
Change the s3 bucket path for these two files, this s3 bucket  must be same␣
↪bucket as S3Bucket. You'll pass the S3Bucket value while creating the␣
↪cloudformation stack.
```

### Steps to Create EMR Cluster and Deploy Fire

- **AWS web Console -> Management tools -> CloudFormation**
  - Click on **Create Stack**.
- Next page is **Select Template**
  - Select the radio-button **Upload a template to Amazon S3**
  - Select the updated **emr-fire-mysql.json** from your system
  - Click Next
- Next page is **Specify Details**
  - Enter CloudFormation stack name

Table 4: Update Parameters where needed

| Name of Parameter | Description |
|---|---|
| AdditionalSecurityGroups | From the list choose the additional secuirty group(sg), it's required because default emr sg's ports are not opened for ssh, fire & etc… |
| AmiId | EMR cluster can be launched using Custom AMI, pass the value if you have a Custom AMI |
| ClusterName | Name for EMR Cluster |
| CoreInstanceType | Provide the required instance type for core nodes, default instance type is m4.xlarge |
| CoreNodes | Choose the required number of core nodes, by default it's 2 |
| EmrVersion | Choose the required EMR version, it's should be above EMR v.5.8.x |
| Environment | By default dev |
| FireVersion | Enter the required version of Fire |
| KeyName | Enter the valid pem key name to connect to emr nodes |
| MasterInstanceType | Provide the required instance type for master nodes, default instance type is m4.xlarge |
| MasterNodes | By default 1 |
| Owner | provide the name of a team or person creating the cluster |
| ReleaseVersion | Enter the required ReleaseVersion, it has to match with fire version |
| S3Bucket | Provide the s3 bucket name, this s3 bucket should be same s3 bucket where deploy-fire.sh and script-runner.jar are uploaded |
| Subnet | Provide the proper subnet name, which has sufficient resources to create emr cluster |
| TaskInstanceType | Optional, required only if you're choosing TaskNodes. Provide the required instance type for task nodes, default instance type is m4.xlarge |
| TaskNodes | Optional, required only if you want to create the cluster with tasknodes.By default zero, enter the required number of nodes |

- Click `Next`
- Next Page is **Options**
  - If required (not mandatory) enter tag details
  - Click `Next`
- Next Page is **Review**
  - Review all the details provided to create an EMR stack
  - Click on `Create`
  - It will start creating the Stack
- Next page is back to **Cloudformation Page**
  - Choose your Stack name
  - Click on `Events` to check the process
  - Click on `Resources` to get the EMR Cluster id
- Once the stack runs successfully, your EMR Cluster and Fire is ready to use. Cluster creation time depends on your EMR cluster configuration
- To **cross check** the Fire installation
  - Go to EMR from AWS web console
  - Choose your EMR Cluster
  - Identify the Master Node Public DNS
  - Go to `http://masternodeip:8085/index.html`

### Connect Fire to the New Cluster

- Go to `Administration/Configuration`
- Click on `Infer Hadoop Configuration`
- Click on the `Save` button

### Load Examples

- In Fire, click on `Load Examples`
- `ssh` to the master node
- `cd /opt/fire/fire-3.1.0`
- `hadoop fs -put data`

### Create hadoop user

- Go to `Administration/User`
- Click on `Add User`
- Create a new user with username `hadoop`
- Log out and log back in as user `hadoop`

### Start running the Examples

- Go to `Applications`
- Start building your Applications.

### Summary

Using the above CFT you have your EMR cluster with Fire running seamlessly.

# CHAPTER 18

# AZURE Integration

## 18.1 AZURE Guide

### 18.1.1 Introduction

Fire Insights is the flagship product from Sparkflows. It is seamlessly integrated with Azure. With Fire Insights you can perform self-serve data processing, analytics and machine learning on Azure.

Fire Insights integrates with Azure Databricks, ADLS, HDInsight etc.

Fire Insights comes with a number of components including:

- **Workflow Editor** : To create workflows for data processing, analytics and machine learning.
- **300+ Processors** : These include reading data from various stores, data processing, machine learning and visualizations.
- **Execution Engine** : For executing the workflow on Azure VM or HDInsight
- **Scheduler** : For scheduling running the workflows at certain time intervals
- **I-Dashboard** : For Visualization using chart, dashboard

### 18.1.2 Deployment Guide

Fire Insights can be easily installed on an Azure Standalone VM.

#### prerequisite:

- java 8 should be installed
- if you do not already have it, Need to install
- Download it from below link:

```
https://www.oracle.com/in/java/technologies/javase/javase-jdk8-downloads.html
```

- Install using below command (Centos):

```
yum localinstall jdk-8uxxx-linux-x64.rpm
```

- Set the below in .bash_profile:

```
export JAVA_HOME=/usr/java/jdk1.8.0_xxx-amd64/
```

Below are the overall steps for installing Fire Insights on VM.

- *ssh into the Azure VM*
- *Download Fire Insights* from https://www.sparkflows.io/download
- *Unzip it*
- *Create H2 Database*
- *Start Fire*

### Steps

- Create a VM on Azure:

```
Create a vm if you do not already have it running.
```

- Update the inbound rule

```
- ssh port ie 22 should be accessible to ssh to Azure VM.
- We would have Fire listening on ports 8080, so just ensure its opened.
```

- ssh into the VM:

```
ssh -i my.pem userp@public ip.
```

- Just Confirm that java 8 is already installed, if not follow above steps:

```
java -version
```

- Download the fire tgz file by one of the following options:
  - **https://www.sparkflows.io/download** OR
  - **https://www.sparkflows.io/archives** OR
  - wget https://s3.amazonaws.com/sparkflows-release/fire/rel-x.y.z/2/fire-x.y.z.tgz
- Unpack it:

```
tar xvf fire-x.y.z.tgz
```

- Create H2 DB:

```
Fire stores its metadata into the embedded H2 database. You can also connect it␣
↪to an external MySQL database.

  cd <fire install_dir>
  ./create-h2-db.sh
```

- Launch Fire Server:

```
cd <fire install_dir>
./run-fire-server.sh start
```

- Open your web browser and navigate to:

```
<machine_ip>:8080
```

- Login with the following default username and password:

```
username : admin
password : admin
```

### Loading Example Workflows

- From the home page of Fire Insights, click on ***Load Example Applications***
- Upload the Fire examples data with default or if data is available at anyother location, point to that location:

### Install and Start Running Example Workflows

- Start off with executing the example workflows:

```
- Fire comes pre-packaged with a number of example workflows, you can start␣
↪executing.
```

## 18.1.3 Azure Databricks Integration Steps

Fire Insights integrates with Databricks. It submits jobs to the Databricks clusters using the REST API of Databricks and have the results displayed back in Fire Insights.

Fire also fetches the list of Databases and Tables from Databricks, making it easier for the user to build their workflows and execute them. In addition Fire displays the list of Databricks clusters running for the user.

- Running Databricks on Azure : https://docs.microsoft.com/en-us/azure/azure-databricks/quickstart-create-databricks-workspace-portal

Below are the steps for Integrating Fire Insights with your Databricks Clusters.

### Install Fire Insights

Install Fire Insights on any machine. The machine has to be reachable from the Databricks cluster.

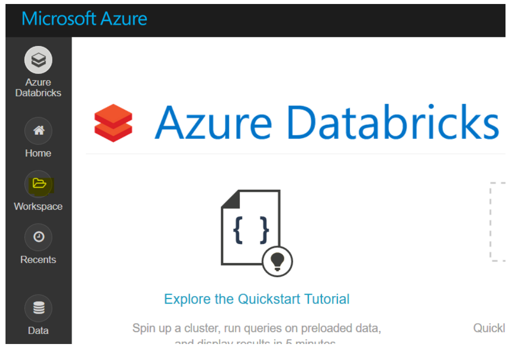
### Upload Fire Core Jar to Databricks

Upload Fire Insights jar to Databricks. Fire Insights jobs running on Databricks make use of this jar file.

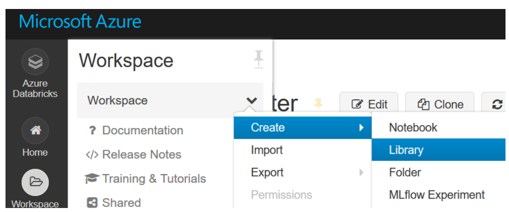
Upload `fire-x.y.z/fire-core-lib/fire-spark_2_4-core-3.1.0-jar-with-dependencies.jar` to Databricks. Upload it under Workspace as a Library on to Databricks.

## 1. Login to `Databricks Cluster`

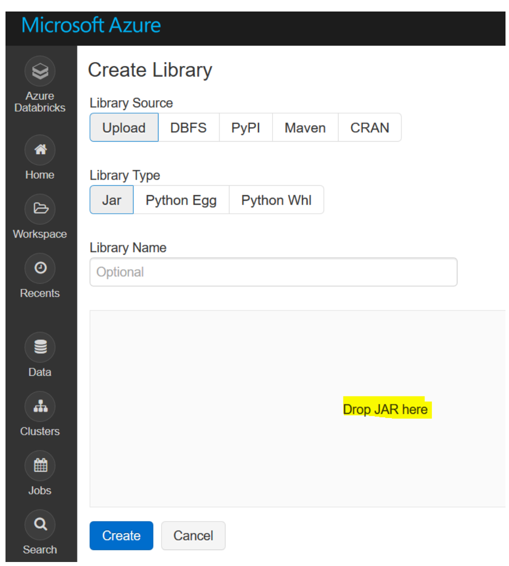
## 2. Click on `workspace` in the left side pane



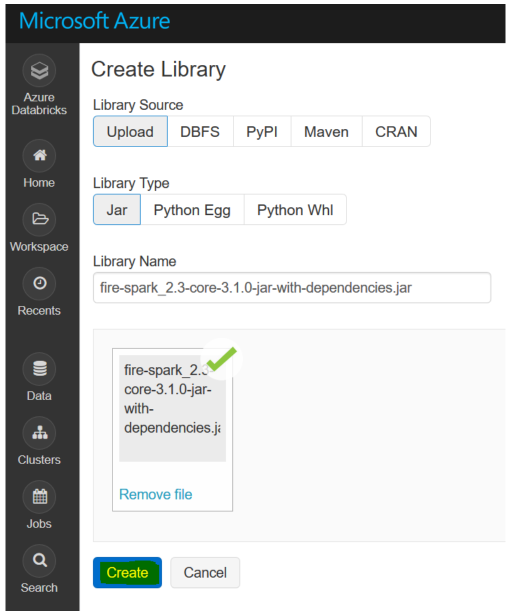

## 3. Create a new Library




## 4. Upload `fire-spark_2_4-core-3.1.0-jar-with-dependencies.jar` from your machine by Clicking on `Drop JAR here`



### 5. Once `fire-spark_2_4-core-3.1.0-jar-with-dependencies.jar` is uploaded, click on `Create`



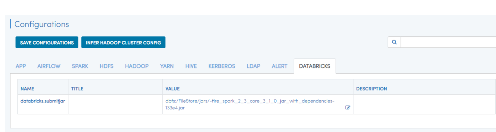

- Check the box with `Install automatically on all clusters`, in order to avoid installing it manually to every cluster.

---


### Configure the Uploaded Library in Fire Insights

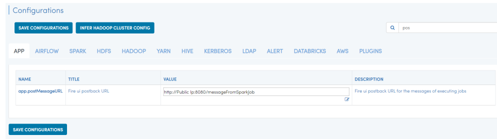
Configure the path of the uploaded fire core jar library in Databricks in Fire Insights.

This has to be done under Administration/Configuration.



### Configure app.postMessageURL in Fire Insights

Configure `app.postMessageURL` to be the IP of the machine on which Fire Insights is installed. Jobs running on Databricks would post back results to Fire Insights using this URL.



### Install Databricks JDBC Driver

Fire needs the Databricks JDBC Driver to be installed. Install it in the `fire-user-lib` and `fire-server-lib` folder of the Fire installation.

You can download the Databricks JDBC Driver from the Databricks site :

- https://docs.databricks.com/bi/jdbc-odbc-bi.html
- https://databricks.com/spark/odbc-driver-download

The driver is available as a zip file. eg: `SimbaSparkJDBC-2.6.3.1003.zip`

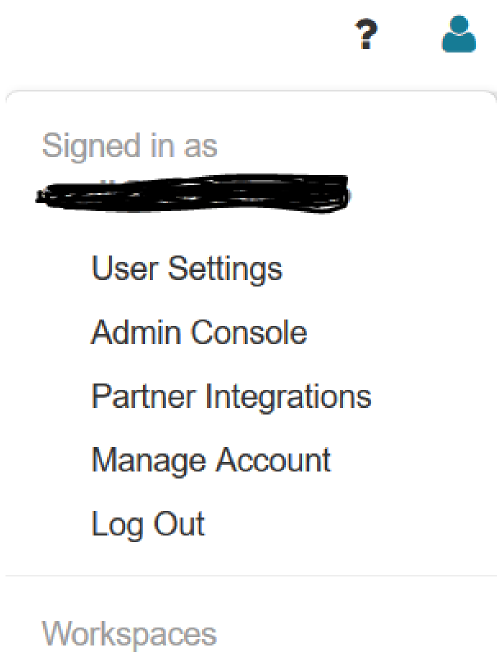
- Unzip the downloaded file. It will create a directory like `SimbaSparkJDBC-2.6.3.1003`
- Copy the jdbc jar file named `SparkJDBC4.jar` into `fire-x.y.z/fire-user-lib` and `fire-x.y.z/fire-server-lib`

### Create your REST API token in Databricks

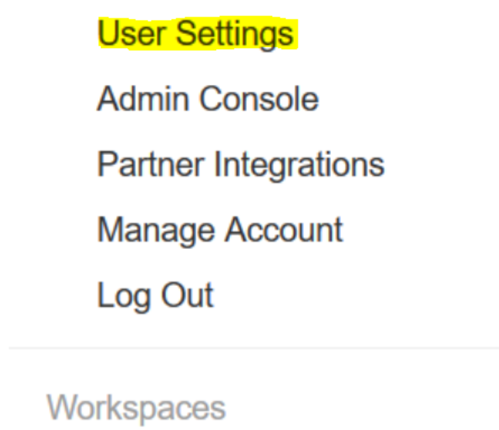
Create your token in Databricks. It would be used in making REST API calls to Databricks from Fire Insights.

### 1. Login to your Databricks Account

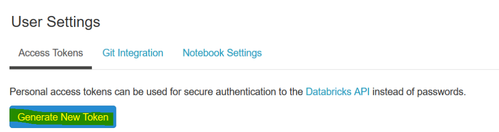
### 2. Click on `Account` icon in right corner top



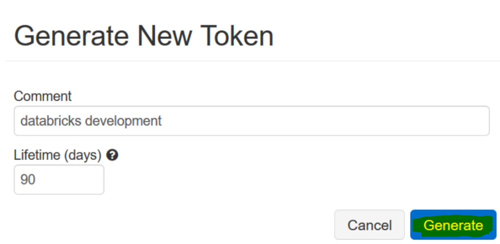
### 3. Click on `User Settings`




### 4. Click on `Generate New Token`




### 5. Add `comment` & `Lifetime(days)` for token expiry & Click on `Generate`




### 6. Copy the token generated. Click on `DONE`

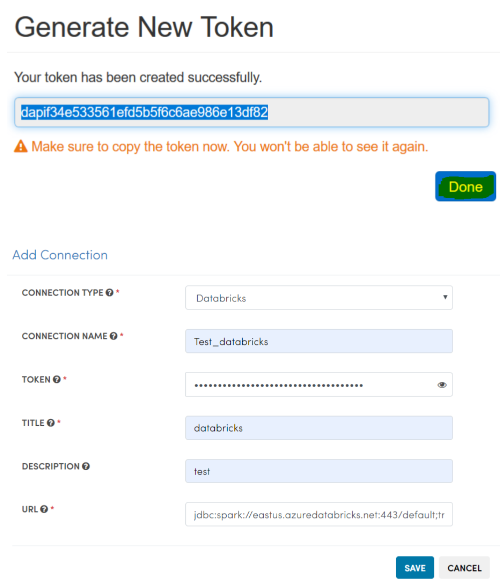
## Create Databricks Connection in Fire Insights

Create a connection in Fire Insights to Databricks.

It can be created by the Administrator under Administration/Global Connections. These connections are available for everyone to use.

It can also be created by any user with their Application. In this case, it is only available to the Application and its users.

- Specify your Databricks Token.
- Specify the Databricks JDBC URL of your cluster in Databricks.




Now we are ready to start using the Databricks Connection in Fire Insights to:

- Browse DBFS
- View your Databricks Clusters
- Browse your Databricks Databases & Tables
- Create Workflows which Read from and Write to Databricks

## 18.1.4 ADLS Integration

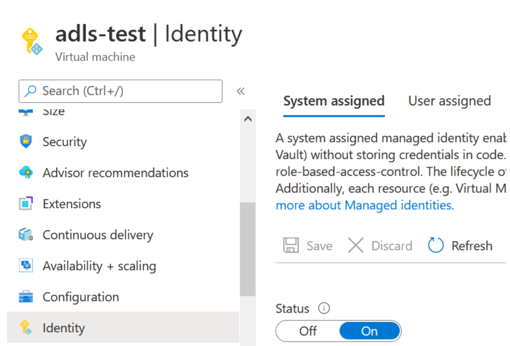
Fire Insights integrated with azure data lake storage, once configured you can use the filesystem for accessing data from it.

Below are the steps to Configured adls using managed identity

Managed identity allow the users to access the azure resources without hardcoding any credentials in code.

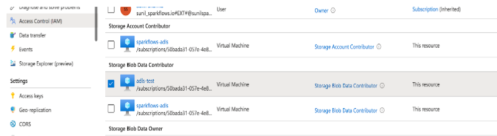
### System identity need to be enabled

System identity need to be enabled on vm where Fire Insights is running or need to be install



### In storage account, add the role to provide the access

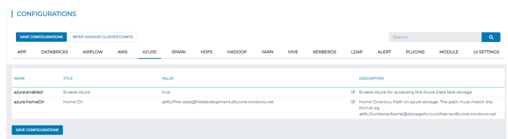
In storage account, add the role to provide the access to Azure vm with needed access



### login to Fire Insights

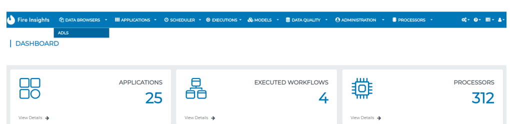
login to Fire Insights application and add below parameter in Configuration under administration section for AZURE.

- `azure.enabled to true`
- `azure.homeDir as abfs://containerName@storageAccountName.dfs.core.windows.net`




### Save Configuration

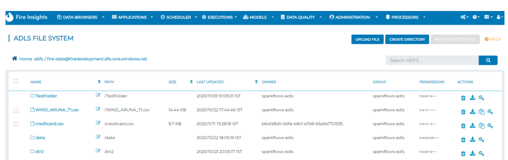
Save the above configuration and refresh the page & Click on Data browser to see ADLS page




### Click on Data browser

Click on ADLS to see ADLS FILESYSTEM in *DATA BROWSERS*

Once the above configurations done, you can start using those file while creating dataset and workflow.


Fire Insights
DATA BROWSERS
APPLICATIONS
SCHEDULER
EXECUTIONS
MODELS
DATA QUALITY
ADMINISTRATION
PROCESSORS
ADLS FILE SYSTEM
UPLOAD FILE
CREATE DIRECTORY
MOVE SELECTED FILES
HELP
Home abfs: / fire-data@firedevelopment.dfs.core.windows.net
Search HDFS
NAME
PATH
SIZE
LAST UPDATED
OWNER
GROUP
PERMISSIONS
ACTIONS
TestFolder /TestFolder 2020/11/09 10:09:21 IST sparkflows-adls sparkflows-adls rwxr-x---
WIND_ARUNA_T1.csv /WIND_ARUNA_T1.csv 14.44 KB 2020/10/22 17:44:46 IST sparkflows-adls sparkflows-adls rw-r-----
creditcard.csv /creditcard.csv 9.7 KB 2020/11/11 13:28:18 IST b64f28d0-66fa-48c1-a7d8-65a9a77c1535 sparkflows-adls rw-r--r--
data /data 2020/10/22 18:05:19 IST sparkflows-adls sparkflows-adls rwxrwx---
dir2 /dir2 2020/10/23 20:05:17 IST sparkflows-adls sparkflows-adls rwxr-x---

CHAPTER 19

# Load Balancer Integration

## 19.1 Load Balancer

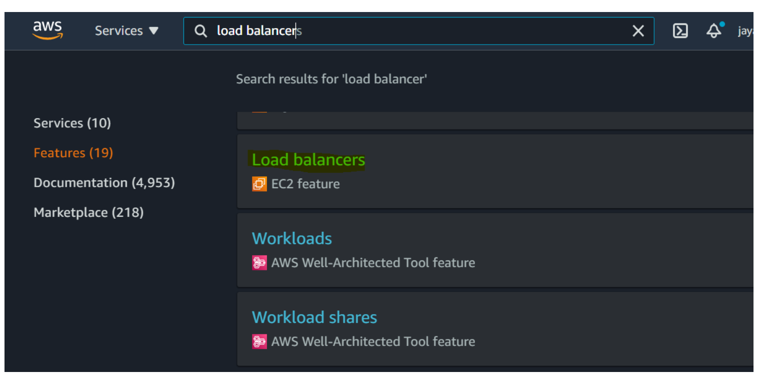
Below are steps to Configure Network Load balancer and route using Route 53 in AWS

### 19.1.1 AWS Network Load balancer

It Explains about Creating Network Load balancer in AWS and Configuring it VM running with Fire Insights.

Below are steps involved in Creating Network Load balancer in AWS.

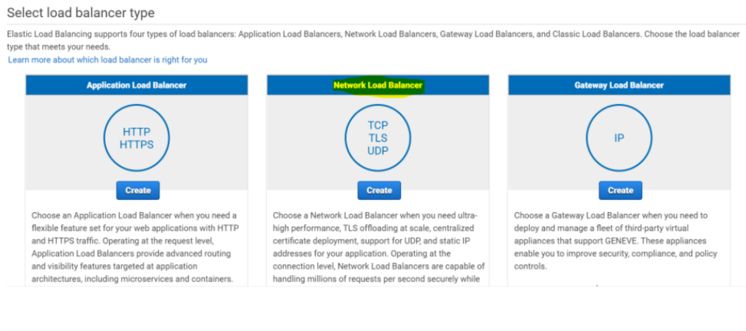
- Login with AWS Console and search for load balancer with EC2 feature.




- Create Load Balancer & select Network Load Balancer.
- Configure Load balancer

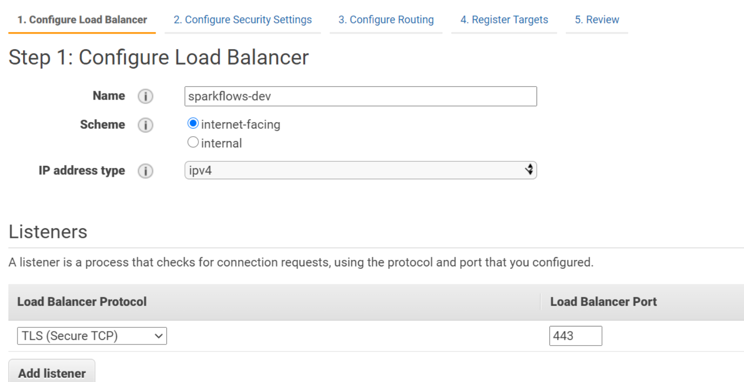
```
Add Name
Scheme : internet-facing
IP address type : ipv4
```

(continues on next page)




(continued from previous page)

```
Listeners
Load Balancer Protocol : TLS (SECURETCP)  Port: 443
Availability Zones
VPC : select VPC where application vm is running.
Availability Zones : select the specific zone.
```

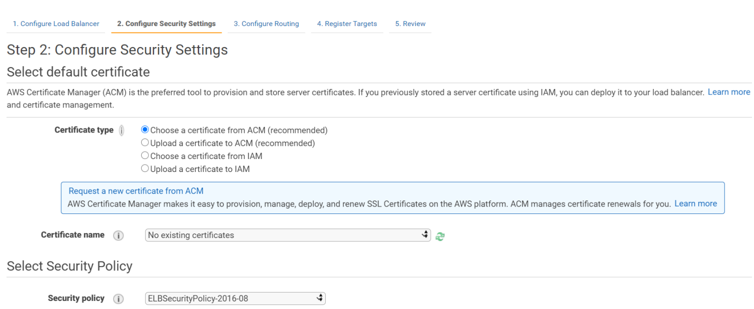



- Configure Security Settings

Select default certificate.

AWS Certificate Manager (ACM) is the preferred tool to provision and store server certificates. If you previously stored a server certificate using IAM, you can deploy it to your load balancer.

```
Certificate type
Certificate name
Security policy
```

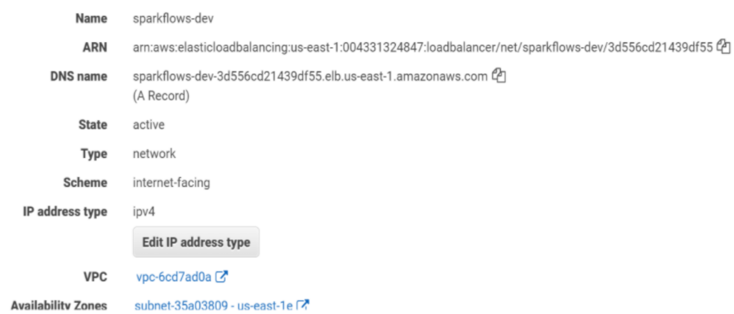


**Note:** Make sure to add certificate either through ACM or IAM

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https-ssl-upload.html

- Configure Routing

```
Target group
Name : A name of target group
Target type :  Instance
Protocol : TCP
Port : 80
Register Target
```




- Port forwarding

As Fire Insights by default running on port 8080 for HTTP & 8443 for HTTPS, Make sure forward HTTP or HTTPS to specified port on which Fire Insights is running.

```
sudo firewall-cmd --add-forward-port=port=443:proto=tcp:toport=8443 --permanent
sudo firewall-cmd --reload
```

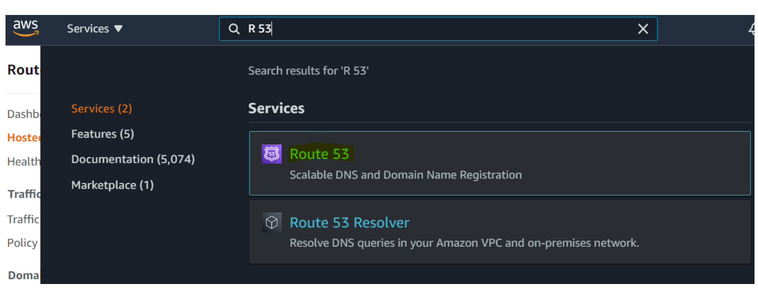
## 19.1.2 Route 53

It Explains about Configuring Route 53 to Network Load balancer.

Below ares steps to follow:

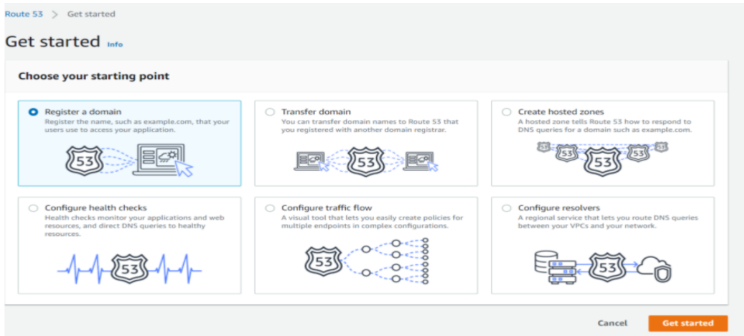
- Login to AWS Console and Type R 53 in search box

Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/



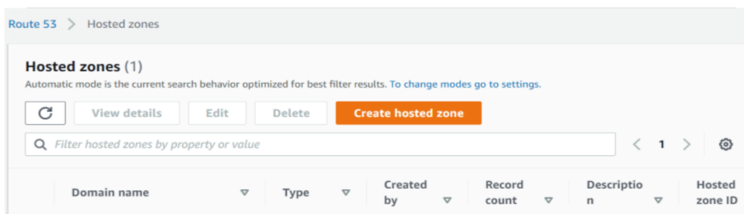

- Get started with R 53 Dashboard

```
Register a domain
```




- Hosted zone

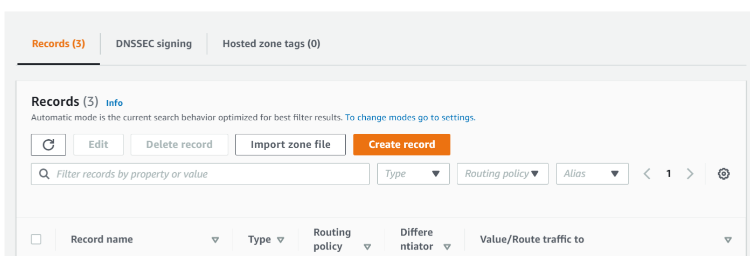
Create hosted zone




- Create records

Create records and Registered Network load balancer to it.

```
Value/Route traffic to : Alias to Network LB
Select Zone
By default load balancer domain name should be populated.
Record type : A -Routes traffic to IPV4 address and some aws resources.
Routing policy : Simple Routing
```




Record name Info
To route traffic to a subdomain, enter the subdomain name. For example, to route traffic to blog.example.com, enter blog. If you leave this field blank, the default record name is the name of the domain.
blog
.sparkflows.net
Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [ \ ] ^ _ ` { | } . ~
Value/Route traffic to Info
The option that you choose determines how Route 53 responds to DNS queries. For most options, you specify where you want to route internet traffic.
Alias to Network Load Balancer
US East (N. Virginia) [us-east-1]
sparkflows-dev-3d556cd21439df55.elb.us-east-1.amazonaws.com.
Record type Info
The DNS type of the record determines the format of the value that Route 53 returns in response to DNS queries.
A – Routes traffic to an IPv4 address and some AWS resources
Choose when routing traffic to AWS resources for EC2, API Gateway, Amazon VPC, CloudFront, Elastic Beanstalk, ELB, or S3. For example: 192.0.2.44.
Routing policy Info
The routing policy determines how Amazon Route 53 responds to queries.
Simple routing
Evaluate target health
Select Yes if you want Route 53 to use this record to respond to DNS queries only if the specified AWS resource is healthy.

CHAPTER 20

# Superset

## 20.1 Superset

Superset enables powerful Visualizations. Superset can connect with Databricks clusters and display data from Tables in Databricks.

Below are steps involved in Installing Superset and Configuring to Databricks.

### 20.1.1 Installation

Ensure that Superset machine has python 3.6.0+ installed on it.

#### Steps involved in installing apache superset (centos7)

- Install Superset:

```
pip install apache-superset
```

- Initialize the database:

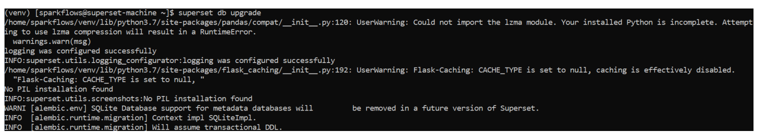
```
superset db upgrade
```

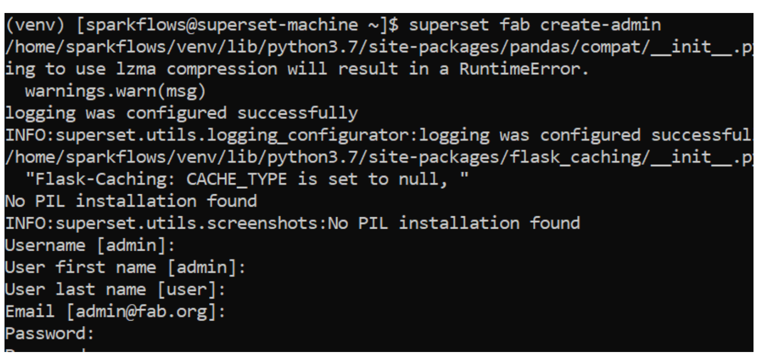
- Create an admin user (you will be prompted to set a username, first and last name before setting a password):

```
export FLASK_APP=superset
superset fab create-admin
```

- Load some data to play with:


(venv) [sparkflows@superset-machine ~]$ pip install apache-superset
Requirement already satisfied: apache-superset in ./venv/lib/python3.7/site-packages (0.38.0)
Requirement already satisfied: click<8 in ./venv/lib/python3.7/site-packages (from apache-superset) (7.1.2)
Requirement already satisfied: flask-compress in ./venv/lib/python3.7/site-packages (from apache-superset) (1.8.0)
Requirement already satisfied: flask-talisman in ./venv/lib/python3.7/site-packages (from apache-superset) (0.7.0)
Requirement already satisfied: slackclient==2.5.0 in ./venv/lib/python3.7/site-packages (from apache-superset) (2.5.0)
Requirement already satisfied: flask<2.0.0,>=1.1.0 in ./venv/lib/python3.7/site-packages (from apache-superset) (1.1.2)
Requirement already satisfied: sqlalchemy-utils<0.37,>=0.36.6 in ./venv/lib/python3.7/site-packages (from apache-superset) (0.36.8)
Requirement already satisfied: pathlib2 in ./venv/lib/python3.7/site-packages (from apache-superset) (2.3.5)
Requirement already satisfied: polyline in ./venv/lib/python3.7/site-packages (from apache-superset) (1.4.0)
Requirement already satisfied: sqlparse==0.3.0 in ./venv/lib/python3.7/site-packages (from apache-superset) (0.3.0)
Requirement already satisfied: croniter>=0.3.28 in ./venv/lib/python3.7/site-packages (from apache-superset) (0.3.36)
Requirement already satisfied: gunicorn<20.1,>=20.0.2 in ./venv/lib/python3.7/site-packages (from apache-superset) (20.0.4)
Requirement already satisfied: pandas<1.2,>=1.1.2 in ./venv/lib/python3.7/site-packages (from apache-superset) (1.1.2)
Requirement already satisfied: pyarrow<1.1,>=1.0.1 in ./venv/lib/python3.7/site-packages (from apache-superset) (1.0.1)
Requirement already satisfied: humanize in ./venv/lib/python3.7/site-packages (from apache-superset) (3.1.0)
Requirement already satisfied: backoff>=1.8.0 in ./venv/lib/python3.7/site-packages (from apache-superset) (1.10.0)
Requirement already satisfied: parsedatetime in ./venv/lib/python3.7/site-packages (from apache-superset) (2.6)
Requirement already satisfied: contextlib2 in ./venv/lib/python3.7/site-packages (from apache-superset) (0.6.0.post1)
Requirement already satisfied: markdown>=3.0 in ./venv/lib/python3.7/site-packages (from apache-superset) (3.3.3)
Requirement already satisfied: cachelib<0.2,>=0.1.1 in ./venv/lib/python3.7/site-packages (from apache-superset) (0.1.1)
Requirement already satisfied: retry>=0.9.2 in ./venv/lib/python3.7/site-packages (from apache-superset) (0.9.2)
Requirement already satisfied: wtforms-json in ./venv/lib/python3.7/site-packages (from apache-superset) (0.3.3)
Requirement already satisfied: geopy in ./venv/lib/python3.7/site-packages (from apache-superset) (2.0.0)
Requirement already satisfied: python-dateutil in ./venv/lib/python3.7/site-packages (from apache-superset) (2.8.1)
Requirement already satisfied: flask-migrate in ./venv/lib/python3.7/site-packages (from apache-superset) (2.5.3)
Requirement already satisfied: multidict<7.0,>=4.5 in ./venv/lib/python3.7/site-packages (from aiohttp>3.5.2->slackclient==2.5.0->apache-superset) (5.0.2)
Requirement already satisfied: MarkupSafe>=0.23 in ./venv/lib/python3.7/site-packages (from Jinja2>=2.10.1->flask<2.0.0,>=1.1.0->apache-superset) (1.1.1)
Requirement already satisfied: zipp>=0.5 in ./venv/lib/python3.7/site-packages (from importlib-metadata; python_version < "3.8"->markdown>=3.0->apache-superset) (3.4.0)
Requirement already satisfied: Mako in ./venv/lib/python3.7/site-packages (from alembic>=0.7->flask-migrate->apache-superset) (1.1.3)
Requirement already satisfied: python-editor>=0.3 in ./venv/lib/python3.7/site-packages (from alembic>=0.7->flask-migrate->apache-superset) (1.0.4)
Requirement already satisfied: amqp<2.7,>=2.6.0 in ./venv/lib/python3.7/site-packages (from kombu<4.7,>=4.6.10->celery!=4.4.1,<5.0.0,>=4.3.0->apache-superset) (2.6.1)
Requirement already satisfied: dnspython>=1.15.0 in ./venv/lib/python3.7/site-packages (from email-validator<2,>=1.0.5->flask-appbuilder<4.0.0,>=3.0.1->apache-superset) (2.
0.0)
Requirement already satisfied: idna>=2.0.0 in ./venv/lib/python3.7/site-packages (from email-validator<2,>=1.0.5->flask-appbuilder<4.0.0,>=3.0.1->apache-superset) (2.10)
Requirement already satisfied: Babel>=2.3 in ./venv/lib/python3.7/site-packages (from Flask-Babel<2,>=1->flask-appbuilder<4.0.0,>=3.0.1->apache-superset) (2.9.0)
Requirement already satisfied: python3-openid>=2.0 in ./venv/lib/python3.7/site-packages (from Flask-OpenID<2,>=1.2.5->flask-appbuilder<4.0.0,>=3.0.1->apache-superset) (3.2
.0)
Requirement already satisfied: pyrsistent>=0.14.0 in ./venv/lib/python3.7/site-packages (from jsonschema<4,>=3.0.1->flask-appbuilder<4.0.0,>=3.0.1->apache-superset) (0.17.3
)
Requirement already satisfied: pyparsing>=2.0.2 in ./venv/lib/python3.7/site-packages (from packaging->bleach<4.0.0,>=3.0.2->apache-superset) (2.4.7)
Requirement already satisfied: pycparser in ./venv/lib/python3.7/site-packages (from cffi!=1.11.3,>=1.8->cryptography>=2.4.2->apache-superset) (2.20)
Requirement already satisfied: defusedxml in ./venv/lib/python3.7/site-packages (from python3-openid>=2.0->Flask-OpenID<2,>=1.2.5->flask-appbuilder<4.0.0,>=3.0.1->apache-su
perset) (0.6.0)


(venv) [sparkflows@superset-machine ~]$ superset db upgrade
/home/sparkflows/venv/lib/python3.7/site-packages/pandas/compat/__init__.py:120: UserWarning: Could not import the lzma module. Your installed Python is incomplete. Attempt
ing to use lzma compression will result in a RuntimeError.
warnings.warn(msg)
logging was configured successfully
INFO:superset.utils.logging_configurator:logging was configured successfully
/home/sparkflows/venv/lib/python3.7/site-packages/flask_caching/__init__.py:192: UserWarning: Flask-Caching: CACHE_TYPE is set to null, caching is effectively disabled.
"Flask-Caching: CACHE_TYPE is set to null, "
No PIL installation found
INFO:superset.utils.screenshots:No PIL installation found
WARNI [alembic.env] SQLite Database support for metadata databases will be removed in a future version of Superset.
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.


(venv) [sparkflows@superset-machine ~]$ superset fab create-admin
/home/sparkflows/venv/lib/python3.7/site-packages/pandas/compat/__init__.p
ing to use lzma compression will result in a RuntimeError.
warnings.warn(msg)
logging was configured successfully
INFO:superset.utils.logging_configurator:logging was configured successful
/home/sparkflows/venv/lib/python3.7/site-packages/flask_caching/__init__.p
"Flask-Caching: CACHE_TYPE is set to null, "
No PIL installation found
INFO:superset.utils.screenshots:No PIL installation found
Username [admin]:
User first name [admin]:
User last name [user]:
Email [admin@fab.org]:
Password:

```
superset load_examples
```

- Create default roles and permissions:

```
superset init
```

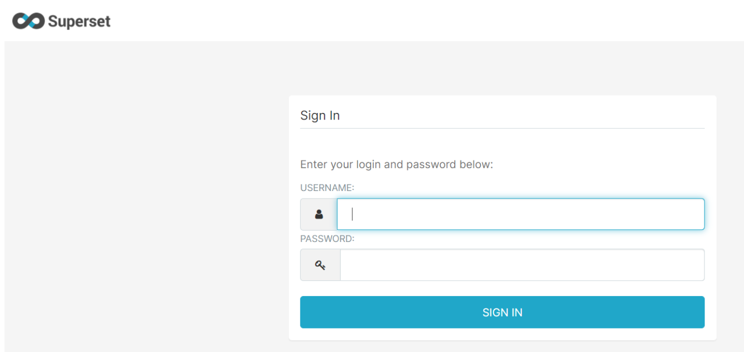
- Start a development web server on port 8088, using Gunicorn in background:

```
nohup gunicorn -b 0.0.0.0:8088 --limit-request-line 0 --limit-request-field_size␣
↪0 "superset.app:create_app()"
```

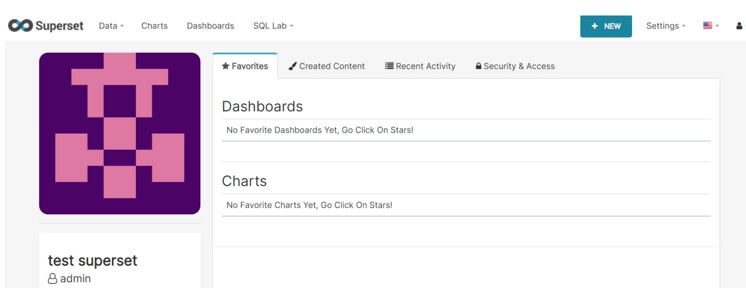
Once above command runs successfully, ensure that port 8088, on which Superset is running is accessible from your browser

- Open browser and login with public ip and port:

```
http://public-ip:8088/login
```




- Use your created credentials to login:




### 20.1.2 Connecting Superset with Databricks

Once Superset is running, you can configure Databricks database.

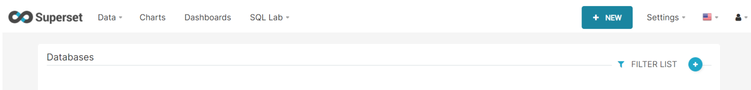
**Note:** Make sure that the Databricks cluster is up.

### Install the Python dependencies

Install Needed python dependency for Databricks on the Superset VM:

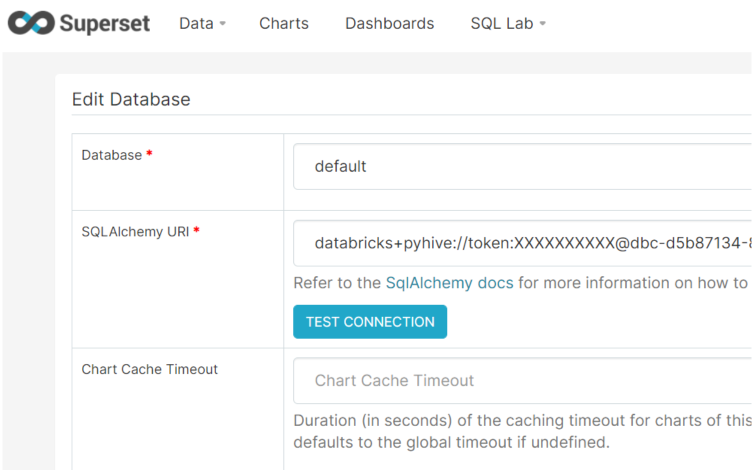
```
pip install databricks-dbapi
pip install databricks-dbapi[sqlalchemy]
```

Once the above two python databricks dependencies have been installed successfully, restart superset server & Login to Superset UI & Click on database

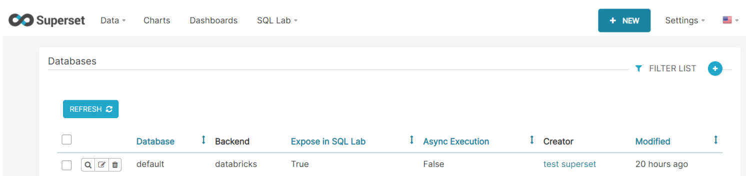



Now you can add databricks database by Clicking on NEW Tab & add Databricks *Database name & SQLAlchemy URI*:

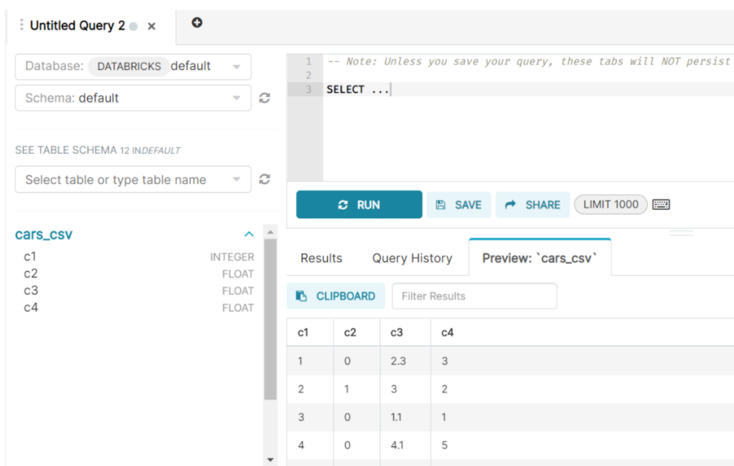
```
databricks+pyhive://token:<token>@<companyname>.cloud.databricks.com:443/<database>?
↪cluster=<cluster_id>]
```




Click on TEST CONNECTION to test your connection. It should not throw any error and SAVE it, Once the database is saved successfully, it would be available in Superset database list page.




Now You can start using databricks database tables for charts and visualizations


Untitled Query 2
Database: DATABRICKS default
Schema: default
SEE TABLE SCHEMA 12 IN DEFAULT
Select table or type table name
cars_csv
c1 INTEGER
c2 FLOAT
c3 FLOAT
c4 FLOAT
-- Note: Unless you save your query, these tabs will NOT persist i
SELECT ...
RUN
SAVE
SHARE
LIMIT 1000
Results
Query History
Preview: `cars_csv`
CLIPBOARD
Filter Results
c1 c2 c3 c4
1 0 2.3 3
2 1 3 2
3 0 1.1 1
4 0 4.1 5

# CHAPTER 21

# Python

## 21.1 Python Integration

Sparkflows supports Python in Workflows in a few ways:

- PySpark Processor

The PySpark Processor allows writing PySpark/Python code to processes the incoming DataFrame and create a new DataFrame. It can also be used to build scikit-learn models etc.

- Jython Processor

The Jython Processor allows writing Jython code to processes the incoming DataFrame and create a new DataFrame.

- Pipe Python Processor

Pipe Python Processor allows writing Python script to process the incoming DataFrame.

The incoming DataFrame is piped to the python script.

The Python script takes in each record of the DataFrame as a comma separated string. It parses the string, processes the record and writes out the new record.

### 21.1.1 PySpark Processor

Fire Insights provides a PySpark processor for writing PySpark/Python code.

#### Interface

In the PySpark Processor, we have to implement the myfn function which gets invoked:

```
def myfn(spark: SparkSession, workflowContext: WorkflowContext, id: int, inDF:␣
↪DataFrame):
```

(continues on next page)

(continued from previous page)

```
* spark : SparkSession object
* workflowContext : Can be used for outputting results to the user
* id : id of the current processor
* inDF : Input PySpark dataframe
```

### WorkflowContext

WorkflowContext provides the following methods for outputting data to the user:

```
* def outStr(self, text: str)
* def outNameValue(self, nm: str, val: str)
* def outSchema(self, id: int, title: str, df: DataFrame)
* def outDataFrame(self, id: int, title: str, df: DataFrame)
* def outPandasDataframe(self, id: int, title: str, df: pd.DataFrame)
* def outNumpy1darray(self, id: int, title: str, arr: np.ndarray)
* def outNumpy2darray(self, id: int, title: str, arr: np.ndarray)
```

### Example 1

Below is an example code for the PySpark Node.

```
from pyspark.sql.types import StringType
from pyspark.sql.functions import *
from pyspark.sql import *
from workflowcontext import *

def myfn(spark: SparkSession, workflowContext: WorkflowContext, id: int, inDF: DataFrame):
  house_type_udf = udf(lambda bedrooms: "big house" if int(bedrooms) >2 else "small house", StringType())
  filetr_df = inDF.select("id", "price", "lotsize", "bedrooms")
  outDF = filetr_df.withColumn("house_type", house_type_udf(filetr_df.bedrooms))
  return outDF
```

### Example 2

Below is another example which uses sklearn

```
from pyspark.sql.types import StringType
from pyspark.sql.functions import *
from pyspark.sql import *
from workflowcontext import *

import numpy as np
import pandas as pd

from sklearn.linear_model import LinearRegression
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import metrics

from joblib import dump, load
```

(continues on next page)

(continued from previous page)

```
def myfn(spark: SparkSession, workflowContext: WorkflowContext, id: int, inDF:␣
↪DataFrame):
  # Convert the Spark DataFrame to a Pandas DataFrame using Arrow
  dataset = inDF.select("*").toPandas()

  dataset = dataset.fillna(method='ffill')

  X = dataset[
        ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
↪'chlorides', 'free sulfur dioxide',
         'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol']].values

  y = dataset['quality'].values

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
↪state=0)

  # There are three steps to model something with sklearn
  # 1. Set up the model
  model = LinearRegression()
  # 2. Use fit
  ft = model.fit(X_train, y_train)
  print(ft)
  # 3. Check the score
  scr = model.score(X_test, y_test)
  workflowContext.outStr("Model Score : " + str(scr))

  # 4. Print model
  workflowContext.outStr("Model Coeffient : " + str(model.coef_))
  workflowContext.outStr("Model Intercept : " + str(model.intercept_))

  # 5. Predict test data
  y_pred = model.predict(X_test)

  # 6. See difference between actual and predicted value
  df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
  df1 = df.head(25)
  workflowContext.outPandasDataframe(id, "Actual - Predicted : ", df1)

  # 7. Evaluate the performance
  workflowContext.outStr("Mean Absolute Error:" + str(metrics.mean_absolute_error(y_
↪test, y_pred)))
  workflowContext.outStr("Mean Squared Error:" + str(metrics.mean_squared_error(y_
↪test, y_pred)))
  workflowContext.outStr("Root Mean Squared Error:" + str(np.sqrt(metrics.mean_
↪squared_error(y_test, y_pred))))

  return inDF
```

### 21.1.2 Jython Processor

Sparkflows has a Jython Processor.

The Jython Processor allows writing Jython code to process the incoming DataFrame. It then produces a resulting DataFrame.

In the Jython node, the following variables are available:

- **inDF** : Incoming Spark DataFrame
- **spark** : The Spark Session object

#### Example Jython Code

Below are some example Jython code which can be used.

##### Select a specific column from the DataFrame

- outDF = inDF.select("c2")

##### Count the number of records after grouping them

- outDF = inDF.groupBy("c2").count()

##### Run a SQL on the input DataFrame

The Jython Processor registers the incoming dataframe as a temporary table with a configurable name.

The below SQL in Jython script, performs a SELECT on the registered temporary table.

- outDF = spark.sql("SELECT c1, c2 FROM fire_temp_table")

##### Run a SQL followed by further grouping and count

- outDF = spark.sql("SELECT c1, c2 FROM fire_temp_table")
- outDF = outDF.groupBy("c2").count()

##### Read from HDFS and create a new DataFrame

The below Jython script, reads a JSON file from HDFS.

- outDF = spark.read().json("data/people.json")

### 21.1.3 Pipe Python Processor

Fire Insights has a Pipe Python Processor.

It pipes the incoming DataFrame through pipe to the Python Script. It also passes the Schema of the DataFrame to the Python script through the command line argument. (argv[1])

The Python script is written in the Workflow Editor.

Below is an example workflow containing Pipe Python Processor.


### Input DataFrame Schema

The schema of the incoming dataframe is also passed into the Python script as an `argument`. It can be used in the Python script as needed.

The format of the dataframe schema is below:

```
colname1:datatype1|colname2:datatype2|colname3:datatype3
```

Below is an example of printing the arguments and an example result:

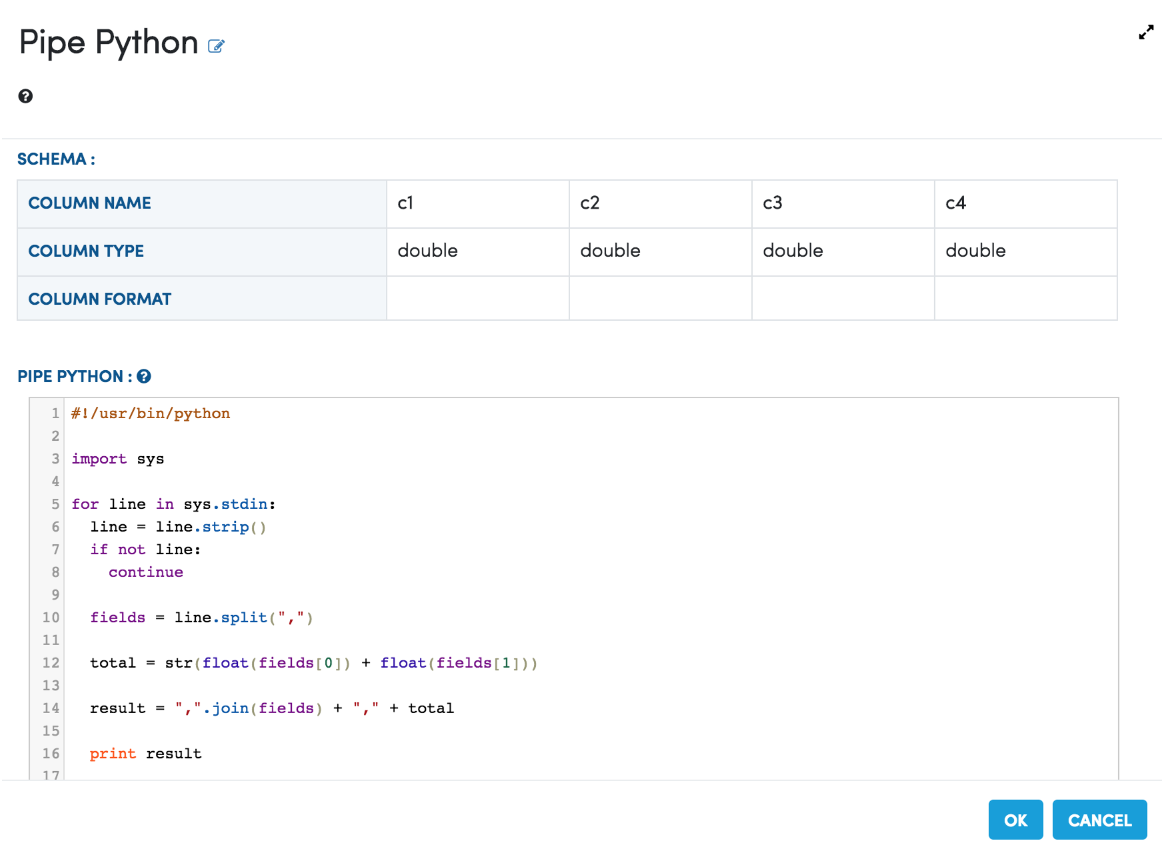
```
print "The arguments are: " , str(sys.argv)

['/tmp/fire/scripts/pipepython-1899418263068404925.py',
↪'id:DoubleType|label:DoubleType|f1:DoubleType|f2:DoubleType']
```

### Simple Example

The below example reads in the incoming records, parses them, adds a new column whose value is the sum of the first and second fields. Finally it write out the updated record back for Spark to read:

```
#!/usr/bin/python

import sys

for line in sys.stdin:
  line = line.strip()
  if not line:
    continue

  fields = line.split(",")

  total = str(float(fields[0]) + float(fields[1]))

  result = ",".join(fields) + "," + total

  print result
```

Below is the code in the Dialog box of the Pipe Python Processor of the Workflow.




### Output Schema of the Python Script

The output schema of the Python Script is used in the Spark code for recreating the Spark DataFrame from the data received from running the Python script.

It has to be specified in the Pipe Python Processor Dialog.

### Program Execution Output

Below is the output produced when executing the workflow.

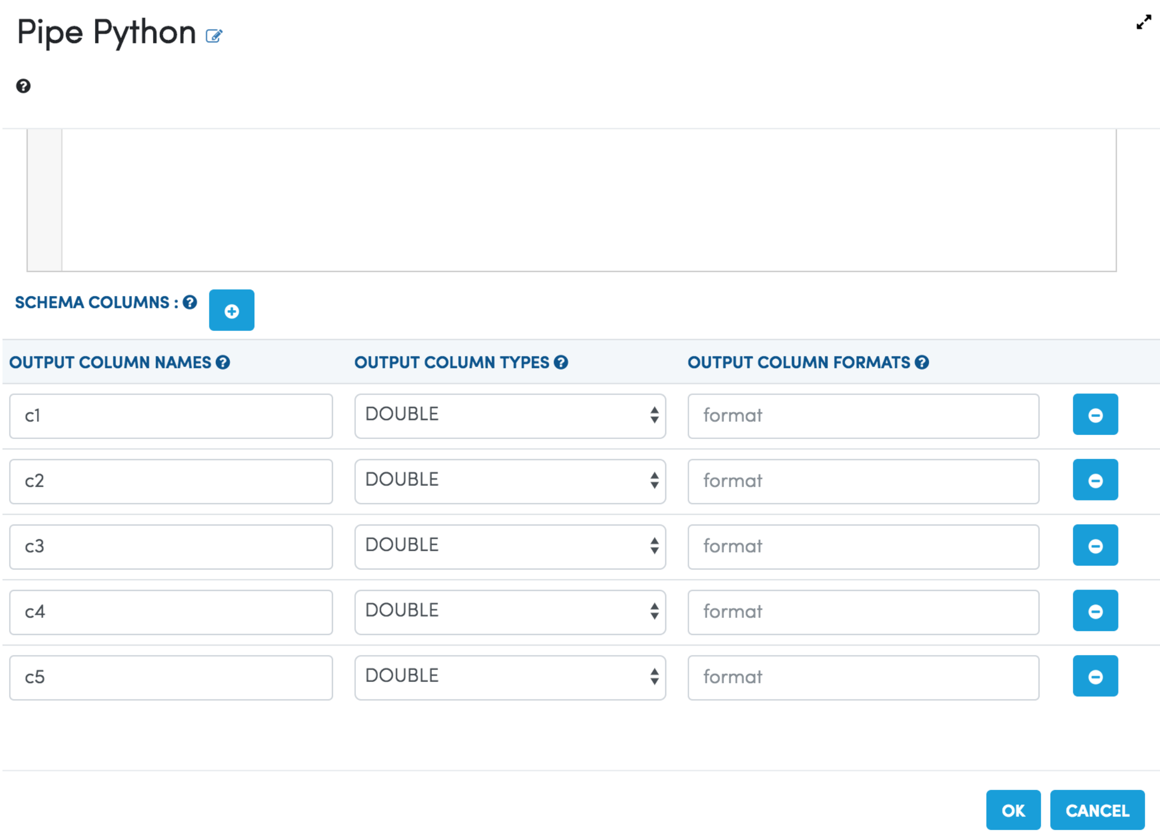
## 21.1.4 Pipe Python2 Processor

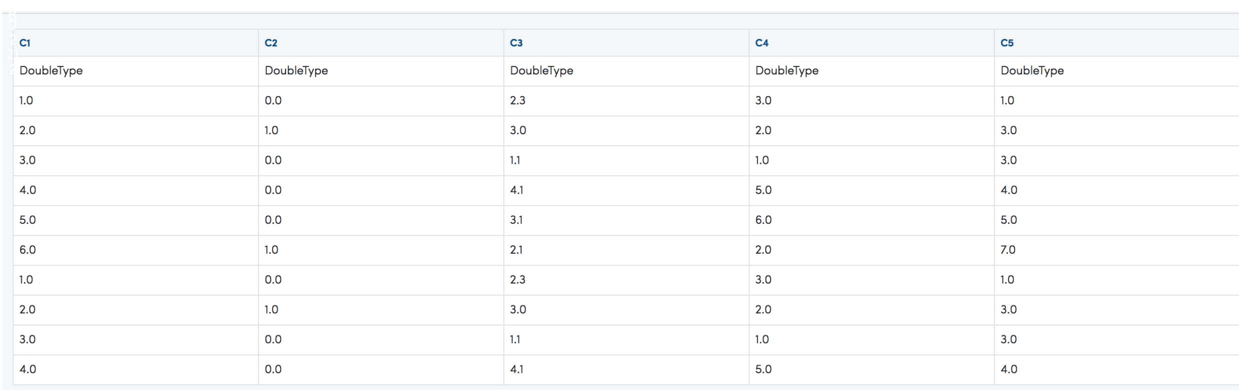
Fire Insights has a Pipe Python2 Processor.

It pipes the incoming DataFrame through pipe to the Python Script. It also passes the Schema of the DataFrame to the Python script through the command line argument. (argv[1])

The Python script is written in the Workflow Editor.

Below is an example workflow containing Pipe Python2 Processor.



| C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|
| DoubleType | DoubleType | DoubleType | DoubleType | DoubleType |
| 1.0 | 0.0 | 2.3 | 3.0 | 1.0 |
| 2.0 | 1.0 | 3.0 | 2.0 | 3.0 |
| 3.0 | 0.0 | 1.1 | 1.0 | 3.0 |
| 4.0 | 0.0 | 4.1 | 5.0 | 4.0 |
| 5.0 | 0.0 | 3.1 | 6.0 | 5.0 |
| 6.0 | 1.0 | 2.1 | 2.0 | 7.0 |
| 1.0 | 0.0 | 2.3 | 3.0 | 1.0 |
| 2.0 | 1.0 | 3.0 | 2.0 | 3.0 |
| 3.0 | 0.0 | 1.1 | 1.0 | 3.0 |
| 4.0 | 0.0 | 4.1 | 5.0 | 4.0 |

### Input DataFrame Schema

The schema of the incoming dataframe is also passed into the Python script as an `argument`. It can be used in the Python script as needed.

The format of the dataframe schema is below:

```
colname1:datatype1|colname2:datatype2|colname3:datatype3
```

Below is an example of printing the arguments and an example result:

```
print "The arguments are: " , str(sys.argv)

['/tmp/fire/scripts/pipepython-1899418263068404925.py',
↪'id:DoubleType|label:DoubleType|f1:DoubleType|f2:DoubleType']
```

### Reading in Data in Python into a Pandas DataFrame

Below is an example script which reads in the input lines and converts it to a Pandas DataFrame. It parses the schema passed in `argv[1]` to extract the column names which is used in creating the Pandas DataFrame:

```
#!/usr/bin/python

import sys
import pandas as pd

dataframe_list_of_rows = []

for line in sys.stdin:

    line = line.strip()
    if not line:
        continue

    row_list = []
    for field in line.split(","):
        row_list.append(field)

    # convert list to tuple
    row_tuple = tuple(row_list)
    dataframe_list_of_rows.append(row_tuple)


# generate column names
schema = sys.argv[1]
column_names = []
schema_columns = schema.split("|")
for column_name_with_type in schema_columns:
    column_name_with_type_split = column_name_with_type.split(":")
    column_names.append(column_name_with_type_split[0])

# create dataframe from the input rows
input_dataframe = pd.DataFrame.from_records(dataframe_list_of_rows, columns=column_
↪names)
```

### Transform the Pandas DataFrame

Now that we have the Pandas DataFrame in `input_dataframe`, we can transform it to create the result DataFrame - `output_dataframe`. In the below example, we are just setting the output dataframe to the input dataframe:

```
output_dataframe = input_dataframe
```

### Writing the Pandas DataFrame schema back to Spark

Below is an example code for writing the Pandas Schema back to Spark. It is used in inferring the schema output of the Python code. This way users do not have to reenter the schema of the output in the Workflow:

```
dataframe_dtypes = output_dataframe.dtypes

f = open(sys.argv[2],'w+')
f.write(str(dataframe_dtypes))
f.close()
```

Fire expects each line of the schema file to contain the following:

- Name of the column
- Data Type of the column

There can be multiple spaces between the name and the data type.

Fire uses the below for mapping from the data type to Spark DataFrames Data Types:

- int : integer
- float : float
- double : double
- boolean : boolean
- string : string

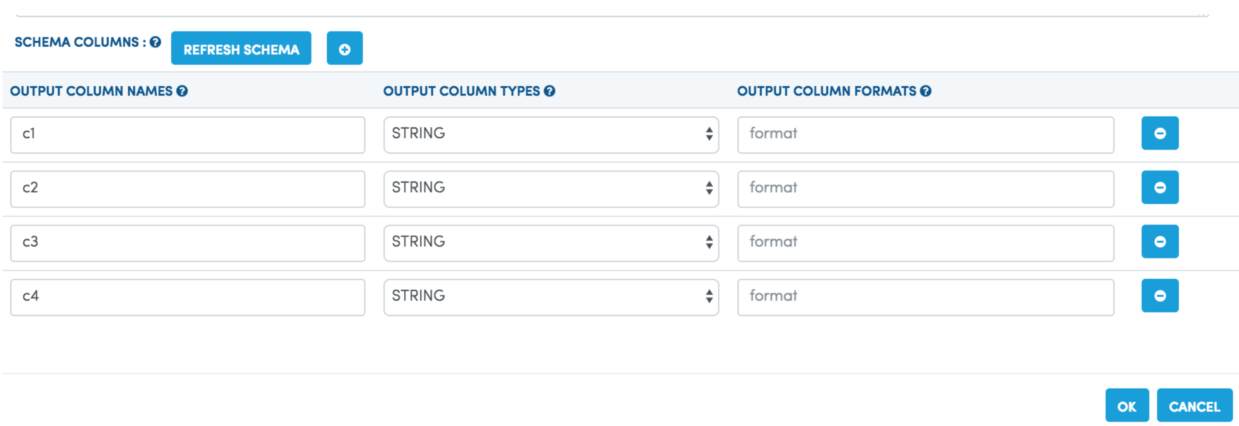
### Writing the Pandas DataFrame back to Spark

Below is an example code for writing the Pandas DataFrame back to Spark:

```
# iterate over the dataframe created and return it to the pipeNode
for index, row in output_dataframe.iterrows():
  list = row.tolist()
  row_string = ','.join(str(e) for e in list)
  print(row_string)
```

### Output Schema of the Python Script

The output schema of the Python Script is written to a file which is read by the Spark Code. Clicking on **Refresh Schema** infers the Python Schema output into Spark.


SCHEMA COLUMNS :
REFRESH SCHEMA
OUTPUT COLUMN NAMES
OUTPUT COLUMN TYPES
OUTPUT COLUMN FORMATS
c1
STRING
format
c2
STRING
format
c3
STRING
format
c4
STRING
format
OK
CANCEL

# CHAPTER 22

# Performance

## 22.1 Performance Tuning

Performance is the cornerstone of any Big Data Processing. Fire is extremely optimized for best performance. Each of the Processors are written for extreme performance, the engine is optimized for the best performance.

There are certain things which need to be taken into account for any Spark job. Fire makes it extremely easy to apply them to a Workflow.

### 22.1.1 Caching Level

Setting the right caching level of the Dataset outputs of the nodes is very important for performance in Apache Spark.

Fire allows you to set the caching output of the Dataset of any Processor.

#### When to use Caching

In general the default Caching does not have to be changed.

It is important to set Caching in the following scenarios:

- If the Dataset is going to be reused later. Below are some examples.
  - A Dataset is read from HBase. Then another dataset is read and the two are joined. In this case it is a good idea to Cache the dataset read from HBase.
  - A Dataset is joined with another Dataset. The result is then joing with another Dataset. In this case it is a good idea to cache the result of the first Join.
- A Dataset which is used in machine learning.
- Whenever a Dataset computation is expensive (JOIN etc.), caching can help in case the executor fails, the blocks are evicted from memory.

### 22.1.2 Executor Memory, vcores

When running Apache Spark jobs, we can define the number of executors, executor memory and number of vcores per executor.

Normally dynamic allocation of executors is enabled, and we do not need to specify the number of executors.

Certain jobs need higher executor memory and number of vcores. These can be specified with `--executor-memory` and `--executor-vcores`.

These additional spark configs can be specified in Fire in the Execute page. They can also be specified when the jobs are scheduled for execution.

### 22.1.3 Repartioning

Repartioning splits the datasets into the specified number of partitions.

This can help with performance

#### When saving to JDBC/File etc.

When saving a Dataset, the parallelism depends on the number of partitions of the Dataset. In case there are too few partitions, repartitioning the Dataset before saving would increase the parallelism.

Parallelism is also a double-edged sword. It is not a good idea to say have too many parallel connections to a Relational Database as it would put heavy load on the RDBMs.

### 22.1.4 Debug Mode

Fire Insights allows you to run the workflow in Debug mode.

In the debug mode a count is performed on the output from each Processor. This helps to know which Processor is exactly taking more time.

Apache Spark in general executes the DAG lazily. It starts the execution of the DAG only when it hits an Action. Hence, many times we do not know which Processor is actually taking more time.

Forcing Action with count in Debug mode forces execution of that step and insights into the time taken by the Processor.

CHAPTER 23

# Developer Guide

## 23.1 Developer Guide

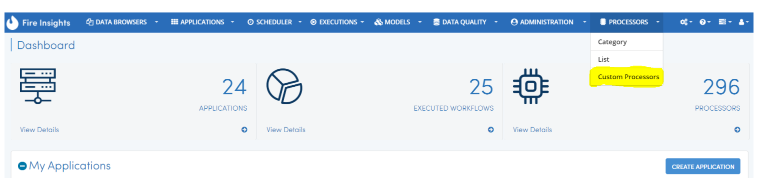
### 23.1.1 Custom Node Development in Browser

Fire Insights enables you to write custom nodes from your Browser.

You would provide the execute method for the Processor and the Schema update code. You would also provide the details of the widgets through which the user would provide the parameters for the new custom node.

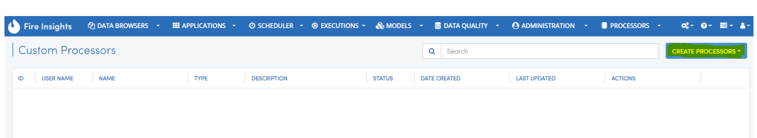
Below are the steps for creating the custom node.

Once you login to Fire Insights application, there is `PROCESSORS` menu on top, select Custom Processors.



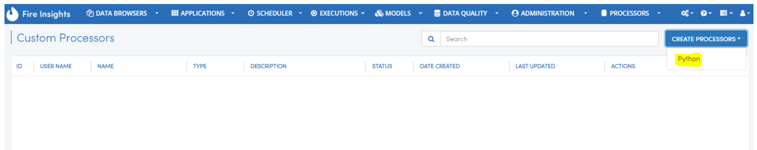

#### Click on CREATE PROCESSORS

Click on `CREATE PROCESSORS` to start creating the new processor.

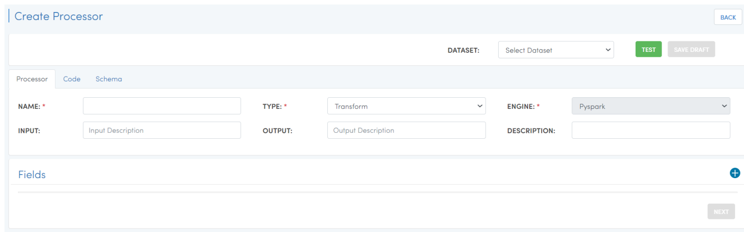



It would open up the Create Processor Page as below.

Enter the name and other details for the new processor.

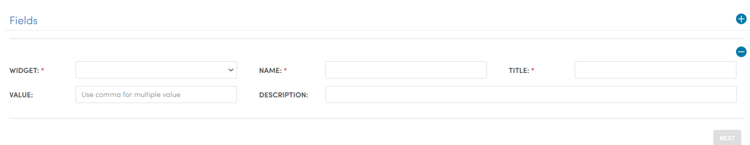



Then provide details for the various fields of the new processors. These fields would appear in the processor dialog when used in the workflow editor.




Click on the + sign to add a new field. For each field provide the following:

- WIDGET
- NAME
- TITLE
- VALUE
- DESCRIPTION




Finally click on the `Next` button to go to the Code tab.

## Execute Code

The Code tab is where you write the execution code for the new Custom Processor.

It shows the default template which you can update

Then click on `Next` button to go to the Schema tab.

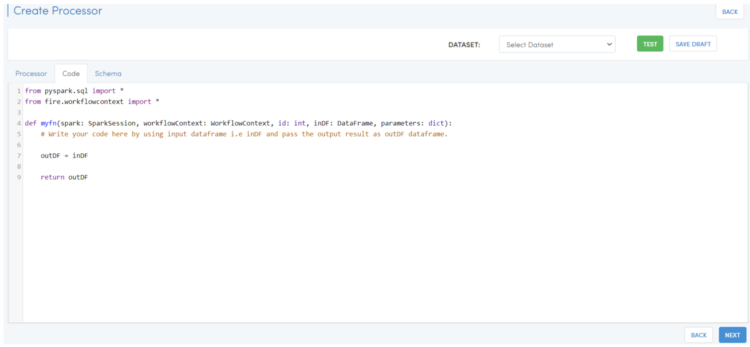
## Schema Update Code

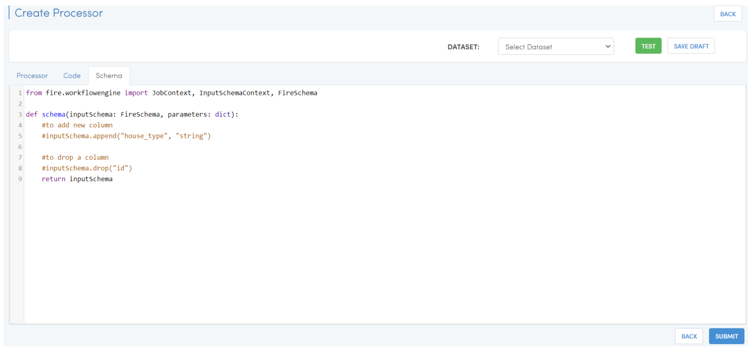
The Schema tab is where you add the code which updates the incoming schema to produce the output schema from this processor.

It displays the default template code which you can update.

Finally click on the `Submit` button to finish creating the new custom processor.

Once the custom processor is submitted successfully, it will be visible in `Custom Processors` list page.




## Testing the custom processor

Fire Insights enables you to seamlessly Test your custom processor.

When editing the custom processor, select the Dataset for the data you want to feed to the custom processor. Then click on `Test` to view the output of the new custom processor.

## Using the new Processor

The processor is now available in the Workflow Editor.

You can click on the custom processor to start using it in your workflow.

You can also export & import them

## Export Custom Processors

Fire Insights enables you to `export` Custom Processors from Browser to local machine.

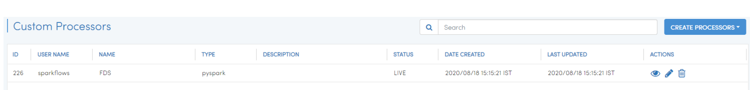
Below are the steps to export Custom Processors.

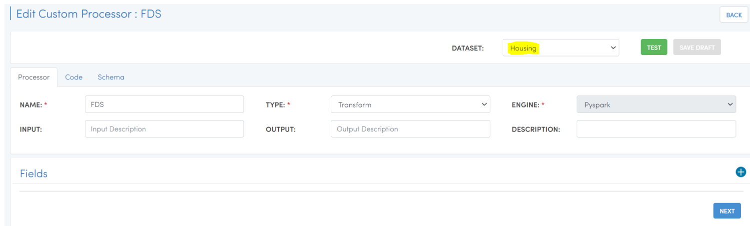
Login to Fire Insights & go to Custom Processors list page.

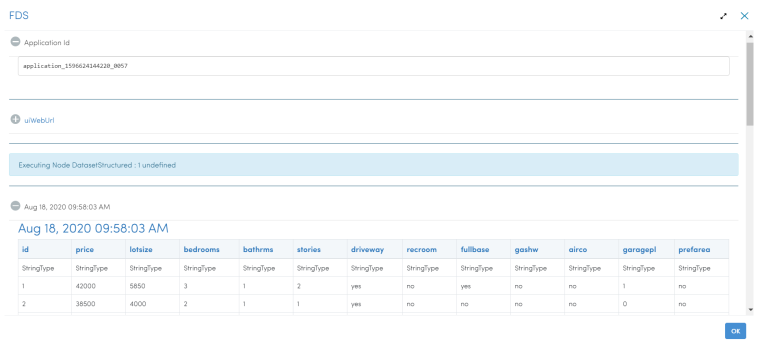
Select the `Custom Processors` which you want to export and click on export.

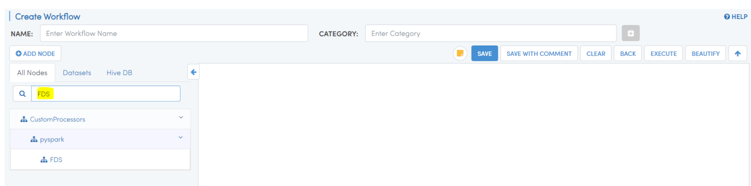
NOTE: you can export multiple `Custom Processors` at a time too.

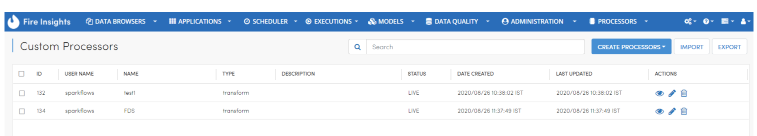
Once you click on export button, the selected Custom Processors will be downloaded to local machine in zip format.

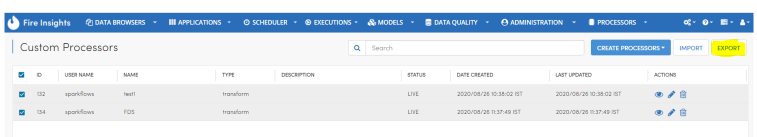


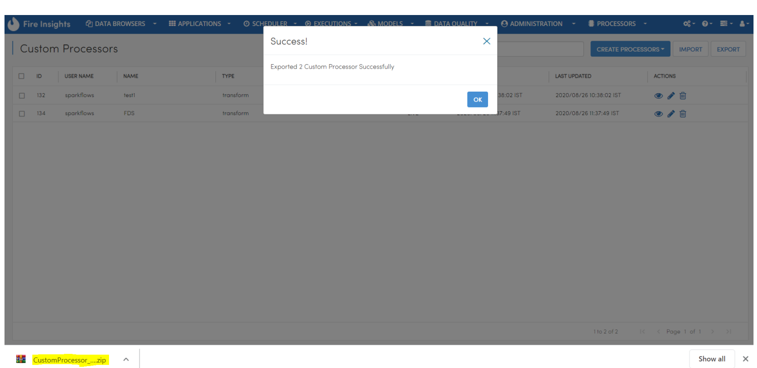

Edit Custom Processor : FDS
BACK
DATASET:
Housing
TEST
SAVE DRAFT
Processor
Code
Schema
NAME: *
FDS
TYPE: *
Transform
ENGINE: *
Pyspark
INPUT:
Input Description
OUTPUT:
Output Description
DESCRIPTION:
Fields
NEXT


FDS
Application Id
uiWebUrl
Executing Node DatasetStructured : 1 undefined
Aug 18, 2020 09:58:03 AM
OK


Create Workflow
HELP
NAME:
Enter Workflow Name
CATEGORY:
Enter Category
ADD NODE
SAVE
SAVE WITH COMMENT
CLEAR
BACK
EXECUTE
BEAUTIFY
All Nodes
Datasets
Hive DB
FDS
CustomProcessors
pyspark


Fire Insights
Custom Processors
CREATE PROCESSORS
IMPORT
EXPORT


Fire Insights
Custom Processors
CREATE PROCESSORS
IMPORT
EXPORT

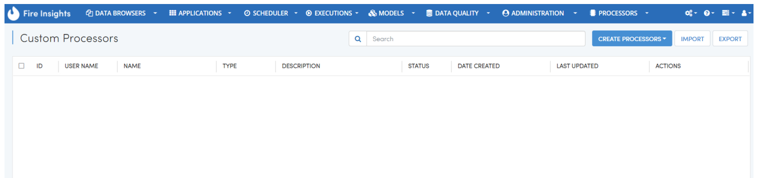

Success!
Exported 2 Custom Processor Successfully
OK
CustomProcessor_....zip
Show all

### Import Custom Processors

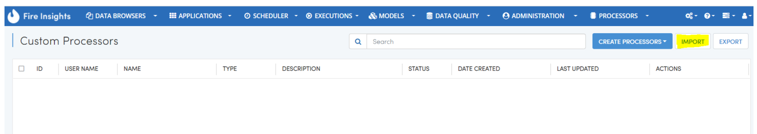
Fire Insights enables you to `import` Custom Processors to different environment.

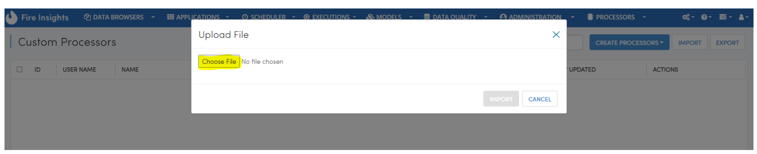
Below are the steps to Import Custom Processors.

Login to Fire Insights & go to Custom Processors list page.




Select the `IMPORT` button, it will open a new windows to upload zip file from local machine.




Once you upload zip file of Custom Processors from local machine, press `IMPORT` button to import it.

NOTE: You can import multiple `Custom Processors` at a time too.

Once you Click on IMPORT button, success message will display on imported Custom Processors.

After success import, you can view those `Custom Processors` in Custom Processors list page.

Now you can use those Custom Processors in your workflow.

## 23.1.2 Custom Node Development & Deployment (Java/Scala)

Fire Insights follows an open and extensible architecture allowing developers to add new custom nodes/processors that can be exposed in Fire UI and embedded into workflows.

**The details for building new nodes are available at the URL below:**

- https://github.com/sparkflows/writing-new-node

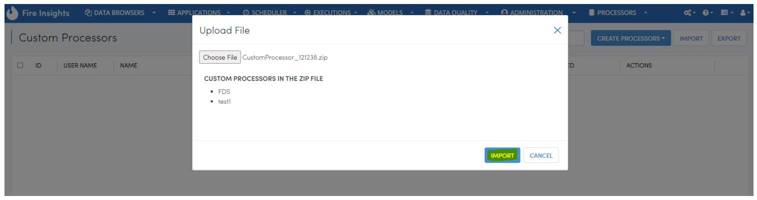
**Examples of more complex nodes are at the URL below :**

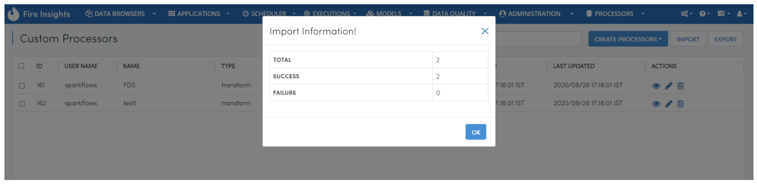
- https://github.com/sparkflows/sparkflows-stanfordcorenlp

### Step 1 : Start by cloning the github repo: writing-new-node

The easiest way to start writing a new node or processor is by cloning the `writing-new-node` repo using the command below:

- git clone https://github.com/sparkflows/writing-new-node.git




### Step 2 : Install the Fire core jar to the local maven repository

Insall the Fire core jar to your local maven repository. The pom.xml contains the dependency for it.

- mvn install:install-file -Dfile=fire-spark_2.4-core-3.1.0.jar -DgroupId=fire -DartifactId=fire-spark_2.4-core -Dversion=3.1.0 -Dpackaging=jar

### Step 3 : Code the new custom node

The custom node might be a `Dataset` node or a `Transform` node.

A `Dataset` node reads data from some source into a Dataframe. It passes on this new Dataframe to the next node. Examples of data sources include:

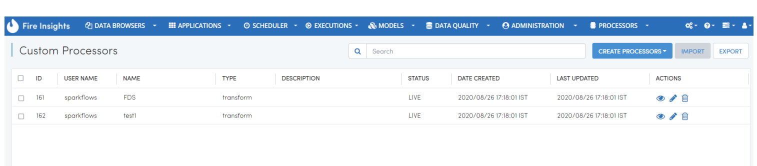
- Files on HDFS
- HIVE tables
- HBase tables
- Cassandra
- MongoDB
- Salesforce / Marketo

A `Transform` node receives an input Dataframe(s), transforms it and sends the transformed Dataframe to the next node.

### Writing a Dataset node

Create a new class that extends the `NodeDataset` class.

- Override the `execute()` method. The `execute()` method will read in data from the defined source into a Dataframe. It would then pass on the resulting DataFrame to output node(s).
- Override the `getOutputSchema()` method to return the schema of of the Dataframe created by the node.



#### Writing a Transform node

Create a new class that extends the `Node` class.

- Override the `execute()` method. The `execute()` method will `transform` the incoming DataFrame and then pass on the resulting DataFrame to output node(s).
- If the node is updating the incoming schema, also override the `getOutputSchema()` method. Otherwise the incoming schema to this node is sent to the next node(s).

#### Examples of Custom Nodes

Example of custom nodes are available at:

- https://github.com/sparkflows/writing-new-node/tree/master/src/main/java/fire/nodes/examples

### Step 4 : Create the node JSON file

Create the JSON file for the new node. The JSON file is used for displaying the new node in the `Workflow Editor` and capturing the user inputs of the various fields of the node through a `Dialog box`. The JSON for the node also captures the name of the `Java/Scala class` which has the implementation code for the Node.

Fire supports various `widgets types` for capturing the details of the fields from the user through the `Node Dialog Box`.

#### Widget Types

The details of the various widget types is available at the URL below:

- https://github.com/sparkflows/writing-new-node/blob/master/docs/README_Processor_JSON.md

#### Examples of Node JSON

- https://github.com/sparkflows/writing-new-node/blob/master/json/nodes/testprintnrows.json
- https://github.com/sparkflows/writing-new-node/blob/master/json/nodes/testmovingaverage.json

### Step 5 : Deploy the Custom Node in the Fire Server

Now that you have created a new node, follow the steps below to deploy it into the Fire Server:

- Create a jar file with `mvn clean package`
- Copy the jar file created in the previous step (target/writing-new-node-3.1.0.jar) into `fire-user-lib` directory of Fire Insights.
- Place the JSON file for the new node under the `nodes` directory.
- `Restart` the Fire Server.

The new node would be picked up by the Fire Server and be visible in the `Workflow Editor`. Check that new node is available as expected in the `Workflow Editor`.

### Use the custom node in Spark submit when running on the Spark cluster

- Select the custom node jar checkbox when executing the workflow containing the custom node.
- You can also include the custom node with `--jars <...>` when running the workflow on the cluster

## 23.1.3 Databricks Custom Node Example JSON

Custom Nodes in Fire Insights can be exported as zip files and then subsequently imported into Fire Insights.

Click on the clink below to download a custom node zip file containing scorecardpy binning custom node.

Import it into Fire Insights by going to Processors/Custom Nodes.

The code looks like below:

### Execution Code

```
from pyspark.sql import DataFrame, SparkSession
from fire.workflowcontext import WorkflowContext
import scorecardpy as sc

def myfn(spark: SparkSession, workflowContext: WorkflowContext, id: int, inDF: DataFrame, parameters: dict):
    # Write your code here by using input dataframe i.e inDF and pass the output result as outDF dataframe.

    pandas_df = inDF.toPandas()
    variables = ["purpose"]
    stopLimit = 0.1
    countDistrLimit = 0.05
    binNumLimit = 8
    method = "tree"
    positive = "bad|1"
    workflowContext.outStr(id, "Method: " + parameters['method'] + ", Positive:" + parameters['positive'])

    bins = sc.woebin(pandas_df, y="creditability", x=variables, stop_limit=float(stopLimit),
                     count_distr_limit=float(countDistrLimit),
                     bin_num_limit=int(binNumLimit), method=method, positive=positive)
    bins_ply = sc.woebin_ply(pandas_df, bins)
    spark_df = spark.createDataFrame(bins_ply)
    outDF = spark_df
    return outDF
```

### Schema Propagation Code

```
from fire.workflowengine.workflow import JobContext
from fire.workflowengine.fireschema import FireSchema

def schema(inputSchema: FireSchema, parameters: dict):
    #to add new column
    #inputSchema.append("house_type", "string")

```

(continues on next page)

(continued from previous page)

```
    #to drop a column
    #inputSchema.drop("id")
    inputSchema.append('purpose_woe', 'double')

    return inputSchema
```

### 23.1.4 Building and Running Custom Node

Fire Insights allows you to build your own Custom Nodes.

In this tutorial we would build a custom node built upon scorecardpy.

#### Install the scorecardpy dependencies

Since we are using the library scorecardpy, we would install its packages both on the Fire Insights machine and on the Databricks cluster.

Use the command below to install it on the Fire Insights machines:

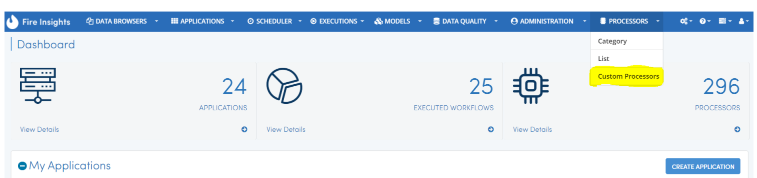
- `pip install scorecardpy`

Install it on your Databricks cluster with the below:

- Open a Notebook
- %sh pip install scorecardpy

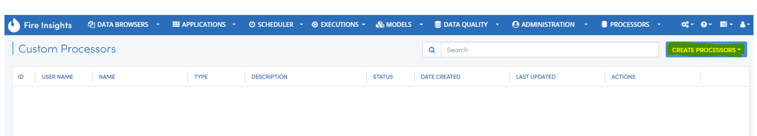
#### Go to Custom Processors

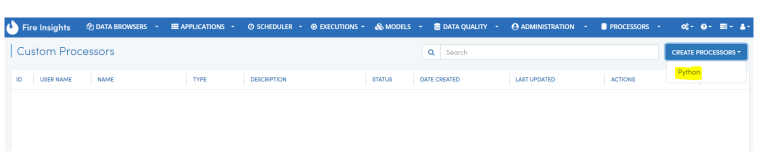
Once you login to Fire Insights application, there is `PROCESSORS` menu in top, select Custom Processors.




#### Click on CREATE PROCESSORS

Click on `CREATE PROCESSORS` to start creating the new processor.

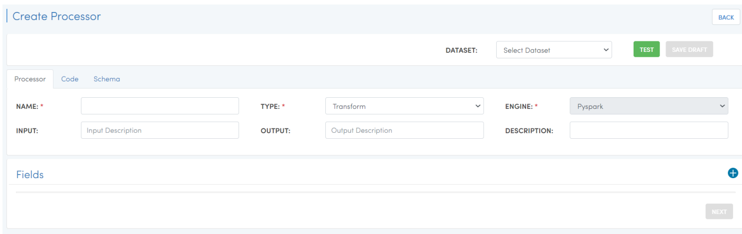


It would open up the Create Processor Page as below.

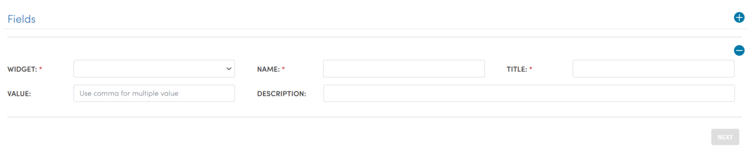
Enter the name and other details for the new processor.

Then provide details for the various fields of the new processors. These fields would appear in the processor dialog when used in the workflow editor.



Click on the + sign to add a new field. For each field provide the following:

- WIDGET
- NAME
- TITLE
- VALUE
- DESCRIPTION



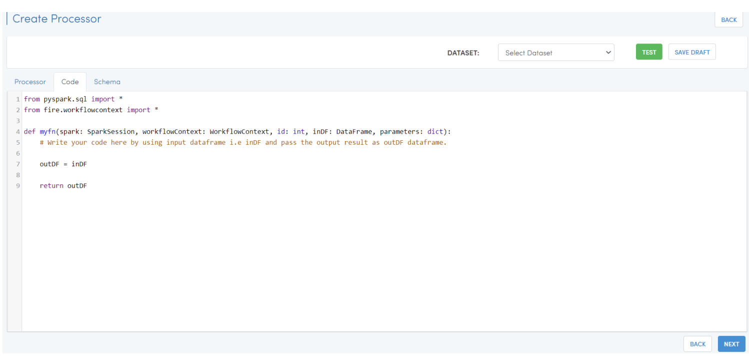
Finally click on the `Next` button to go to the Code tab.

## Execute Code

The Code tab is where you write the execution code for the new Custom Processor.

Its updated for scorecardpy here.

It shows the default template which you can update for scorecardpy.

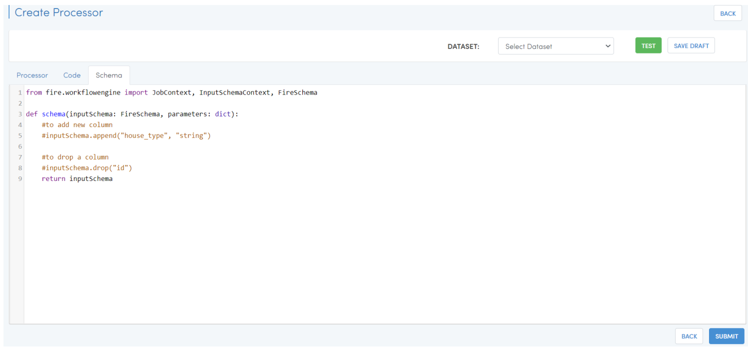


Then click on `Next` button to go to the Schema tab.

### Schema Update Code

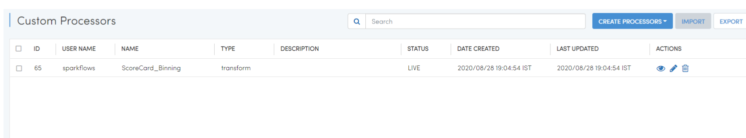
The Schema tab is where you add the code which updates the incoming schema to produce the output schema from this processor.

It displays the default template code which you can update.




Finally click on the `Submit` button to finish creating the new custom processor.

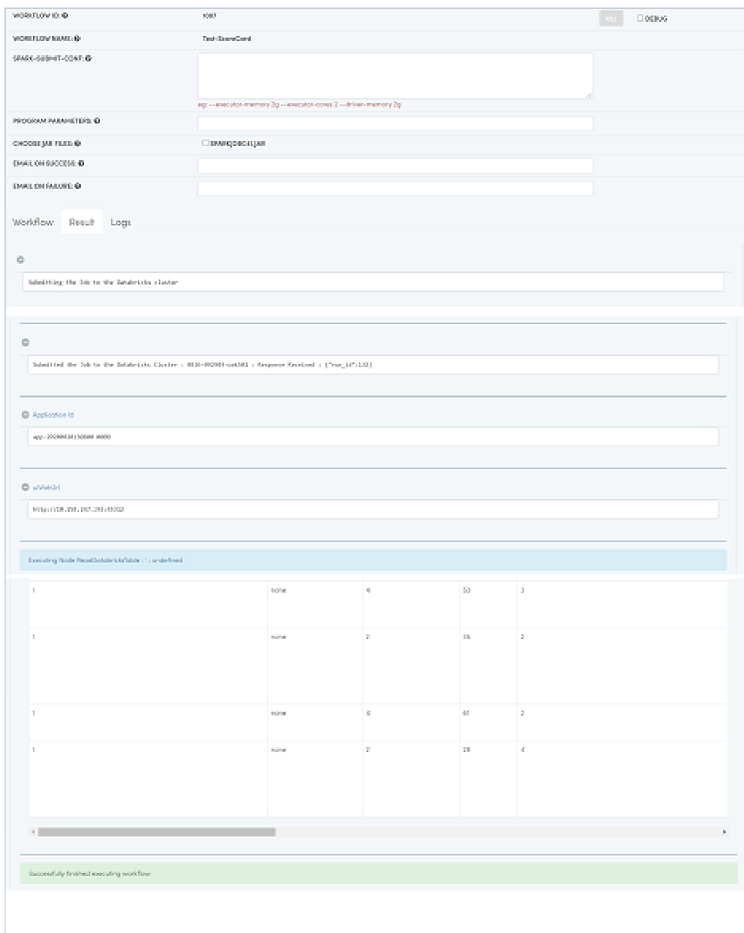
Once the custom processor submitted successfully, it will be vissible in `Custom Processors` list page.




### Using the new Processor

The processor is now available in the Workflow Editor.


You can click on the custom processor to start using it in your workflow & submit the job.


WORKFLOW ID:
DEBUG
WORKFLOW NAME:
PROGRAM PARAMETERS:
EMAIL ON SUCCESS:
EMAIL ON FAILURE:
Workflow
Result
Logs
Application Id
Successfully finished executing workflow

CHAPTER 24

# Processors

## 24.1 Processors

### 24.1.1 16-Utilities

#### 03-Execution

#### ExecuteInLoop

#### Type

transform

#### Class

fire.nodes.etl.NodeLoop

#### Fields

| Name | Title | Description |
|---|---|---|
| loopCols | Loop Columns | |

#### ReadParameters

Reads in the parameters from the given file.

### Input

Input file has records in the following form on each line : name=value

### Output

It adds the input parameters into the JobContext

### Type

shellcommand

### Class

fire.nodes.util.NodeReadParameters

### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the parameters file containing the parameter name and value in each line |

## SpecifyParameters

Provides additional parameters to the workflow. When running with spark-submit, variables can also be given on the command line with –var name=value.

### Type

doc

### Class

fire.nodes.util.NodeSpecifyParameters

### Fields

| Name | Title | Description |
|---|---|---|
| names | Parameter Names | Parameter Names |
| values | Parameter Values | Parameter Values |

#### ExecuteWorkflow

Fires the given workflow. Does not wait for the workflow to complete to resume execution

##### Type

transform

##### Class

fire.nodes.util.NodeExecuteWorkflow

##### Fields

#### 02-Data-Partition

#### Coalesce

This node coalesces the DataFrame into specified number of Partitions

##### Input

This type of node takes in a DataFrame and transforms it to another DataFrame.

##### Output

The output DataFrame has the specified number of partitions

##### Type

transform

##### Class

fire.nodes.etl.NodeCoalesce

##### Fields

| Name | Title | Description |
|---|---|---|
| numPartitions | Number of Partitions | input for number of partitions |

#### Details

This node coalesces the DataFrame into specified number of Partitions.

It is specially helpful for the case when too many small files are being created. In such a scenario, the Coalesce node can be used to limit the number of output files produced.

### Repartition

This node repartitions incoming dataframe into a specified number of partitions

#### Input

It accepts a DataFrame as input from the previous Node

#### Type

transform

#### Class

fire.nodes.etl.NodeRepartition

#### Fields

| Name | Title | Description |
|---|---|---|
| numPartitions | Number of Partitions | Number of Partitions |

### NumberOfPartitions

This node will get the number partitions in input dataframe.

#### Type

transform

#### Class

fire.nodes.util.NodeGetNumberOfPartitions

#### Fields

### 01-Spark-Performance

#### CacheDataFrame

Caches the DataFrame with the provided StorageLevel

##### Input

It takes in a DataFrame as input

##### Output

The input DataFrame is cached with the specified storage level and send to the output

##### Type

transform

##### Class

fire.nodes.util.NodeCacheDataFrame

##### Fields

| Name | Title | Description |
| --- | --- | --- |
| storageLevel | Storage Level | storage level name |

#### PrintSparkConfiguration

Print the all spark configuration used in workflow.

##### Type

transform

##### Class

fire.nodes.util.NodeSparkConfiguration

#### Fields

#### UnpersistDataFrame

Unpersists the output DataFrames of the given Nodes

#### Input

It takes in a DataFrame as input

#### Output

The outputs the incoming DataFrame

#### Type

transform

#### Class

fire.nodes.util.NodeUnpersistDataFrame

#### Fields

| Name | Title | Description |
|---|---|---|
| nodeIdsToUnpersist | Node ID to Unpersist | Output of node to unpersist |

### 24.1.2 09-DataProfiling

#### ColumnsCardinality

Distribution of categorical data. Calculates the count of records for each unique value for the column specified.

#### Type

transform

#### Class

fire.nodes.ml.NodeColumnsCardinality

#### Fields

| Name | Title | Description |
|---|---|---|
| maxValuesToDisplay | Max Values To Display | Maximum number of values to display in result. |
| inputCols | Column Names | Name of columns for the cardinality data |

### SummaryStatistics

Summary statistics provide useful information about sample data. eg: measures of spread.

#### Type

transform

#### Class

fire.nodes.ml.NodeSummary

#### Fields

#### Details

Summary statistics provides useful information about sample data. eg: measures of spread.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/mllib-statistics.html#summary-statistics

Summary Node provides a table consist of informations such as number of non-null entries (count), mean, standard deviation, and minimum and maximum value for each numerical column.

### SkewnessAndKurtosis

#### Type

transform

#### Class

fire.nodes.etl.NodeSkewnessAndKurtosis

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Column Names | Name of columns to get the skewness and kurtosis. |

### HistoGram

Computes a histogram of the data using number of bins evenly spaced between the minimum and maximum of the specific columns.

#### Type

transform

#### Class

fire.nodes.ml.NodeHistoGramCal

#### Fields

| Name | Title | Description |
| --- | --- | --- |
| inputCols | Column Name | Name of column |
| bins | Number of Bins | Number of Bins |

### FlagOutlier

Flag the outlier based on the selected column using Box-and-Whisker technique.

#### Type

transform

#### Class

fire.nodes.ml.NodeFlagOutlier

#### Fields

| Name | Title | Description |
| --- | --- | --- |
| inputCol | Input Column to flag the outlier | The Input Column to flag the outlier |
| lowerQuantile | LowerQuantile | |
| upperQuantile | UpperQuantile | |

### DistinctValuesInColumn

#### Type

transform

#### Class

fire.nodes.etl.NodeDistinctValues

#### Fields

| Name | Title | Description |
|---|---|---|
| distinctCols | Column Names | Name of columns to get the distinct combination of values. |

### NullValuesInColumn

Number of Null Values in Selected Columns.

#### Type

transform

#### Class

fire.nodes.etl.NodeNullValuesInColumn

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Column Names | Name of columns for Number of Null Values Check |

### CrossTab

Categorical V.S. Categorical

#### Type

transform

#### Class

fire.nodes.ml.NodeCrosstab

#### Fields

### GraphWeekDayDistribution

This node Finds the distribution of Week Days from Date values

#### Type

transform

#### Class

fire.nodes.graph.NodeGraphWeekDayDistribution

#### Fields

### Correlation

calculates the correlation between two series of data.

#### Input

It takes in a DataFrame and transforms it to another DataFrame

#### Output

The input DataFrame is passed along to the next Processors

#### Type

transform

#### Class

fire.nodes.ml.NodeCorrelation

#### Fields

#### Details

This node calculates the correlation between two series of data in a common operation in Statistics.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/mllib-statistics.html#correlations

#### GraphYearDistribution

This node Finds the distribution of Years from Date values

##### Type

transform

##### Class

fire.nodes.graph.NodeGraphYearDistribution

##### Fields

#### GraphMonthDistribution

This node Finds the distribution of months from Date values

##### Type

transform

##### Class

fire.nodes.graph.NodeGraphMonthDistribution

##### Fields

### 24.1.3 05-FeatureEngineering

#### WordCount

##### Type

transform

##### Class

fire.nodes.ml.NodeWordCount

##### Fields

#### MovingWindowFunctions

This node calculates the moving values of selected functions for the field(input column).

#### Input

It accepts a DataFrame as input from the previous Node

#### Output

A new columns is added which contains the results of applying the selected function on the given column of the input DataFrame

#### Type

transform

#### Class

fire.nodes.etl.NodeMovingWindowFunctions

#### Fields

| Name | Title | Description |
| --- | --- | --- |
| windowStart | Window Start | value to be used to calculate the window from |
| windowEnd | Window End | value to be used to calculate the window to |
| partitionCol | Partition Column Name | partition column to split the incoming dataframe for the sliding/window operation |
| orderCol | Order Column Name | the order of the selected column for the sliding/window operation |
| inputCols | Input Columns | input column name for calc |
| functions | Functions | |

### DateToAge

This node converts a date-column into columns of age (both in years and in days).

#### Type

transform

#### Class

fire.nodes.etl.NodeDateToAge

#### Fields

| Name | Title | Description |
|---|---|---|
| inputColName | Input Column Name | Input Column Name |
| yearsOutputColName | Years Output Column Name | Num Years Output Column Name |
| daysOutputColName | Days Output Column Name | Num Days Output Column Name |

#### Details

Calculates age from the given date or timestamp column. Age is calculated and displayed in years and days columns.

#### Examples

#### Examples when date is 06-25-2019

dd-MM-yyyy : 20-09-2018 , 0 year : 278 days MM-dd-yyyy : 09-30-2018 , 0 year : 268 days yyyy-MM-dd : 2012-01-31 , 7 year : 2702 days

## 24.1.4 01-IO

### 02-ReadStructured

#### ReadExcel

Dataset Node for reading Excel files

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetExcel

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Excel file |
| sheetName | Sheetname | Excel Sheet Name |
| header | Header | Does the file have a header row |
| outputColNames | Column Names for the Excel | New Output Columns of the SQL |
| outputColTypes | Column Types for the Excel | Data Type of the Output Columns |
| outputColFormats | Column Formats for the Excel | Format of the Output Columns |

### EmptyDataset

It creates an empty DataFrame

#### Input

It does not read any input

#### Output

It creates an empty DataFrame

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetEmpty

#### Fields

### ReadCSV

It reads in CSV files and creates a DataFrame from it

#### Input

It reads in CSV text files

### Output

It creates a DataFrame from the data read and sends it to its output

### Type

dataset

### Class

fire.nodes.dataset.NodeDatasetCSV

### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Text file/directory |
| separator | Separator | CSV Separator |
| header | Header | Does the file have a header row |
| dropMalformed | Drop Malformed | Whether to drop Malformed records or error |
| outputColNames | Column Names for the CSV | New Output Columns of the SQL |
| outputColTypes | Column Types for the CSV | Data Type of the Output Columns |
| outputColFormats | Column Formats for the CSV | Format of the Output Columns |

## ReadAvro

Dataset Node for reading Apache Avro files

### Input

It reads in Avro files

### Output

It creates a DataFrame from the data read and sends it to its output

### Type

dataset

### Class

fire.nodes.dataset.NodeDatasetAvro

### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Avro file/directory |
| outputColNames | Column Names for the Avro | Output Columns of the Avro |
| outputColTypes | Column Types for the Avro | Data Type of the Output Columns |
| outputColFormats | Column Formats for the Avro | Format of the Output Columns |

## ReadXML

It reads in XML files and creates a DataFrame from it

### Input

It reads in XML text files

### Output

It creates a DataFrame from the data read and sends it to its output

### Type

dataset

### Class

fire.nodes.dataset.NodeDatasetXML

### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Text file/directory |
| rowTag | Row Tag | Row Tag |
| outputColNames | Column Names for the CSV | New Output Columns of the SQL |
| outputColTypes | Column Types for the CSV | Data Type of the Output Columns |
| outputColFormats | Column Formats for the CSV | Format of the Output Columns |

### QueryJDBCConnection

This node executes query in Relational Databases using JDBC and creates a DataFrame from it

#### Input

It reads data from Relational Databases

#### Output

It creates a DataFrame from the data read and sends it to its output

#### Type

dataset

#### Class

fire.nodes.dataset.NodeJDBCQueryUsingConnection

#### Fields

| Name | Title | Description |
|---|---|---|
| connection | Connection | The JDBC connection to connect |
| query | Query | |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

### JDBCIncrementalLoad

This node is used to load incremental data from RDBMS to Hive.

#### Input

RDBMS detail like url, username , password, hivedb , hive table name

#### Type

dataset

### Class

fire.nodes.dataset.NodeDatasetJDBCIncrementalLoad

### Fields

| Name | Title | Description |
|---|---|---|
| sqldb | SqlDB | |
| sqlServer | SqlServer | |
| sqlUser | SqlUser | |
| password | password | |
| sqltable | SqlTable | |
| sqlkeycolumn | SqlKeyColumn | |
| homeDirectory | Config Path | |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

## DB2 JDBC

This node reads data from other databases using JDBC.

### Type

dataset

### Class

fire.nodes.dataset.NodeDatasetJDBC

### Fields

| Name | Title | Description |
|---|---|---|
| url | DB2 JDBC URL | The JDBC URL to connect to |
| user | User | User for connecting to the DB |
| password | Password | Password for connecting to the DB |
| dbtable | DB2 Table | The JDBC table that should be read. Note that anything that is valid in a FROM clause of a SQL query can be used. For example, instead of a full table you could also use a subquery in parentheses. |
| driver | DB2 Driver | The class name of the JDBC driver needed to connect to this URL |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

## ReadParquet

Dataset Node for reading Apache Parquet files

### Input

It reads in Parquet files

### Output

It creates a DataFrame from the data read and sends it to its output

### Type

dataset

### Class

fire.nodes.dataset.NodeDatasetParquet

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Parquet file/directory |
| outputColNames | Column Names for the Parquet | Output Columns of the Parquet |
| outputColTypes | Column Types for the Parquet | Data Type of the Output Columns |
| outputColFormats | Column Formats for the Parquet | Format of the Output Columns |

### ReadDatabricksTable

This node reads data from Relational Databases using JDBC and creates a DataFrame from it

#### Input

It reads data from Relational Databases

#### Output

It creates a DataFrame from the data read and sends it to its output

#### Type

dataset

#### Class

fire.nodes.dataset.NodeReadDatabricksTable

#### Fields

| Name | Title | Description |
|---|---|---|
| db | Databricks Database | Databricks Database |
| table | Databricks Table | Databricks Table from which to read the data |
| driver | Driver | The class name of the JDBC driver needed to connect to this URL |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

### JDBCConnection

This node reads data from Relational Databases using JDBC and creates a DataFrame from it

#### Input

It reads data from Relational Databases

#### Output

It creates a DataFrame from the data read and sends it to its output

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetJDBCUsingConnection

#### Fields

| Name | Title | Description |
|---|---|---|
| connection | Connection | The JDBC connection to connect |
| dbtable | DB Table | The JDBC table that should be read. Note that anything that is valid in a FROM clause of a SQL query can be used. For example, instead of a full table you could also use a subquery in parentheses. |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

### CreateDataset

Creates a dataset with the specified number of Rows and 9 pre-defined columns

#### Input

It does not read data from any external source

#### Output

It creates a DataFrame with the specified number of Rows

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetCreate

#### Fields

| Name | Title | Description |
|---|---|---|
| numRows | Number of Rows | Number of Rows in the Output Dataset |

### DatasetStructured

This Node creates a DataFrame by reading data from HDFS, HIVE etc. The dataset has been defined earlier in Fire by using the Dataset Feature. As a user, you just have to select the Dataset of your interest.

#### Input

It reads in data from HIVE or files HDFS

#### Output

It creates a DataFrame from the input data and sends it to its output.

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetStructured

#### Fields

| Name | Title | Description |
|---|---|---|
| dataset | Dataset | Selected Dataset |

### Details

This Node creates a DataFrame by reading data from HDFS, HIVE etc.

The data has been defined earlier in Fire by using the Dataset Feature. As a user, you just have to select the Dataset of your interest.

### ReadJDBC

This node reads data from Relational Databases using JDBC and creates a DataFrame from it

### Input

It reads data from Relational Databases

### Output

It creates a DataFrame from the data read and sends it to its output

### Type

dataset

### Class

fire.nodes.dataset.NodeDatasetJDBC

### Fields

| Name | Title | Description |
| --- | --- | --- |
| url | URL | The JDBC URL to connect to |
| user | User | User for connecting to the DB |
| password | Password | Password for connecting to the DB |
| dbtable | DB Table | The JDBC table that should be read. Note that anything that is valid in a FROM clause of a SQL query can be used. For example, instead of a full table you could also use a subquery in parentheses. |
| driver | Driver | The class name of the JDBC driver needed to connect to this URL |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

### ReadHanaCsv

It reads in Hana CSV files and creates a DataFrame from it

#### Input

It reads in CSV text files and sql file to create schema from it

#### Output

It creates a DataFrame from the data read and sends it to its output

#### Type

dataset

#### Class

fire.nodes.dataset.NodeReadHANACSVDump

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Text file/directory |
| hdfsSqlFile | SQL File | Path of the sql file that contains create table script. |
| outputColNames | Column Names for the CSV | New Output Columns of the SQL |
| outputColTypes | Column Types for the CSV | Data Type of the Output Columns |
| outputColFormats | Column Formats for the CSV | Format of the Output Columns |

### URLSingleRecordJSONReader

It reads in single record JSON from the given URL and creates a DataFrame from it

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetURLSingleRecordJsonReader

### Fields

| Name | Title | Description |
|---|---|---|
| URL | URL | URL from where to read the JSON string from |
| outputColNames | Column Names | Column Names |
| outputColTypes | Column Types | Data Types |
| outputColFormats | Column Formats | Formats |

### ReadLibsvm

It reads in Libsvm files and creates a DataFrame from it

### Input

It reads in Libsvm text files

### Output

It creates a DataFrame from the data read and sends it to its output

### Type

dataset

### Class

fire.nodes.dataset.NodeDatasetLibsvm

### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Text file/directory |
| numFeatures | NumFeatures | Number of features in feature column |
| outputColNames | Column Names for the CSV | New Output Columns of the SQL |
| outputColTypes | Column Types for the CSV | Data Type of the Output Columns |
| outputColFormats | Column Formats for the CSV | Format of the Output Columns |

### ReadJSON

Dataset Node for reading JSON files

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetJSON

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the JSON file/directory |
| multiLine | Multi Line | |
| outputColNames | Column Name | New Output Column Name |
| outputColTypes | Column Type | Data Type of the Output Column |
| outputColFormats | Column Format | Format of the Output Column |

#### Details

It reads in JSON files. Each JSON record has to be on a separate line for Spark to handle it correctly.

There cannot be line break within a record.

### URLTextFileReader

Reads text file from the given URL and creates a DataFrame from it. Each line in the file is a record in the DataFrame.

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetUrlTextFileReader

#### Fields

| Name | Title | Description |
|---|---|---|
| url | URL | URL of the file |

### ReadShapeFile

It reads in Shape files and creates a DataFrame from it

#### Input

It reads in Shape files

#### Output

It creates a DataFrame from the data read and sends it to its output

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetShapeFile

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the input directory |

### 03-ReadUnstructured

#### TextFiles

Reads in Text Files from a given path and loads each line as a separate Row

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetTextFiles

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Text file/directory |
| outputCol | Output Column Name | Text Lines Column in the Output DataFrame |

### WholeTextFiles

Reads in Whole Text Files directory from a given path and loads each files as a separate Row with key(file name and values(file content)

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetWholeTextFiles

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Text files directory |

### Tika

Reads in files from a given path and parses them with Apache Tika

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetTika

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the file/directory |
| fileNameCol | File Name Column | File Name Column in the Output DataFrame |
| contentCol | Content Column | Tika output Column in the Output DataFrame |

### PDF

Reads in PDF Files from a given path and extracts the text content from them

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetPDF

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the PDF file/directory |
| fileNameCol | File Name | File Name Column in the Output DataFrame |
| contentCol | File Content | File Content Column in the Output DataFrame |

### PDFImageOCR

Reads in PDF Files from a given path, extracts the images from them and converts them to text with Tesseract

#### Input

It reads in a PDF file or a directory containing PDF files

#### Output

It creates a DataFrame from the data read and sends it to its output

#### Type

dataset

#### Class

fire.nodes.dataset.NodeDatasetPDFImageOCR

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path of the PDF files | Path of the PDF file/directory |
| fileNameCol | File Name Column | File Name Column in the Output DataFrame |
| outputCol | Column Name which contains the result of OCR | OCR output Column in the Output DataFrame |

#### BinaryFiles

Reads in Binary Files from a given path and loads them as FileName/Content

##### Type

dataset

##### Class

fire.nodes.dataset.NodeDatasetBinaryFiles

##### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Binary file/directory |
| fileNameCol | File Name Column | File Name Column in the Output DataFrame |
| binaryContentCol | Binary File Content Column | Binary File Content Column in the Output DataFrame |

##### Details

It creates a new Dataframe from some data. Data can be in binary, text, parquet, pdf, image files.

### 03-Save

#### SaveJDBC

This node writes data to databases using JDBC.

##### Type

transform

##### Class

fire.nodes.save.NodeSaveJDBC

#### Fields

| Name | Title | Description |
|---|---|---|
| url | URL | The JDBC URL to connect to |
| table | DB Table | The JDBC table to write to |
| driver | Driver | The class name of the JDBC driver needed to connect to the URL |
| user | User | Username with which to connect to the DB |
| password | Password | Password with which to connect to the DB |
| truncate | Truncate | Whether to truncate the table in case Save Mode is Overwrite |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the table Exists |

### UpsertJDBC

This node insert or update the data to databases using JDBC.

#### Type

transform

#### Class

fire.nodes.save.NodeUpsertJDBC

#### Fields

| Name | Title | Description |
|---|---|---|
| primaryKeyColumn | PrimaryKeyColumn | Key column name in table |
| url | URL | The JDBC URL to connect to |
| table | DB Table | The JDBC table to write to |
| driver | Driver | The class name of the JDBC driver needed to connect to the URL |
| user | User | Username with which to connect to the DB |
| password | Password | Password with which to connect to the DB |

### SaveCSV

Saves the DataFrame into the specified location in CSV Format

#### Type

transform

#### Class

fire.nodes.save.NodeSaveCSV

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path where to save the CSV files |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |
| header | Header | Should a Header Row be saved with each File? |
| partitionColNames | Partition Column Names | Partition Column Names |

### SaveJSON

Saves the DataFrame into the specified location in JSON Format

#### Type

transform

#### Class

fire.nodes.save.NodeSaveJSON

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path where to save the JSON files |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |
| partitionColNames | Partition Column Names | Partition Column Names |

### KafkaProducer

Write out the Dataframe to a specified Apache Kafka Topic

#### Type

transform

#### Class

fire.nodes.save.NodeKafkaProducer

#### Fields

| Name | Title | Description |
|---|---|---|
| brokers | Kafka Brokers | Brokers |
| topic | Topic | Kafka Topic to write out the incoming Dataframe to |

### SaveParquet

Saves the DataFrame into the specified location in Parquet Format. When running on Hadoop, it is saved onto HDFS.

#### Type

transform

#### Class

fire.nodes.save.NodeSaveParquet

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path where to save the Parquet files |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |
| partitionColNames | Partition Column Names | Partition Column Names |

### SaveORC

Saves the DataFrame into the specified location in ORC Format

#### Type

transform

#### Class

fire.nodes.save.NodeSaveORC

### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path where to save the ORC files |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |

## InsertIntoHIVETable

Saves the DataFrame into an Apache HIVE Table

### Type

transform

### Class

fire.nodes.save.NodeInsertIntoTable

### Fields

| Name | Title | Description |
|---|---|---|
| database | HIVE Database | Name of the HIVE Database |
| table | HIVE Table | Name of the HIVE table |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |
| partitionBy | Partition By | Partition By Column (can be empty) |
| bucketBy | Bucket By | Bucket By Column (can be empty) |

### Details

When using Insert Into Table, the HIVE table has to already exist.

Otherwise it throws the following exception:

org.apache.spark.sql.catalyst.analysis.NoSuchTableException: Table or view 'xyz' not found in database 'abc';

## SaveAsHIVETable

Saves the DataFrame into an Apache HIVE Table

### Type

transform

### Class

fire.nodes.save.NodeSaveAsTable

### Fields

| Name | Title | Description |
|---|---|---|
| database | HIVE Database | Name of the HIVE Database |
| table | HIVE Table | Name of the HIVE table |
| partitionBy | Partition By | List of columns to partition by - separated by space |
| format | Format | File format when saving to HIVE Table |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |

### Details

If the HIVE table does not exist, it would create the table.

### SaveAvro

Saves the DataFrame into the specified location in Apache Avro Format

### Type

transform

### Class

fire.nodes.save.NodeSaveAvro

### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path where to save the Avro files |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |

### 01-Connectors

### Salesforce

This node reads data from Salesforce.

### Type

dataset

### Class

fire.nodes.salesforce.NodeReadSalesforce

### Fields

| Name | Title | Description |
|---|---|---|
| sql | SQL | Sql for reading salesforce data ex - select id, name, amount from opportunity |
| userNmae | User Name | UserName of Salesforce |
| password | Password | Password of Salesforce |
| readOption | Read Option | Pulling data/Object from salesforce |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

## ReadMarketo

Node for reading Marketo files

### Type

dataset

### Class

fire.nodes.marketo.NodeReadMarketo

### Fields

| Name | Title | Description |
|---|---|---|
| clientId | Client Id | Marketo account clientId |
| clientSecret | Client Secret | Marketo account clientSecret |
| instanceUrl | Instance Url | Instance URL to be used to access Marketo. It has to be specified without /rest. i.e it should be like https://119-AAA-888.mktorest.com |
| object | Object | Object to be queried from Marketo. ex. leads |
| filterType | Filter Type | Filter field to be used |
| filterValues | Filter Values | Comma separated filter values to be applied |
| fromDate | From Date | (Optional) Datatime from which the data has to be fetched. It has to be in ISO 8601 format |
| customObject | Custom Object | (Optional) Boolean to specify if the specified object is custom object, Default value is false |
| apiVersion | Api Version | (Optional) API Version to be used. Default value is v1 |
| modifiedFields | Modified Fields | (Optional) Fields to be considered for leadChanges. It has to be comma separated field names |
| queryType | Query Type | Query Type of Marketo |
| outputColNames | Column Names for the Marketo | New Output Columns of the SQL |
| outputColTypes | Column Types for the Marketo | Data Type of the Output Columns |
| outputColFormats | Column Formats for the Marketo | Format of the Output Columns |

## SaveRedshift-AWS

This node save data to Redshift using JDBC.

### Type

transform

### Class

fire.nodes.aws.NodeSaveRedshift

#### Fields

| Name | Title | Description |
|---|---|---|
| url | URL | The JDBC URL to connect to |
| dbtable | Redshift Table | The Redshift table that should be write. Note that anything that is valid in a FROM clause of a SQL query can be used. For example, instead of a full table you could also use a subquery in parentheses. |
| awsAccessKeyId | AWS Access Key Id | AWS Access Key Id |
| awsSecretAccessKey | AWS Secret Access Key | AWS Secret Access Key |
| tempS3Dir | Temporary S3 directory | Temporary S3 directory |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |

### WriteToSnowFlake

#### Type

transform

#### Class

fire.nodes.snowflake.NodeWriteToSnowFlake

#### Fields

| Name | Title | Description |
|---|---|---|
| sfUrl | SF Url | SnowFlake URL to connect to |
| sfUser | SF User | User for connecting to the SnowFlake |
| sfPassword | SF Password | Password for connecting to the SnowFlake |
| sfDatabase | SF Database | Database for connecting to the SnowFlake |
| sfSchema | SF Schema | Schema for connecting to the SnowFlake |
| sfWarehouse | SF Warehouse | Warehouse for connecting to the SnowFlake |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the table Exists |
| dbtable | SF Table | |

### SaveCassandra

Saves the rows of the incoming DataFrame into Apache Cassandra

### Type

transform

### Class

fire.nodes.cassandra.NodeSaveCassandra

### Fields

| Name | Title | Description |
|---|---|---|
| table | Cassandra Table Name | Cassandra Table into which data gets loaded |
| keyspace | Cassandra Keyspace Name | The keyspace where table is looked for |
| host | Host | |
| username | Username | |
| password | Password | |

## ExecuteQueryInSnowFlake

### Type

dataset

### Class

fire.nodes.snowflake.NodeExecuteQueryInSnowFlake

### Fields

| Name | Title | Description |
|---|---|---|
| sfUrl | SF Url | SnowFlake URL to connect to |
| sfUser | SF User | User for connecting to the SnowFlake |
| sfPassword | SF Password | Password for connecting to the SnowFlake |
| sfDatabase | SF Database | Database for connecting to the SnowFlake |
| sfSchema | SF Schema | Schema for connecting to the SnowFlake |
| sfWarehouse | SF Warehouse | Warehouse for connecting to the SnowFlake |
| query | SF Query | |
| outputColNames | Output Column Names | Name of the Output Columns |
| outputColTypes | Output Column Types | Data Type of the Output Columns |
| outputColFormats | Output Column Formats | Format of the Output Columns |

### ReadMongoDB

Reads data from MongoDB

#### Type

dataset

#### Class

fire.nodes.mongodb.NodeReadMongoDB

#### Fields

| Name | Title | Description |
|---|---|---|
| mongoURI | MongoDB URI | URI of MongoDB to read from |
| mongoDBName | MongoDB Database | Name of the MongoDB database to read from |
| mongoTableName | MongoDB Table | Name of the MongoDB table to read from |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

### SaveMongoDB

It Saves the incoming Dataframe into MongoDB

#### Input

It takes in a DataFrame as input

#### Output

Incoming dataFrame is passed along to the next nodes.

#### Type

transform

#### Class

fire.nodes.mongodb.NodeSaveMongoDB

#### Fields

| Name | Title | Description |
|---|---|---|
| mongoURI | mongo URI | URI of mongodb |
| mongoDBName | mongoDB Name | mongoDB Name |
| mongoTableName | mongo Table Name | mongo Table Name |

### ReadDatabricksTable

This node reads a table from Databricks

#### Input

It reads data from Databricks Table

#### Output

It creates a DataFrame from the data read and sends it to its output

#### Type

dataset

#### Class

fire.nodes.databricks.NodeReadDatabricksTable

#### Fields

### ReadHIVETable

This node reads data from Apache HIVE table and creates a DataFrame from it

#### Input

It reads in CSV text files

#### Output

It creates a DataFrame from the data read and sends it to its output

#### Type

dataset

#### Class

fire.nodes.hive.NodeHIVE

#### Fields

### SaveHBase

Saves all the rows in the incoming DataFrame onto Apache HBase using the specific field mapping

#### Input

It takes in a DataFrame as input

#### Output

Incoming dataFrame is passed along to the next nodes.

#### Type

transform

#### Class

fire.nodes.hbase.NodeSaveHBase

#### Fields

#### Details

SaveHBase node saves all the rows in the incoming DataFrame onto HBase using the specific field mapping.

The DataFrame columns which do not have to be loaded into HBase are left empty.

### SaveElasticSearch

Stores the rows of the incoming DataFrame into Elastic Search

#### Type

transform

#### Class

fire.nodes.elasticsearch.NodeSaveElasticSearch

### Fields

| Name | Title | Description |
|---|---|---|
| indexName | Index Name | Name of the Elastic Search Index |
| elasticSearchHost | Elastic Search Host | Name of the Elastic Search Host |
| elasticSearchPort | Elastic Search Port | Port of Elastic Search |
| esIndexAutoCreate | es.index.auto.create | ES Index Auto Create |
| esNodesWANOnly | es.nodes.wan.only | ES Nodes WAN Only |
| esNodesIngestOnly | es.nodes.ingest.only | ES Nodes Ingest Only |
| esNodesDataOnly | es.nodes.data.only | ES Nodes Data Only |
| esNetHttpAuthUser | es.net.http.auth.user | Username |
| esNetHttpAuthPass | es.net.http.auth.pass | Password |
| esConfKeys | Config Key/Value Pairs | More Config Values |
| esConfValues | Config Key/Value Pairs | More Config Values |

### ReadFromSnowFlake

### Type

dataset

### Class

fire.nodes.snowflake.NodeReadFromSnowFlake

### Fields

| Name | Title | Description |
|---|---|---|
| sfUrl | SF Url | SnowFlake URL to connect to |
| sfUser | SF User | User for connecting to the SnowFlake |
| sfPassword | SF Password | Password for connecting to the SnowFlake |
| sfDatabase | SF Database | Database for connecting to the SnowFlake |
| sfSchema | SF Schema | Schema for connecting to the SnowFlake |
| sfWarehouse | SF Warehouse | Warehouse for connecting to the SnowFlake |
| dbtable | SF Table | |
| outputColNames | Output Column Names | Name of the Output Columns |
| outputColTypes | Output Column Types | Data Type of the Output Columns |
| outputColFormats | Output Column Formats | Format of the Output Columns |

### SFTP

Secure file transfer protocol

### Type

dataset

### Class

fire.nodes.sftp.NodeSftp

### Fields

| Name | Title | Description |
|---|---|---|
| sftpHost | Sftp Host | IP address of sftp |
| sftPort | Sft Port | Port no of SFTP. Default port is 22 |
| sftpUser | Sftp User Name | SFTP User Name |
| sftpPass | Sftp Password | SFTP User Password |
| sftpUserDir | Sftp User Directory | user directory path(File take from) |
| sftpDirectory | Sftp Directory | server directory path(Inside SFTP uploads folder '/uploads') |
| pemKey | Pem Key | Path of pem key directory |

## ReadCassandra

This node reads data from Apache Cassandra

### Type

dataset

### Class

fire.nodes.cassandra.NodeReadCassandra

### Fields

| Name | Title | Description |
|---|---|---|
| table | Cassandra Table | Cassandra Table from which to read the data |
| keyspace | Cassandra Keyspace | Cassandra Keyspace |
| host | Cassandra host | |
| username | Username | |
| password | Password | |

## SaveDatabricksTable

This node saves a input data as table in Databricks

#### Input

It take dataframe as input data.

#### Output

It creates a Table in Databricks from the dataframe(input data).

#### Type

transform

#### Class

fire.nodes.databricks.NodeSaveDatabricksTable

#### Fields

| Name | Title | Description |
|---|---|---|
| database | Databricks Database | Name of the Database |
| table | Databricks Table | Name of the table |
| partitionBy | Partition By | List of columns to partition by - separated by space |
| format | Format | File format when saving to Table |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |

### ReadRedshift-AWS

This node reads data from Redshift using JDBC.

#### Type

dataset

#### Class

fire.nodes.aws.NodeReadRedshift

### Fields

| Name | Title | Description |
|---|---|---|
| url | URL | The JDBC URL to connect to |
| dbtable | Redshift Table | The Redshift table that should be read. Note that anything that is valid in a FROM clause of a SQL query can be used. For example, instead of a full table you could also use a subquery in parentheses. |
| awsAccessKeyId | AWS Access Key Id | AWS Access Key Id |
| awsSecretAccessKey | AWS Secret Access Key | AWS Secret Access Key |
| tempS3Dir | Temporary S3 directory | Temporary S3 directory |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

## ReadElasticSearch

Reads data from Elastic Search

### Type

dataset

### Class

fire.nodes.elasticsearch.NodeReadElasticSearch

#### Fields

| Name | Title | Description |
|---|---|---|
| indexName | Index Name | Name of the Elastic Search Index |
| elasticSearchHost | Elastic Search Host | Name of the Elastic Search Host |
| elasticSearchPort | Elastic Search Port | Port of Elastic Search |
| temporaryTable | Spark Temporary Table for Reading from ES | Spark Temporary Table to be used for reading from Elastic Search |
| sql | SQL for reading from Elastic Search | SQL for reading from Elastic Search. Where condition can be applied here for limiting the rows read from ES. |
| outputColNames | Column Names of the Table | Output Columns Names of the Table |
| outputColTypes | Column Types of the Table | Output Column Types of the Table |
| outputColFormats | Column Formats | Output Column Formats |

### 24.1.5 11-ML-SparkML

#### 12-FreqPatternMining

#### FPGrowth

Does Pattern Mining using FPGrowth Algorithm

#### Type

transform

#### Class

fire.nodes.ml.NodeFPGrowth

#### Fields

| Name | Title | Description |
|---|---|---|
| transactionCol | Transaction Column | Input data set, each element contains a transaction |
| minSupport | Min Support | The minimum support for an itemset to be identified as frequent |
| numPartitions | Number of Partitions | The number of partitions used to distribute the work |

#### Details

This node does Pattern Mining using FPGrowth Algorithm.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/mllib-frequent-pattern-mining.html

### 04-FeatureTransformers

#### VectorAssembler

Merges multiple columns into a vector column

##### Input

It takes in a DataFrame and transforms it to another DataFrame

##### Output

It adds a new column to the incoming DataFrame. The new column contains the values of the input columns concatenated into a vector in the specified order.

##### Type

ml-transformer

##### Class

fire.nodes.ml.NodeVectorAssembler

##### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Input Columns | Input column of type - all numeric, boolean and vector |
| outputCol | Output Column | Output column name |

#### IDF

Compute the Inverse Document Frequency (IDF) given a collection of documents.

##### Input

It takes in a DataFrame and transforms it to another DataFrame

#### Output

The output DataFrame contains a new column of type vector, It takes feature vectors (generally created from HashingTF) as input and scales each column. Intuitively, it down-weights columns which appear frequently in a corpus.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeIDF

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | Input Column Name |
| outputCol | Output Column | Output column name |
| minDocFreq | MinDocFreq | The minimum of documents in which a term should appear. |

### StopWordsRemover

Filters out stop words from input. Null values from input array are preserved unless adding null to stopWords explicitly.

#### Output

It adds a new column containing the sequence of strings from the input column but with the stop words removed, to the incoming DataFrame.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeStopWordsRemover

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | Column containing the array text from which the stop words have to be removed |
| outputCol | Output Column | Contains array of text by dropping list of stop words |
| caseSensitive | Case Sensitive | Case Sensitive |
| stopWords | Comma Separated List of Custom Stop Words. If not provided, the default list of stop words would be used. | Custom List of Stop Words |

#### Details

Stop words filters out stop words from input. Null values from input array are preserved unless adding null to stop-Words explicitly.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-features.html#stopwordsremover

### Tokenizer

A tokenizer that converts the input string to lowercase and then splits it by white spaces.

#### Input

It takes in a DataFrame and transforms it to another DataFrame

#### Output

It adds a new column containing the results of tokenization of the input column, to the incoming DataFrame.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeTokenizer

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | Column containing text (such as sentence) |
| outputCol | Output Column | Output column name |

### PolynominalExpansion

Perform feature expansion in a polynomial space

#### Input

It takes in a DataFrame and transforms it to another DataFrame

#### Output

The output DataFrame contains a new column of type vector, Expanding your features into a polynomial space, which is formulated by an n-degree combination of original dimensions.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodePolynominalExpansion

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | The input column name |
| outputCol | Output Column | The output column name |
| degree | Degree | The polynomial degree to expand, which should be >= 1. A value of 1 means no expansion. |

### VectorIndexer

Vector Indexer indexes categorical features inside of a Vector. It decides which features are categorical and converts them to category indices. The decision is based on the number of distinct values of a feature.

#### Input

It takes in a DataFrame and transforms it to another DataFrame

#### Output

It indexes categorical features in datasets of Vectors and stores the result into a new column of the DataFrame.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeVectorIndexer

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | The Input column name |
| outputCol | Output Column | Output column name |
| maxCategories | Maximum Categories | Threshold for the number of values a categorical feature can take. If a feature is found to have > maxCategories values, then it is declared continuous. Must be >= 2 |

### Normalizer

Normalizer is a Transformer which transforms a dataset of Vector rows, normalizing each Vector to have unit norm.

#### Input

It takes in a DataFrame and transforms it to another DataFrame

#### Output

It adds a new column containing the normalized value of the input column, to the incoming DataFrame.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeNormalizer

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | The input column name |
| outputCol | Output Column | The output column name |
| p | P | Normalization in L^p space. Must be >= 1. (default: p = 2) |

#### Details

Normalizer is a Transformer which transforms a dataset of Vector rows, normalizing each Vector to have unit norm.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-features.html#normalizer

### OneHotEncoder

Maps a column of label indices to a column of binary vectors, with at most a single one-value

#### Input

It takes in a DataFrame and transforms it to another DataFrame

#### Output

The output DataFrame contains a new column which contains the mapping of a column of label indices to a column of binary vectors, with at most a single one-value.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeOneHotEncoder

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Input Columns | Input columns for encoding |
| outputCols | Output Columns | Output columns |

### NGramTransformer

Converts the input array of strings into an array of n-grams. Null values in the input array are ignored. It returns an array of n-grams where each n-gram is represented by a space-separated string of words.When the input is empty, an empty array is returned. When the input array length is less than n (number of elements per n-gram), no n-grams are returned

#### Input

It takes in a DataFrame as input and transforms it to another DataFrame

#### Output

It adds a new column consisting of a sequence of nn-grams where each nn-gram is represented by a space-delimited string of nn consecutive words, to the incoming DataFrame

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeNGramTransformer

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | Contains sequence of strings |
| inputColStringArrCol | List of Words | Sequence of words |
| outputCol | Output Column | Consist of a sequence of n-grams where each n-gram is represented by a space-delimited string of n consecutive words |
| numberOfGrams | Number of Grams | Sequence of 'string array' for integer 'Number of Grams' |

#### Details

This node converts the input array of strings into an array of n-grams. Null values in the input array are ignored. It returns an array of n-grams where each n-gram is represented by a space-separated string of words.When the input is empty, an empty array is returned. When the input array length is less than n (number of elements per n-gram), no n-grams are returned"

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-features.html#n-gram

#### Binarizer

Binarize a column of continuous features given a threshold.

##### Input

This type of node takes in a DataFrame and transforms it to another DataFrame

##### Output

A new column containing the binarized values is added to the incoming DataFrame

##### Type

ml-transformer

##### Class

fire.nodes.ml.NodeBinarizer

##### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | The input column name |
| outputCol | Output Column | The output column name |
| threshold | Threshold | The features greater than the threshold, will be binarized to 1.0.The features equal to or less than the threshold, will be binarized to 0.0. |

##### Details

This node binarizes a column of continuous features given a threshold.

More at Spark MLlib/ML docs page : https://spark.apache.org/docs/latest/ml-features.html#binarizer

#### VectorFunctions

Vector Functions for transforming Vectors

##### Type

ml-transformer

#### Class

fire.nodes.ml.NodeVectorFunctions

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | The Input column name |
| vectorFunction | Vector Function | Vector Function Name |
| parameter | Parameter | Parameter for the Function |
| outputCol | Output Column | Output column name |

### WordToScoreMapping

Map the original word of hashValue to score.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeWordToScoreMapping

#### Fields

| Name | Title | Description |
|---|---|---|
| words | Words | Array of words |
| features | Features | Vector with hash value of words |
| output | Output | |

### IndexString

Maps a column of indices back to a new column of corresponding string values. The index-string mapping is either from the ML attributes of the input column, or from user-supplied labels

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeIndexString

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | Column containing label indices |
| outputCol | Output Column | Output column name |

#### Details

This node maps a column of indices back to a new column of corresponding string values. The index-string mapping is either from the ML attributes of the input column, or from user-supplied labels

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-features.html#indextostring

### QuantileDiscretizer

QuantileDiscretizer takes a column with continuous features and outputs a column with binned categorical features.

#### Input

It takes in a DataFrame and transforms it to another DataFrame

#### Output

The output DataFrame contains a new column of binned categorical features.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeQuantileDiscretizer

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | The Input column name |
| outputCol | Output Column | Output column name |
| numBuckets | NumBuckets | Maximum number of buckets (quantiles or categories) into which the data points are grouped. Must be >= 2. |

### Details

QuantileDiscretizer takes a column with continuous features and outputs a column with binned categorical features.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-features.html#quantilediscretizer

## SQLTransformer

This node runs the given SQL on the incoming DataFrame using Spark ML SQLTransformer

### Type

transform

### Class

fire.nodes.ml.NodeSQLTransformer

### Fields

| Name | Title | Description |
|---|---|---|
| tempTable | Temp Table | Temp Table Name to be used |
| sql | SQL | SQL to be run |
| outputColNames | Output Column Names | Name of the Output Columns |
| outputColTypes | Output Column Types | Data Type of the Output Columns |
| outputColFormats | Output Column Formats | Format of the Output Columns |

## StringIndexer

StringIndexer encodes a string column of labels to a column of label indices

### Input

It takes in a DataFrame and transforms it to another DataFrame

### Output

It adds a new column containing the encoding of the string column of labels to a column of label indices, to the incoming DataFrame.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeStringIndexer

#### Fields

| Name | Title | Description |
|---|---|---|
| handleInvalid | Handle Invalid | Invalid entries to be skipped or thrown error |
| inputCols | Input Columns | Input columns for encoding |
| outputCols | Output Columns | Output columns |

### 03-FeatureExtraction

#### RFormula

RFormula feature selection, RFormula selects columns specified by an R model formula. Currently we support a limited subset of the R operators, including '~', '.', ':', '+', and '-'

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeRFormula

#### Fields

| Name | Title | Description |
|---|---|---|
| featuresCol | Features Column | The features column name |
| formula | Formula | formula |
| labelCol | Label Column | The label column name |

#### HashingTF

Maps a sequence of terms to term frequencies using the hashing trick.

#### Input

It takes in a DataFrame as input and transforms it to another DataFrame

#### Output

A new column is added to the input DataFrame containing hashing of the bag of words into a feature vector

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeHashingTF

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | Contains sets of terms. In text processing, a 'set of terms' might be a bag of words |
| outputCol | Output Column | Output column name |

### CountVectorizer

Extracts the vocabulary from a given collection of documents and generates a vector of token counts for each document.

#### Input

It takes in a DataFrame as input and transforms it to another DataFrame

#### Output

It adds a new column to the incoming DataFrame containing the vector of token counts in the input column, to generate the output DataFrame

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeCountVectorizer

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | Input column name |
| outputCol | Output Column | Output column name |
| vocabularySize | Vocabulary Size | Max size of the vocabulary. |

#### Details

CountVectorizer and CountVectorizerModel aim to help convert a collection of text documents to vectors of token counts. When an a-priori dictionary is not available, CountVectorizer can be used as an Estimator to extract the vocabulary and generates a CountVectorizerModel. The model produces sparse representations for the documents over the vocabulary, which can then be passed to other algorithms like LDA.

More at Spark MLlib/ML docs page : https://spark.apache.org/docs/latest/ml-features.html#countvectorizer

### Word2Vec

Transforms vectors of words into vectors of numeric codes for the purpose of further processing by NLP or machine learning algorithms.

#### Input

It takes in a DataFrame as input and transforms it to another DataFrame

#### Output

A new column containing feature vector is added to the incoming DataFrame

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeWord2Vec

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | Contains sequences of words |
| inputColStringArrCol | Text Array Column | The text array column which is produced |
| outputCol | Output Column | Output column name |
| vectorSize | Vector Size | Vector Size |
| minCount | Min Count | Min Count |

### Details

Word2Vec is an Estimator which takes sequences of words representing documents and trains a Word2VecModel. The model maps each word to a unique fixed-size vector. The Word2VecModel transforms each document into a vector using the average of all words in the document; this vector can then be used for as features for prediction, document similarity calculations, etc.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-features.html#word2vec

## 11-CollaborativeFiltering

### ALS

Alternating Least Squares (ALS) matrix factorization.

### Input

It takes in a DataFrame as input and performs ALS

### Output

It generates the ALSModel and passes it to the next Predict and ModelSave Nodes. It also passes the incoming DataFrame to the next Nodes

### Type

ml-estimator

### Class

fire.nodes.ml.NodeALS

### Fields

| Name | Title | Description |
|---|---|---|
| userCol | User Column | The column name for user ids. |
| itemCol | Item Column | The column name for item ids. |
| ratingCol | Rating Column | The column name for ratings. |
| predictionCol | Prediction Column | The prediction column created during model scoring |
| maxIter | Max iterations | The maximum number of iterations. |
| regParam | Regularization Param | The regularization parameter.(>=0) |
| alpha | Alpha | The alpha parameter in the implicit preference formulation.(>=0) |
| checkpointInterval | Checkpoint Interval | The checkpoint interval. |
| nonnegative | Non negative | Whether to apply nonnegativity constraints. |
| numItemBlocks | Num Item Blocks | The number of item blocks. |
| numUserBlocks | Num User Blocks | The number of user blocks. |
| rank | Rank | The rank of the matrix factorization. |
| seed | Seed | Random Seed. |
| implicitPrefs | Implicit Prefs | whether to use implicit preference |

### Details

Collaborative filtering is commonly used for recommender systems. These techniques aim to fill in the missing entries of a user-item association matrix. spark.mllib currently supports model-based collaborative filtering, in which users and products are described by a small set of latent factors that can be used to predict missing entries. spark.mllib uses the alternating least squares (ALS) algorithm to learn these latent factors.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html

### 09-Regression

### GBTRegression

It supports both continuous and categorical features.

### Input

This takes in a DataFrame and performs Logistic Regression

### Output

It generates the GBTRegression and passes it to the next Predict and ModelSave Nodes. The input DataFrame is also passed along to the next nodes.

### Type

ml-estimator

#### Class

fire.nodes.ml.NodeGBTRegression

#### Fields

#### Details

GBT Regression supports both continuous and categorical features.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-classification-regression.html#gradient-boosted-trees-gbts

### AFTSurvivalRegression

Accelerated failure time (AFT) model which is a parametric survival regression model for censored data.

#### Output

It generates the LAFTSurvivalRegressionModel and passes it to the next Predict and ModelSave Nodes. The input DataFrame is also passed along to the next nodes.

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeAFTSurvivalRegression

#### Fields

| Name | Title | Description |
|---|---|---|
| featuresCol | Features Column | Features column of type vectorUDT for model fitting |
| labelCol | Label Column | The label column for model fitting |
| censorCol | Censor Column | Indicator of the event has occurred or not. If the value is 1.O, it means the event has occurred i.e. uncensored; otherwise censored |
| fitIntercept | Fit Intercept | Whether to fit an intercept term |
| maxIter | Maximum Iterations | Maximum number of iterations (>= 0) |
| tol | Tolerance | The convergence tolerance for iterative algorithms |
| quantileProbabilities | QuantileProbabilities | Values of the quantile probabilities array should be in the range (0, 1) |
| quantilesCol | Quantiles Column | The quantiles column created during model scoring |
| predictionCol | Prediction Column | The prediction column created during model scoring |

#### Details

More at Spark MLlib/ML docs page : https://spark.apache.org/docs/latest/ml-classification-regression.html#survival-regression

### XGBoostRegressor

#### Input

It takes in a DataFrame as input and performs XGBoost Regression

#### Output

The XGBoost Model generated is passed along to the next nodes. The input DataFrame is also passed along to the next nodes

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeXGBoostRegressor

## Fields

| Name | Title | Description |
| --- | --- | --- |
| featuresCol | Features Column | Features column of type vectorUDT for model fitting |
| labelCol | Label Column | The label column for model fitting |
| predictionCol | Prediction Column | The prediction column created during model scoring. |
| maxDepth | Max Depth | The Maximum depth of a tree |
| maxBins | Max Bins | The maximum number of bins used for discretizing continuous features.Must be >= 2 and >= number of categories in any categorical feature. |
| maxLeaves | Max Leaves | |
| numRound | Num Round | |
| numWorkers | Num Workers | |
| objective | Objective | |
| eta | Eta | |
| regLambda | Reg Lambda | |
| regAlpha | Reg Alpha | |
| subsample | Subsample | |
| sampleType | SampleType | |
| treeMethod | TreeMethod | |
| useExternalMemory | UseExternalMemory | |
| seed | Seed | |
| baseScore | Base Score | |
| minChildWeight | Min Child Weight | |
| colsampleBylevel | ColSampleByLevel | |
| colsampleBytree | ColSampleByTree | |
| minSplitLoss | MinSplitLoss | |
| maxDeltaStep | MaxDeltaStep | |
| sketchEps | SketchEps | |
| scalePosWeight | ScalePosWeight | |
| growPlicy | GrowPlicy | |
| normalizeType | NormalizeType | |
| skipDrop | SkipDrop | |
| rateDrop | RateDrop | |

## DecisionTreeRegression

It supports both continuous and categorical features.

## Input

This takes in a DataFrame and performs Decision Tree Regression

#### Output

The Decision Tree Regression Model generated is passed along to the next nodes. The input DataFrame is also passed along to the next nodes

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeDecisionTreeRegression

#### Fields

#### Details

Decision tree supports both continuous and categorical features.

More at Spark MLlib/ML docs page : https://spark.apache.org/docs/1.6.0/ml-classification-regression.html#decision-tree-regression

### RandomForestRegression

It supports both continuous and categorical features.

#### Input

This takes in a DataFrame and performs Random Forest Regression

#### Output

It generates the Random Forest Regression Model and passes it to the next Predict and ModelSave Nodes. The input DataFrame is also passed along to the next nodes.

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeRandomForestRegression

#### Fields

#### LinearRegression

The interface for working with linear regression models and model summaries is similar to the logistic regression case.

##### Input

This takes in a DataFrame and performs Logistic Regression

##### Output

It generates the LinearRegressionModel and passes it to the next Predict and ModelSave Nodes. The input DataFrame is also passed along to the next nodes.

##### Type

ml-estimator

##### Class

fire.nodes.ml.NodeLinearRegression

##### Fields

##### Details

The interface for working with linear regression models and model summaries is similar to the logistic regression case.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-classification-regression.html#linear-regression

### 08-Clustering

#### LDA

LDA is given a collection of documents as input data, via the featuresCol parameter. Each document is specified as a Vector of length vocabSize, where each entry is the count for the corresponding term (word) in the document

##### Input

It takes in a DataFrame as input and performs LDA

##### Output

LDA Model is passed to the next Node for Prediction or Storing

### Type

ml-estimator

### Class

fire.nodes.ml.NodeLDA

### Fields

| Name | Title | Description |
|---|---|---|
| featuresCol | Features Column | Features column of type vectorUDT for model fitting. |
| k | K | The number of topics to create. |
| maxIter | Max Iterations | The maximum number of iterations. |
| optimizer | Optimizer | Optimizer or inference algorithm used to estimate the LDA model. |
| topicDistributionCol | TopicDistributionColumn | Output column with estimates of the topic mixture distribution for each document |
| checkpointInterval | checkpointInterval | The checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that the cache will get checkpointed every 10 iterations. |
| subsamplingRate | subsamplingRate | Fraction of the corpus to be sampled and used in each iteration of mini-batch gradient descent, in range (0, 1]. |
| seed | Seed | Random Seed. |
| maxTermsPerTopic | MaxTermsPerTopic | Number of Terms in Topics |

### GaussianMixture

This class performs expectation maximization for multivariate Gaussian Mixture Models (GMMs). A GMM represents a composite distribution of independent Gaussian distributions with associated mixing weights specifying each's contribution to the composite.

### Input

It takes in a DataFrame as input and performs GaussianMixture clustering

### Output

The input DataFrame is passed along to the next Processors

### Type

ml-estimator

#### Class

fire.nodes.ml.NodeGaussianMixture

#### Fields

| Name | Title | Description |
|---|---|---|
| featuresCol | Features Column | Features column of type vectorUDT for model fitting. |
| k | K | The number of clusters to create. |
| maxIter | Max Iterations | The maximum number of iterations. |
| predictionCol | Prediction Column | The prediction column created during model scoring. |
| seed | Seed | Random Seed. |
| tol | Tolerence | The convergence tolerance for iterative algorithms. |

#### Details

GaussianMixture clustering will maximize the log-likelihood for a mixture of k Gaussians, iterating until the log-likelihood changes by less than convergenceTol, or until it has reached the max number of iterations. While this process is generally guaranteed to converge, it is not guaranteed to find a global optimum.

More at Spark MLlib/ML docs page : https://spark.apache.org/docs/2.2.0/mllib-clustering.html#gaussian-mixture

### KMeans

K-means clustering with support for k-means|| initialization proposed by Bahmani et al

#### Input

It takes in a DataFrame as input and performs K-Means clustering

#### Output

The input DataFrame is passed along to the next Processors

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeKMeans

#### Fields

| Name | Title | Description |
|---|---|---|
| featuresCol | Features Column | Features column of type vectorUDT for model fitting. |
| k | K | The number of clusters to create. |
| maxIter | Max Iterations | The maximum number of iterations. |
| predictionCol | Prediction Column | The prediction column created during model scoring. |
| seed | Seed | Random Seed. |
| tol | Tolerence | The convergence tolerance for iterative algorithms. |
| initMode | initMode | The initialization algorithm mode. |
| initSteps | initSteps | The number of steps for the k-means‖ initialization mode. It will be ignored when other initialization modes are chosen. |

#### Details

K-means clustering with support for k-means‖ initialization proposed by Bahmani et al

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/mllib-clustering.html#k-means

### 05-DimensionalityReduction

#### SVD

#### Type

transform

#### Class

fire.nodes.ml.NodeSVD

#### Fields

#### PCA

Trains a model to project vectors to a low-dimensional space using PCA.

#### Input

This takes in a DataFrame as input

#### Output

The output DataFrame is a projection of the vectors in the incoming DataFrame to a low-dimensional space using PCA

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodePCA

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | The input column name |
| outputCol | Output Column | The output column name |
| k | K | The number of principal components |

#### Details

PCA trains a model to project vectors to a low-dimensional space using PCA.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-features.html#pca

### 02-FeatureScaler

#### MinMaxScaler

MinMaxScaler transforms a dataset of Vector rows, rescaling each feature to a specific range (often [0, 1])

#### Input

It takes in a DataFrame as input and transforms it to another DataFrame

#### Output

A new column containing the scaled features is added to the incoming DataFrame

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeMinMaxScaler

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | The input column name |
| outputCol | Output Column | The output column name |
| max | Max | The upper bound after transformation, shared by all features |
| min | Min | The lower bound after transformation, shared by all features |

### StandardScaler

StandardScaler transforms a dataset of Vector rows, normalizing each feature to have unit standard deviation and/or zero mean.

#### Input

It takes in a DataFrame as input and transforms it to another DataFrame

#### Output

It adds a new column containing the transform of the input Vector column to unit standard deviation and/or zero mean features to the incoming DataFrame.

#### Type

ml-transformer

#### Class

fire.nodes.ml.NodeStandardScaler

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | The input column name |
| outputCol | Output Column | The output column name |
| withMean | With Mean | Centers the data with mean before scaling. |
| withStd | With Standard Dev | Scales the data to unit standard deviation |

#### Details

StandardScaler transforms a dataset of Vector rows, normalizing each feature to have unit standard deviation and/or zero mean.

StandardScaler is an Estimator which can be fit on a dataset to produce a StandardScalerModel; this amounts to computing summary statistics. The model can then transform a Vector column in a dataset to have unit standard deviation and/or zero mean features.

If the standard deviation of a feature is zero, it will return default 0.0 value in the Vector for that feature.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-features.html#standardscaler

#### 17-Util

#### Spark ML Model Load

#### Type

ml-modelload

#### Class

fire.nodes.ml.NodeModelLoad

#### Fields

#### TrainValidationSplit

This node represents Train Validation Split from Spark ML

#### Input

TrainValidationSplit takes an Estimator, a set of ParamMaps provided in the estimatorParamMaps parameter, and anEvaluator.

#### Output

The incoming DataFrame is passed to the output.

#### Type

ml-trainvalidationsplit

#### Class

fire.nodes.ml.NodeTrainValidationSplit

#### Fields

| Name | Title | Description |
|---|---|---|
| trainRatio | Train Ratio | Training Ratio |

#### Details

This node represents Train Validation Split from Spark ML.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-guide.html#example-model-selection-via-train-validation-split

### Spark ML Model Save

This node saves the ML model generated at the specified path

#### Input

It takes in a Model and DataFrame as input.

#### Output

The incoming dataframe is passed to the output.

#### Type

ml-modelsave

#### Class

fire.nodes.ml.NodeModelSave

#### Fields

### Spark ML ROC

#### Type

transform

#### Class

fire.nodes.etl.NodeROC

#### Fields

| Name | Title | Description |
|---|---|---|
| probabilityCol | Probability Column | |
| labelCol | Label Column | |

### CrossValidator

This node represents Cross Validator from Spark ML

#### Input

It takes in a DataFrame, Estimator and Evaluator as input.

#### Output

The incoming dataframe is passed to the output.

#### Type

ml-crossvalidator

#### Class

fire.nodes.ml.NodeCrossValidator

#### Fields

| Name | Title | Description |
|---|---|---|
| numFolds | Num Folds | The number of folds |

#### Details

This node represents Cross Validator from Spark ML.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-guide.html#example-model-selection-via-cross-validation

### Spark Pipeline

This node represents Pipeline from Spark ML

#### Input

It takes in a DataFrame as input.

#### Output

The incoming DataFrame is passed to the output.

#### Type

ml-pipeline

#### Class

fire.nodes.ml.NodePipeline

#### Fields

#### Details

This node represents Pipeline from Spark ML.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-guide.html#pipeline-components

### 07-SplitDataset

#### Split With Stratified Sampling

This node splits the incoming DataFrame into 2. It takes in the fraction to use in splitting the data by Stratified Sampling.

#### Input

It takes in a DataFrame as input

#### Output

The input DataFrame is split into 2 DataFrames and output

#### Type

transform

### Class

fire.nodes.util.SplitWithStratifiedSampling

### Fields

| Name | Title | Description |
|---|---|---|
| keyInputCol | Column Name | column that defines strata |
| fraction | Fraction | sampling fraction for each stratum. If a stratum is not specified, we treat its fraction as zero |
| seed | Seed | random seed |

### Details

Split With Stratified Sampling, which is the preferred way to sample from populations with varing subpopulation sizes.

More details are available at : https://spark.apache.org/docs/latest/api/python/_modules/pyspark/sql/dataframe.html#DataFrame.sampleBy

## Split

This node splits the incoming DataFrame into 2. It takes in the fraction to use in splitting the data. For example, if the fraction is .7, it would split the data into 2 DataFrames, one containing 70% of the rows and the other containing the remaining 30%.

### Input

It takes in a DataFrame as input

### Output

The input DataFrame is split into 2 DataFrames and output

### Type

transform

### Class

fire.nodes.ml.NodeSplit

#### Fields

| Name | Title | Description |
|---|---|---|
| fraction1 | Fraction 1 | Fraction to be used for Splitting the DataFrame into two. The first DataFrame would go to the lower edge output. The other would go to the higher edge output. |

### SplitProbabilityColumn

#### Type

transform

#### Class

fire.nodes.ml.NodeSplitProbabilityCol

#### Fields

| Name | Title | Description |
|---|---|---|
| probabilityColName | Probability Column | |
| numFields | NumFields | Number of fields in probability columns to extract |

## 10-Classification

### MultiLayerPerceptron

It supports creation of full connected neural network.

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeMultilayerPerceptron

### Fields

| Name | Title | Description |
|---|---|---|
| featuresCol | Features Column | Features column of type vectorUDT for model fitting |
| labelCol | Label Column | The label column for model fitting |
| predictionCol | Prediction Column | The prediction column created during model scoring. |
| layers | Layers - comma separated list of integers | The integer array specifying the number of activation units in each layer |
| maxIter | Max number of iterations | Number of iterations to train the Neural network |
| blockSize | Block Size | Block size |
| seed | Seed | The initial seed to initialise the neural network. |
| tol | Tol | |
| solver | Solver | solver |
| stepSize | Step Size | Step size |

## GBTClassifier

Gradient-Boosted Trees (GBTs) is a learning algorithm for classification. It supports binary labels, as well as both continuous and categorical features. Note: Multiclass labels are not currently supported.

### Input

It takes in a DataFrame as input and performs GBT Classification

### Output

The GBT Model generated is passed along to the next nodes. The input DataFrame is also passed along to the next nodes

### Type

ml-estimator

### Class

fire.nodes.ml.NodeGBTClassifier

### Fields

## XGBoostClassifier

### Input

It takes in a DataFrame as input and performs XGBoost Classification

### Output

The XGBoost Model generated is passed along to the next nodes. The input DataFrame is also passed along to the next nodes

### Type

ml-estimator

### Class

fire.nodes.ml.NodeXGBoostClassifier

### Fields

| Name | Title | Description |
|---|---|---|
| featuresCol | Features Column | Features column of type vectorUDT for model fitting |
| labelCol | Label Column | The label column for model fitting |
| predictionCol | Prediction Column | The prediction column created during model scoring. |
| numClass | Num Class | |
| maxDepth | Max Depth | The Maximum depth of a tree |
| maxBins | Max Bins | The maximum number of bins used for discretizing continuous features.Must be >= 2 and >= number of categories in any categorical feature. |
| maxLeaves | Max Leaves | |
| numRound | Num Round | |
| numWorkers | Num Workers | |
| objective | Objective | |
| eta | Eta | |
| regLambda | Reg Lambda | |
| regAlpha | Reg Alpha | |
| subsample | Subsample | |
| sampleType | SampleType | |
| treeMethod | TreeMethod | |
| useExternalMemory | UseExternalMemory | |
| seed | Seed | |
| baseScore | Base Score | |
| minChildWeight | Min Child Weight | |
| colsampleBylevel | ColSampleByLevel | |
| colsampleBytree | ColSampleByTree | |

Continued on next page

Table 1 – continued from previous page

| Name | Title | Description |
|---|---|---|
| minSplitLoss | MinSplitLoss | |
| maxDeltaStep | MaxDeltaStep | |
| sketchEps | SketchEps | |
| scalePosWeight | ScalePosWeight | |
| growPlicy | GrowPlicy | |
| normalizeType | NormalizeType | |
| skipDrop | SkipDrop | |
| rateDrop | RateDrop | |

### LogisticRegression

Logistic regression. Currently, this class only supports binary classification.

#### Input

This takes in a DataFrame and performs Logistic Regression

#### Output

The Logistic Regression Model generated is passed along to the next nodes. The input DataFrame is also passed along to the next nodes

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeLogisticRegression

#### Fields

#### Details

Logistic regression is a popular method to predict a categorical response.

It is a special case of Generalized Linear models that predicts the probability of the outcomes. In spark.ml logistic regression can be used to predict a binary outcome by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial logistic regression.

More details are available at : https://spark.apache.org/docs/2.3.0/ml-classification-regression.html#logistic-regression

#### Examples

##### The below example is available at : https://spark.apache.org/docs/2.3.0/ml-classification-regression.html#logistic-regression

import org.apache.spark.ml.classification.LogisticRegression

// Load training data val training = spark.read.format("libsvm").load("data/mllib/sample_libsvm_data.txt")

**val lr = new LogisticRegression()** .setMaxIter(10) .setRegParam(0.3) .setElasticNetParam(0.8)

// Fit the model val lrModel = lr.fit(training)

// Print the coefficients and intercept for logistic regression println(s"Coefficients: ${lrModel.coefficients} Intercept: ${lrModel.intercept}")

// We can also use the multinomial family for binary classification val mlr = new LogisticRegression()

.setMaxIter(10) .setRegParam(0.3) .setElasticNetParam(0.8) .setFamily("multinomial")

val mlrModel = mlr.fit(training)

// Print the coefficients and intercepts for logistic regression with multinomial family println(s"Multinomial coefficients: ${mlrModel.coefficientMatrix}") println(s"Multinomial intercepts: ${mlrModel.interceptVector}")

#### DecisionTreeClassifier

It supports both binary and multiclass labels, as well as both continuous and categorical features.

#### Input

It takes in a DataFrame and performs Decision Tree Classification

#### Output

The Decision Tree Model generated is passed along to the next nodes. The input DataFrame is also passed along to the next nodes

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeDecisionTreeClassifier

#### Fields

#### Details

Decision trees supports both binary and multiclass labels, as well as both continuous and categorical features.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-classification-regression.html#decision-tree-classifier

### NaiveBayes

Creates a NaiveBayes model. Supports both Multinomial NB which can handle finitely supported discrete data. For example, by converting documents into TF-IDF vectors, it can be used for document classification. By making every vector a binary (0/1) data, it can also be used as Bernoulli NB.The input feature values must be nonnegative

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeNaiveBayes

#### Fields

| Name | Title | Description |
|---|---|---|
| featuresCol | Features Column | Features column of type vectorUDT for model fitting |
| labelCol | Label Column | The label column for model fitting |
| predictionCol | Prediction Column | The prediction column created during model scoring |
| modelType | modelType | The model type. Supported options: multinomial and bernoulli. (default = multinomial) |
| smoothing | Smoothing | The smoothing parameter. |

### RandomForestClassifier

Supports both binary and multiclass labels, as well as both continuous and categorical features.

#### Input

Takes in a DataFrame and performs Random Forest Classification

#### Output

Random Forest Classification Model generated is passed along to the next nodes. The input DataFrame is also passed along to the next nodes

#### Type

ml-estimator

#### Class

fire.nodes.ml.NodeRandomForestClassifier

#### Fields

#### Details

Random forests supports both binary and multiclass labels, as well as both continuous and categorical features.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/ml-classification-regression.html#random-forest-classifier

### 13-EvaluatePredict

### MulticlassClassificationEvaluator

Evaluator for multiclass classification, which expects two input columns: score and label.

#### Type

ml-evaluator

#### Class

fire.nodes.ml.NodeMulticlassClassificationEvaluator

#### Fields

| Name | Title | Description |
|---|---|---|
| labelCol | Label Column | The label column for model fitting. |
| predictionCol | Prediction Column | The prediction column. |
| metricName | Metric Name | The metric used in evaluation. |

#### Details

Evaluator for multiclass classification, which expects two input columns: score and label.

More at Spark MLlib/ML docs page :https://spark.apache.org/docs/1.6.0/mllib-evaluation-metrics.html#multiclass-classification

### RegressionEvaluator

Evaluator for regression, which expects two input columns: prediction and label.

#### Input

It takes in a DataFrame as input

#### Output

The incoming DataFrame is passed to the output

#### Type

ml-evaluator

#### Class

fire.nodes.ml.NodeRegressionEvaluator

#### Fields

| Name | Title | Description |
|---|---|---|
| labelCol | Label Column | The label column for model fitting. |
| predictionCol | Prediction Column | The prediction column. |
| metricName | Metric Name | The metric used in evaluation. |

#### Details

Evaluator for regression, which expects two input columns: prediction and label.

More at Spark MLlib/ML docs page:

http://spark.apache.org/docs/1.6.0/api/scala/index.html#org.apache.spark.ml.evaluation.RegressionEvaluator

### Predict

Predict node takes in a DataFrame and Model and makes predictions

#### Input

It takes in a DataFrame and Model as input

#### Output

A new column containing the predictions is added to the input DataFrame

#### Type

ml-predict

#### Class

fire.nodes.ml.NodePredict

#### Fields

### BinaryClassificationEvaluator

Evaluator for binary classification, which expects two input columns: rawPrediction and label.

#### Output

It outputs the Probability for each class

#### Type

ml-evaluator

#### Class

fire.nodes.ml.NodeBinaryClassificationEvaluator

#### Fields

| Name | Title | Description |
|---|---|---|
| labelCol | Label Column | The label column for model fitting. |
| predictionCol | Prediction Column | The prediction column. |
| metricName | Metric Name | The metric used in evaluation. |

#### Details

Evaluator for binary classification, which expects two input columns: rawPrediction and label.

More at Spark MLlib/ML docs page : http://spark.apache.org/docs/latest/mllib-evaluation-metrics.html#binary-classification

### 06-FeatureSelection

#### ChiSqSelector

Chi-Squared feature selection, which selects categorical features to use for predicting a categorical label.

##### Type

ml-transformer

##### Class

fire.nodes.ml.NodeChiSqSelector

##### Fields

| Name | Title | Description |
|---|---|---|
| featuresCol | Features Column | The features column name |
| outputCol | Output Column | The output column name |
| labelCol | Label Column | The label column name |
| numTopFeatures | NumTopFeatures | Number of features that selector will select (ordered by statistic value descending). |

#### VectorSlicer

VectorSlicer feature selection, which takes a feature vector and outputs a new feature vector with a sub-array of the original features. It is useful for extracting features from a vector column

##### Type

ml-transformer

##### Class

fire.nodes.ml.NodeVectorSlicer

##### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Features Column | The features column name |
| outputCol | Output Column | The output column name |

### 24.1.6 ML-TS

#### ARIMA

##### Type

ml-transformer

##### Class

fire.nodes.ts.NodeAutoARIMA

##### Fields

| Name | Title | Description |
|---|---|---|
| y | Y | The time-series to which to fit the ARIMA estimator |
| seasonal | Seasonal | Whether to fit a seasonal ARIMA. Default is True |
| stepwise | Stepwise | Whether to use the stepwise algorithm to identify the optimal model parameters. |
| trace | Trace | Whether to print status on the fits. |
| suppress_warnings | Suppress Warnings | If suppress_warnings is True, all of the warnings coming from ARIMA will be squelched. |
| error_action | Error Action | If unable to fit an ARIMA for whatever reason, this controls the error-handling behavior. One of (warn, raise, ignore) |
| scoring | Scoring | The metric to use for scoring the out-of-sample data. One of (mse, mae) |
| n_periods | Forecast | Int number of periods to forecast forward. |

#### Prophet

##### Type

ml-transformer

##### Class

fire.nodes.ts.NodeProphet

##### Fields

### 24.1.7 02-Parse

### FieldSplitter

This node splits the string of the specified input column using the specified delimiter

#### Input

It accepts a DataFrame as input from the previous Node

#### Output

New columns are added to the incoming DataFrame with values from the result of splitting the value in the input column

#### Type

transform

#### Class

fire.nodes.etl.NodeFieldSplitter

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | input column name |
| outputCols | Output Columns | new column names separed by comma','.(eg: col1,co2,col3) |
| sep | Separator | separator to split the input column value(default: space) |
| onError | On Error | |

### RegexTokenizer

This node creates a new DataFrame by the process of taking text (such as a sentence) and breaking it into individual terms (usually words) based on regular express

#### Type

transform

#### Class

fire.nodes.etl.NodeRegexTokenizer

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Column | input column for tokenizing |
| outputCol | Tokenized Column | New output column after tokenization |
| pattern | Pattern | The regex pattern used to match delimiters |
| gaps | Gaps | Indicates whether the regex splits on gaps |

### Fixed Length Fields

Fixed Length

#### Type

transform

#### Class

fire.nodes.etl.NodeFixedLength

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column | input column name |
| outputColNames | Column Names for the CSV | New Output Columns of the SQL |
| outputColTypes | Column Types for the CSV | Data Type of the Output Columns |
| colLengths | Length of each column | Length of the columns in characters |
| outputColFormats | Column Formats for the CSV | Format of the Output Columns |

### ApacheLogs

Reads in Apache Log files from a given path, parses them and loads them into a DataFrame

#### Type

dataset

#### Class

fire.nodes.logs.NodeApacheFileAccessLog

### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Full path for the directory or file for the Apache File Logs |

### ParseJSONCol

Parses JSON content in a given Col

#### Type

transform

#### Class

fire.nodes.etl.NodeParseJSONColumn

#### Fields

| Name | Title | Description |
|---|---|---|
| jsonColName | JSON Col Name | Column containing the JSON Content |
| inputCol | Input Col | Input Columns |
| jsonFieldNames | JSON Field names | JSON Field names |
| jsonFieldTypes | JSON Field Type | Data Type of the JSON field |

### OCR

Performs Optical Character Recognition using the Tesseract Library. Please make sure the TESSDATA_PREFIX environment variable is set to the parent directory of your 'tessdata' directory. Download the tessdata directory with git clone https://github.com/tesseract-ocr/tessdata.git

#### Type

transform

#### Class

fire.nodes.ocr.NodeOCRTesseract

#### Fields

| Name | Title | Description |
|---|---|---|
| imageNameCol | Image Name Column | input image column name |
| imageCol | Image Column | input image column name |
| outputCol | Output OCR Column | output column name |

#### MultiRegexExtractor

This node to extract pattren from input columns

#### Input

This type of node takes in a DataFrame and transforms it to another DataFrame

#### Output

This node extract pattren from input columns as specified

#### Type

transform

#### Class

fire.nodes.etl.NodeMultiRegexExtractor

#### Fields

| Name | Title | Description |
|---|---|---|
| inputColNames | InputColumnsName | Columns |
| outputColNames | OuputColumnsName | name of the output column |
| patterns | Patterns | patterns or regex to extract the input column name |
| groups | Groups | An regular expression group number starting with 1, defining which portion of the matching string will be returned |

### 24.1.8 06-Filter

#### FilterByDateRange

This node filters Rows within the given date range

#### Type

transform

#### Class

fire.nodes.etl.NodeFilterByDateRange

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Column | input column name |
| fromDateCol | From Date | Takes Start Date in the form of yyyy-MM-dd |
| toDateCol | To Date | Takes End Date in the form of yyyy-MM-dd |

### FilterByNumberRange

This node filter Rows in the given Number Range

#### Input

It accepts a DataFrame as input from the previous Node

#### Type

transform

#### Class

fire.nodes.etl.NodeFilterByNumberRange

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column Name | input column name |
| lowestValue | Lowest Value | input lowest value |
| highestValue | Highest Value | input highest value |

### ColumnFilter

This node creates a new DataFrame that contains only the selected columns

#### Input

This type of node takes in a DataFrame and transforms it to another DataFrame.

#### Output

This node filters the specified columns from the incoming DataFrame

#### Type

transform

#### Class

fire.nodes.etl.NodeColumnFilter

#### Fields

| Name | Title | Description |
|---|---|---|
| outputCols | Columns | Columns to be included in the output DataFrame |

### RowFilter

This node creates a new DataFrame containing only rows satisfying given condition

#### Input

It accepts DataFrame as input from the previous Node

#### Output

This node filters the rows based on the conditional expression to generate the output DataFrame

#### Type

transform

#### Class

fire.nodes.etl.NodeRowFilter

### Fields

| Name | Title | Description |
|---|---|---|
| conditionExpr | Conditional Expression | The filtering condition. Rows not satisfying given condition will be excluded from output DataFrame. eg: usd_pledged_real > 0 and (category = 1 or category == 2) and goal > 100 |

### Details

This node creates a new DataFrame containing only rows satisfying the given condition.

### Examples of Conditional Expression

col1 > 5 AND col2 > 3

name is not NULL

name is NULL

usd_pledged_real > 0 and (category = "Narrative Film" or category == "Music") and goal > 100

datetime > '2011-01-01 00:00:00.0' (datetime column is of type timestamp)

datetime > '2011-01-01 00:00:00.0' and datetime < '2016-01-01 00:00:00.0'

## FilterByStringLength

This node filters the Rows within the given string length. The column to be used for determining the string length is specified

### Input

It accepts a DataFrame as input from the previous Node

### Type

transform

### Class

fire.nodes.etl.NodeFilterByStringLength

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column Name | input column name |
| minLength | Minimum length | Minimum length of String |
| maxLength | Maximum length | Maximum length of String |

### NodeRowFilterByIndex

This node creates a new DataFrame containing only rows satisfying given condition

#### Input

It accepts DataFrame as input from the previous Node

#### Output

This node filters the rows based on the conditional expression to generate the output DataFrame

#### Type

transform

#### Class

fire.nodes.etl.NodeRowFilterByIndex

#### Fields

| Name | Title | Description |
|---|---|---|
| indexes | Indexes | Comma separated index values starts from 0. ex: 0, 1, 2, 5 |
| indexesRange | IndexesRange | Index ranges example like 10-14 i.e 10, 11, 12, 13, 14. |

### DropColumns

This node creates a new DataFrame by deleting columns specified as an input

#### Input

It takes in a DataFrame as input

#### Output

The specified columns are dropped from the incoming DataFrame to generate the output DataFrame

#### Type

transform

#### Class

fire.nodes.etl.NodeDropColumns

#### Fields

| Name | Title | Description |
|---|---|---|
| dropCols | Columns | The columns to be excluded from the output DataFrame |

### 24.1.9 18-OpenNLP

#### OpenNLPNameFinder

This node finds names using OpenNLP. It takes in the OpenNLP model. Models can be downloaded from http://opennlp.sourceforge.net/models-1.5/

#### Input

It takes in a DataFrame as input.

#### Output

It extracts the names from the specified column and stores the result in the specified output column.

#### Type

transform

#### Class

fire.nodes.opennlp.NodeOpenNLPNameFinder

#### Fields

| Name | Title | Description |
|---|---|---|
| model | Model | Path to the model file (on HDFS when running on the cluster) |
| inputCol | Input Text Column | input column name |
| outputCol | Output Column | Output Column containing the results |

#### Details

This node performs namefinder using OpenNLP to easily detect named entities and numbers in text.

To be able to detect entities the Name Finder needs a model. The model is dependent on the language and entity type it was trained for.

https://opennlp.apache.org/documentation/1.6.0/manual/opennlp.html#tools.namefind.recognition.cmdline

The OpenNLP project offers a number of pre-trained name finder models which are trained on various freely available corpora. They can be downloaded at the OpenNLP download page.

http://opennlp.sourceforge.net/models-1.5/

### OpenNLPSentenceDetector

This node detects sentences using OpenNLP - https://opennlp.apache.org/documentation/1.7.2/manual/opennlp.html#tools.sentdetect. It takes in the OpenNLP model. Models can be downloaded from http://opennlp.sourceforge.net/models-1.5/

#### Input

It takes in a DataFrame as input.

#### Type

transform

#### Class

fire.nodes.opennlp.NodeOpenNLPSentenceDetector

#### Fields

| Name | Title | Description |
|---|---|---|
| model | Model | Path to the model file (on HDFS when running on the cluster) |
| inputCol | Input Text Column | input cpulmn name |
| outputCol | Output Column | Output Column containing the results |

### Details

This node detects sentences using OpenNLP -

https://opennlp.apache.org/documentation/1.7.2/manual/opennlp.html#tools.sentdetect.

It takes in the OpenNLP model. Models can be downloaded from http://opennlp.sourceforge.net/models-1.5/

## NodeOpenNLPDocumentCategorizer

This node classifies text into pre-defined categories using OpenNLP - https://opennlp.apache.org/documentation/1.7.2/manual/opennlp.html#tools.doccat. It takes in the OpenNLP model. Models can be downloaded from http://opennlp.sourceforge.net/models-1.5/

### Input

It takes in a DataFrame as input.

### Output

It finds the Document Category and stores the result in the specified output column.

### Type

transform

### Class

fire.nodes.opennlp.NodeOpenNLPDocumentCategorizer

### Fields

| Name | Title | Description |
|---|---|---|
| model | Model | Path to the model file (on HDFS when running on the cluster) |
| inputCol | Input Text Column | input cpulmn name |
| outputCol | Output Column | Output Column containing the results |

### Details

This node classifies text into pre-defined categories using OpenNLP

https://opennlp.apache.org/documentation/1.7.2/manual/opennlp.html#tools.doccat.

It takes in the OpenNLP model. Models can be downloaded from http://opennlp.sourceforge.net/models-1.5/

### 24.1.10 15-ScoreCardPy

#### Binning Scorecard

##### Type

ml-transformer

##### Class

fire.nodes.scorecardpy.NodeBinning

##### Fields

| Name | Title | Description |
|---|---|---|
| y | Y | |
| x | X | |
| stopLimit | StopLimit | |
| countDistrLimit | CountDistrLimit | |
| binNumLimit | BinNumLimit | |
| method | Methos | |
| positive | Positive | |

#### VariableSelection Scorecard

##### Type

ml-transformer

##### Class

fire.nodes.scorecardpy.NodeVariableSelection

##### Fields

| Name | Title | Description |
|---|---|---|
| y | Y | |
| ivLimit | IvLimit | |
| missingLimit | MissingLimit | |
| identicalLimit | IdenticalLimit | |
| positive | Positive | |

### 24.1.11 03-Prepare

#### 13-Others

##### MultiWindowAnalytics

###### Type

transform

###### Class

fire.nodes.etl.NodeMultiWindowAnalytics

###### Fields

| Name | Title | Description |
|---|---|---|
| analyticsCols | AnalyticsColumn | |
| windowFunctions | Window Function | Window Function Name |
| partitionByCols | PartitionBy | partition column names separated by comma(,) |
| orderByCols | OrderBy | order by column names separated by comma(,) |
| outPutColumns | OutPutColumn | Enter output field(column) name |

##### RoundValue

###### Input

It takes in a DataFrame as input

###### Type

transform

###### Class

fire.nodes.etl.NodeRoundDouble

###### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Input Column | The columns containing double or float values to round. |
| precision | Precision | The scale of the double values to round to. |

### SortBy

It sorts the incoming DataFrame on the fields specified.

#### Type

transform

#### Class

fire.nodes.etl.NodeSortBy

#### Fields

| Name | Title | Description |
|---|---|---|
| description | Description | Description |
| sortByColNames | Columns | Columns on which to Sort By |
| ascDesc | Sorting Order | Whether to sort in ascending or descending order |

### Transpose

This node transposes a dataframe without performing aggregation function by given column(transposeby). ALL INPUT COLUMNS TO THIS NODE HAVE TO BE OF THE SAME TYPE

#### Input

It accepts a DataFrame as input from the previous Node

#### Output

Output dataframe consisting of three columns transposeBy, column_name, column_value

#### Type

transform

#### Class

fire.nodes.etl.NodeTranspose

#### Fields

| Name | Title | Description |
|---|---|---|
| transposeBy | TransposeByColumn Name | transposeBy column name |

### WindowRanking

#### Type

transform

#### Class

fire.nodes.etl.NodeWindowRanking

#### Fields

| Name | Title | Description |
|---|---|---|
| partitionByCols | PartitionBy | partition column names separated by comma(,) |
| orderByCols | OrderBy | order by column names separated by comma(,) |
| windowFunction | Window Function | Window Function Name |

### GeoPoint

#### Type

transform

#### Class

fire.nodes.etl.NodeGeoPoint

#### Fields

| Name | Title | Description |
|---|---|---|
| longitude | Longitude | |
| latitude | Latitude | |

#### MultiWindowRanking

##### Type

transform

##### Class

fire.nodes.etl.NodeMultiWindowRanking

##### Fields

| Name | Title | Description |
|---|---|---|
| windowFunctions | WindowFunction | Window Function Name |
| partitionByCols | PartitionBy | partition column names separated by comma(,) |
| orderByCols | OrderBy | order by column names separated by comma(,) |
| outPutColumns | OutputColumn | Enter output field(column) name |

#### ColumnsRename

This node creates a new DataFrame by renaming existing columns with new name

##### Input

This type of node takes in a DataFrame and transforms it to another DataFrame.

##### Output

The specified columns are renamed to have the new names.

##### Type

transform

##### Class

fire.nodes.etl.NodeColumnsRename

#### Fields

| Name | Title | Description |
|---|---|---|
| currentColNames | Current Column Names | Current Column Names |
| newColNames | Columns New Name | New name for existing columns |

### RecoverHivePartitions

Node to recover the partitions of external hve table.

#### Type

doc

#### Class

fire.nodes.etl.NodeRecoverHivePartitions

#### Fields

| Name | Title | Description |
|---|---|---|
| databaseName | HIVE Database | Name of the HIVE Database |
| tableName | HIVE Table | Name of the HIVE table |

#### Details

This node is used recover the partitions of external hve table.

It will run the command: "MSCK REPAIR TABLE ${databaseName}.${tableName}"

### CDCUsingFullTableMerge

CDC Using Full Table Merge

#### Type

transform

#### Class

fire.nodes.etl.NodeCDCUsingFullTableMerge

### Fields

| Name | Title | Description |
|---|---|---|
| baseTable | Base Table Name | Name of the Base Table |
| idCols | ID Column Names | ID Column names |
| modifiedDateCol | Modified Date Column | Modified Date Column |

## Count

This node counts the number of records in the incoming Dataframe and puts the count into a variable to the used by subsequent Nodes

### Input

It accepts a DataFrame as input from the previous Node

### Output

The incoming DataFrame is sent to the output

### Type

transform

### Class

fire.nodes.etl.NodeCount

### Fields

| Name | Title | Description |
|---|---|---|
| variable | Variable Name | Name of the Variable in which the count is stored |

## Sample

Samples the incoming DataFrame

### Type

transform

### Class

fire.nodes.etl.NodeSample

### Fields

| Name | Title | Description |
|---|---|---|
| withReplacement | With Replacement | With or without Replacement |
| fraction | Fraction | Fraction |
| seed | Seed | Seed |

## SortColumns

It sort the columns selection.

### Type

transform

### Class

fire.nodes.etl.NodeSortColumns

### Fields

| Name | Title | Description |
|---|---|---|
| sortColumnNames | Columns | Sort the Column Name |

## RegisterTempTable

This node registers the incoming DataFrame as a temporary table in Spark

### Input

It accepts a DataFrame as input from the previous Node

### Output

The incoming DataFrame is output without any changes

### Type

transform

### Class

fire.nodes.etl.NodeRegisterTempTable

### Fields

| Name | Title | Description |
|---|---|---|
| tempTable | Temporary Table | Name of the temporary table to be created |

## GeoIP

This node converts IP to geo location

### Input

The input dataframe is passed in the variable inDF

### Output

Transforms the IP to Geo location

### Type

transform

### Class

fire.nodes.etl.NodeGeoIP

### Fields

| Name | Title | Description |
|---|---|---|
| ipCol | IP Column | IP Column in the DataFrame |
| databaseFilePath | Database File Path | Database File Path |

## WindowAnalytics

### Type

transform

### Class

fire.nodes.etl.NodeWindowAnalytics

### Fields

| Name | Title | Description |
|---|---|---|
| partitionByCols | PartitionBy | partition column names separated by comma(,) |
| orderByCols | OrderBy | order by column names separated by comma(,) |
| windowFunction | Window Function | Window Function Name |
| analyticsCol | Analytics Column | |
| window_offset | Window Offset | It's used in lead and lag functions. |

## 10-Condition

### Assert

This Node takes in an expression. It evaluates the expression and based on the results sends the execution to the first or the second output Node

### Input

It accepts a DataFrame as input from the previous Node

### Output

The incoming DataFrame is sent to the output. Only one of the output Nodes receives the DataFrame based on the results of the expression

### Type

transform

### Class

fire.nodes.etl.NodeAssert

### Fields

| Name | Title | Description |
|---|---|---|
| expression | Expression | Expression to be evaluated. It can use variables computed in the previous Nodes |

### Decision

It computes expressions to determine if the condition is met or not. Accordingly proceeds to the next step or stops here.

#### Type

transform

#### Class

fire.nodes.etl.NodeDecision

#### Fields

| Name | Title | Description |
|---|---|---|
| description | Description | Description |
| inputCols | Columns | Columns |
| functions | Function | Function to apply |
| symbols | Symbol | Symbol to apply |
| values | Values | Values |

### 09-Split

### Split By Expression

This node splits the incoming DataFrame into two output DataFrames by applying the conditional logic

#### Input

It accepts a DataFrame as input from the previous Node

#### Type

transform

#### Class

fire.nodes.etl.NodeSplitByExpression

#### Fields

| Name | Title | Description |
| --- | --- | --- |
| conditionExpr | Conditional Expression to split the Data on | Conditional Expression to be used for Splitting the DataFrame into two. DataFrame which matches the condition will go to the lower edge output. The other would go to the higher edge output. |

### SplitByMultipleExpressions

Splits the incoming DataFrame into multiple output DataFrames by applying the conditional logic

#### Input

It accepts a DataFrame as input from the previous Node

#### Type

transform

#### Class

fire.nodes.etl.NodeSplitByMultipleExpressions

#### Fields

| Name | Title | Description |
| --- | --- | --- |
| conditionExpr1 | Conditional Expression 1 to split the Data on | Conditional Expression 1 to be used for Splitting the Dataset |
| conditionExpr2 | Conditional Expression 2 to split the Data on | Conditional Expression 2 to be used for Splitting the Dataset |
| conditionExpr3 | Conditional Expression 3 to split the Data on | Conditional Expression 3 to be used for Splitting the Dataset |
| conditionExpr4 | Conditional Expression 4 to split the Data on | Conditional Expression 4 to be used for Splitting the Dataset |
| conditionExpr5 | Conditional Expression 5 to split the Data on | Conditional Expression 5 to be used for Splitting the Dataset |

### CompareAllColumnsSingleOutput

Compares 2 incoming DataFrames. Outputs 1 DataFrame (A-B) or (B-A) or (A intersection B) based on user's input

#### Type

join

#### Class

fire.nodes.etl.NodeCompareAllColumnsSingleOutput

#### Fields

| Name | Title | Description |
|---|---|---|
| compareOption | Compare Type | Comparision options whether (A-B) or (B-A) or (A intersection B) |

### Compare Specific Columns

Compares 2 incoming DataFrames on specific columns. Outputs 1 DataFrame (A-B) or (B-A) or (A intersection B) based on user's input

#### Type

join

#### Class

fire.nodes.etl.NodeCompareSpecificColumnsSingleOutput

#### Fields

| Name | Title | Description |
|---|---|---|
| columnsToCompare | Columns to Compare | Columns to be used in the comparison |
| compareOption | Compare Type | Comparision options whether (A-B) or (B-A) or (A intersection B) |

### CompareSpecificColumns

Compares 2 incoming DataFrames on specific columns. Outputs 3 DataFrames (A-B), (B-A), (A intersection B)

#### Type

join

#### Class

fire.nodes.etl.NodeCompareSpecificColumns

#### Fields

| Name | Title | Description |
|---|---|---|
| columnsToCompare | Columns to Compare | Columns to be used in the comparison |

### Compare All Columns

Compares 2 incoming DataFrames. Outputs 3 DataFrames (A-B), (B-A), (A intersection B)

#### Type

join

#### Class

fire.nodes.etl.NodeCompareAllColumns

#### Fields

### 11-AddColumn

### Expressions

Expressions

#### Type

transform

#### Class

fire.nodes.etl.NodeExpressions

#### Fields

| Name | Title | Description |
|---|---|---|
| description | Description | Description |
| outputCols | New Columns Name | New Columns Name |
| expressions | Expressions | Expressions |

### AddColumns

This node allows adding new columns with certain values

#### Input

This type of node takes in a DataFrame and transforms it to another DataFrame

#### Output

This node adds the user specified columns to the DataFrame

#### Type

transform

#### Class

fire.nodes.etl.NodeAddColumns

#### Fields

| Name | Title | Description |
|---|---|---|
| addCurrentDateCol | Add Current Date Column | Whether to add the current date as a new column |
| currentDateColName | Current Date Column Name | Name of the new Current Date Column Created |
| addCurrentTimeCol | Add Current Time Column | Whether to add the current time as a new column |
| currentTimeColName | Current Time Column Name | Name of the new Current Time Column Created |
| addConstantStringCol1 | Add Constant String Column | Whether to add a new columns with constant string value |
| constantStringColName1 | Constant String Column Name | Constant String Name |
| constantStringColValue1 | Constant String Column Value | Constant String Value |
| addConstantIntCol1 | Add Constant Integer Column | Whether to add a new columns with constant integer value |
| constantIntColName1 | Constant Integer Column Name | Constant Integer Column Name |
| constantIntColValue1 | Constant Integer Column Value | Constant Integer Value |

### GenerateUID

This node Generates a new column with unique Index/Value for each row in the Dataset for each partition. Each Partition starts a new range.

#### Type

transform

#### Class

fire.nodes.etl.NodeGenerateUID

#### Fields

| Name | Title | Description |
|---|---|---|
| outputCol | UID Column Name | UID column name |

### Hash

This node adds a new Columns which contains the Hash of the specified columns

#### Input

It accepts a DataFrame as input from the previous Node

#### Output

A new column is added to the incoming DataFrame by creating a Hash of the specified input columns.

#### Type

transform

#### Class

fire.nodes.etl.NodeHash

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Columns | Columns to be concatenated |
| hashingAlgorithm | Hashing Algorithm | Hashing Algorithm |
| outputCol | Output Column Name | Column name for Hash |
| bitLength | Bit Length | Bit Length |
| sep | Separator | Separator to be used when concatenating the columns |

### GenerateUUID

This node Generates a Universally Unique ID

#### Input

It accepts a dataframe as input

#### Output

It adds a new column for UUID to the input DataFrame. This new DataFrame is sent to the output

#### Type

transform

### Class

fire.nodes.etl.NodeGenerateUUID

### Fields

| Name | Title | Description |
|---|---|---|
| outputCol | Output Column | Output Column Name |

## CaseWhen

Sets values based on conditions

### Type

transform

### Class

fire.nodes.etl.NodeCaseWhen

### Fields

| Name | Title | Description |
|---|---|---|
| outputCol | Output Column Name | output column name |
| whenConditions | When Condition | When Condition |
| values | Value | Value when this condition is true |
| finallyElse | Else | else |

## ConcatColumns

This node creates a new DataFrame by concatenating the specified columns of the input DataFrame

### Input

It accepts a DataFrame as input from the previous Node

### Output

A new column is added to the incoming DataFrame by concatenating the specified columns. The new DataFrame is sent to the output of this Node.

#### Type

transform

#### Class

fire.nodes.etl.NodeConcatColumns

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Columns | Columns to be concatenated |
| outputCol | Concatenated Column Name | Column name for the concatenated columns |
| sep | Separator | Separator to be used when concatenating the columns |

### ZipWithIndex

This node Generates a new column with unique Index/Value for each row in the Dataset

#### Type

transform

#### Class

fire.nodes.etl.NodeZipWithIndex

#### Fields

| Name | Title | Description |
|---|---|---|
| indexColName | Index Column Name | Index column name |

### 12-CastDataType

### CastToSingleType

This node creates a new DataFrame by casting the specified input columns to a new data type

#### Input

This type of node takes in a DataFrame and transforms it to another DataFrame

### Output

This node casts the data type of columns as specified

### Type

transform

### Class

fire.nodes.etl.NodeCastColumnType

### Fields

| Name | Title | Description |
| --- | --- | --- |
| inputCols | Columns | Columns to be cast to new data type |
| outputColType | New Data Type | New data type(INTEGER, DOUBLE, STRING, LONG, SHORT) |
| replaceExistingCols | Replace Existing Cols | Whether to replace existing columns or create new ones |

## CastToDifferentTypes-2

This node creates a new DataFrame by casting the specified columns into new types

### Input

This type of node takes in a DataFrame and transforms it to another DataFrame

### Output

This node casts the data type of columns as specified

### Type

transform

### Class

fire.nodes.etl.NodeMultiCastColumnType2

#### Fields

| Name | Title | Description |
|---|---|---|
| inputColNames | Columns | Columns |
| outputColTypes | New Data Type | New data type(INTEGER, DOUBLE, STRING, LONG, SHORT) |
| replaceExistingCols | Replace Existing Cols | Whether to replace existing Columns or create New Ones |
| formats | Formats | Formats like yyy-MM-dd used in input & output |

### CastToDifferentTypes-1

This node creates a new DataFrame by casting the specified columns into new types

#### Input

This type of node takes in a DataFrame and transforms it to another DataFrame

#### Output

This node casts the data type of columns as specified

#### Type

transform

#### Class

fire.nodes.etl.NodeMultiCastColumnType

#### Fields

| Name | Title | Description |
|---|---|---|
| inputColNames | Columns | Columns |
| outputColTypes | New Data Type | New data type(INTEGER, DOUBLE, STRING, LONG, SHORT) |
| replaceExistingCols | Replace Existing Cols | Whether to replace existing Columns or create New Ones |

### 06-Math

### MathFunctions

This node performs specified math function on a row

### Input

It accepts a DataFrame as input from the previous Node

### Output

A new column is added which contains the results of applying the Math function on the given column of the input DataFrame

### Type

transform

### Class

fire.nodes.etl.NodeMathFuntions

### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column Name | input column name |
| mathFunction | Math Function | Math Function Name |
| outputCol | Output Column | Output Column Name |
| scale | Scale | Scale to be used when applying the Math Function |

## MathFunctionsMultiple

Math Functions Multiple

### Type

transform

### Class

fire.nodes.etl.NodeMathFunctionsMultiple

#### Fields

| Name | Title | Description |
|---|---|---|
| description | Description | Description |
| inputCols | Columns | Columns |
| functions | Function | Math Function to apply |
| replaceExistingCols | Replace Existing Cols | Replace Existing Columns (true, false) |
| scales | Scale | Scale to be used when applying the Math Function |

### MathExpression

#### Type

transform

#### Class

fire.nodes.etl.NodeMathExpression

#### Fields

| Name | Title | Description |
|---|---|---|
| outputCols | OutPut Column | Output Column Name |
| expressions | Math Expression | Define math expression. |

### 03-DateTime

### DateDifference

This node finds difference between two dates

#### Input

It accepts a DataFrame as input from the previous Node

#### Type

transform

#### Class

fire.nodes.etl.NodeDateDiff

### Fields

| Name | Title | Description |
|---|---|---|
| fromDate | FromDate | From date column name |
| toDate | Todate | To date column name |
| useCurrentDateAsToDateCol | useCurrentDateAsToCol | Current Date As ToDate |
| days | Days | Days difference |
| hours | Hours | Hours difference |
| minutes | Minutes | Minutes difference |
| seconds | Seconds | Seconds difference |

### Details

Calculates difference between 2 given dates. Difference between dates is displayed in days, hours, minutes, and seconds columns.

### Examples

### Format Examples

dd-MM-yy : 30-11-95 to 19-02-18 diff- 8608 days : 206609 hours : 12396574 min : 743794461 : second dd-MM-yyyy : 10-02-1996 to 20-09-2017 diff- 8536 days : 204881 hours : 12292884 min : 737573070 : second MM-dd-yyyy : 19-10-1994 to 06-12-2017 diff- 9015 days : 216377 hours : 12982644 min : 778958670 : second yyyy-MM-dd : 1994-12-25 to 2019-01-16 diff- 8948 days : 214769 hours : 12886164 min : 773169870 : second yyyy-MM-dd HH:mm:ss : 2012-01-31 23:59:59 to 2010-12-30 22:59:59 diff-397 days: 1 hour: 0 minutes : 0 seconds

## TimeFunctions

### Type

transform

### Class

fire.nodes.etl.NodeTimeFunctions

### Fields

| Name | Title | Description |
|---|---|---|
| timeStampCol | TimeStamp Column Name | input column name |
| timeFunctions | Time Functions | Time Functions Name |

### DateTimeFieldExtract

It creates a new DataFrame by extracting Date and Time fields.

#### Input

It takes in a DataFrame as Input

#### Output

Node to extract year/month/dayofmonth/hour/minute/seconad values from TimeStamp

#### Type

transform

#### Class

fire.nodes.etl.NodeDateTimeFieldExtract

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Column | The input column name |
| extractYear | Extract Year | Extract Year |
| extractMonth | Extract Month | Extract Month |
| extractDayOfMonth | Extract Day of Month | Extract Day of Month |
| extractHour | Extract Hour | Extract Hour |
| extractMinute | Extract Minute | Extract Minute |
| extractSecond | Extract Second | Extract Second |
| extractWeekOfYear | Extract WeekOfYear | Extract WeekOfYear |

#### Details

Extracts year, month, day of month, hour, minute, second and week of year in different columns.

### StringToUnixTime

This nodes converts a string to Unix Time

#### Type

transform

#### Class

fire.nodes.etl.NodeStringToUnixTime

#### Fields

| Name | Title | Description |
|---|---|---|
| inputColName | Input Column Name | Input Column Name |
| inputColFormat | Input Column Format | Input Column Format (eg: yyyy-MM-dd HH:mm:ss) |
| outputColName | Output Column Name | Output Column Name |

#### Details

This node converts a string column to unix time (seconds).

#### Examples

#### Format Examples

dd-MM-yy : 31-01-12 dd-MM-yyyy : 31-01-2012 MM-dd-yyyy : 01-31-2012 yyyy-MM-dd : 2012-01-31 yyyy-MM-dd HH:mm:ss : 2012-01-31 23:59:59 yyyy-MM-dd HH:mm:ss.SSS : 2012-01-31 23:59:59.999 yyyy-MM-dd HH:mm:ss.SSSZ : 2012-01-31 23:59:59.999+0100 EEEEE MMMMM yyyy HH:mm:ss.SSSZ : Saturday November 2012 10:45:42.720+0100

Example: Date (string), Format , Unix time (seconds)

> 2003-07-25 , yyy-MM-dd , 1059091200

### StringToDate

This node converts a string column to date using the given date/time format

#### Type

transform

#### Class

fire.nodes.etl.NodeMultiStringToDate

#### Fields

| Name | Title | Description |
|---|---|---|
| inputColNames | Columns | Columns |
| inputColFormats | Input Column Formats | Input Column Formats. eg: yyyy-MM-dd yyyy-MM-dd HH:mm:ss |
| outputColNames | Output Column Names | Output Column Names |
| outputColTypes | New Data Types | New data types (DATE, TIMESTAMP) |

#### Details

This node converts multiple string columns to date/time.

#### Examples

#### Format Examples

dd-MM-yy : 31-01-12 dd-MM-yyyy : 31-01-2012 MM-dd-yyyy : 01-31-2012 yyyy-MM-dd : 2012-01-31 yyyy-MM-dd HH:mm:ss : 2012-01-31 23:59:59 yyyy-MM-dd HH:mm:ss.SSS : 2012-01-31 23:59:59.999 yyyy-MM-dd HH:mm:ss.SSSZ : 2012-01-31 23:59:59.999+0100 EEEEE MMMMM yyyy HH:mm:ss.SSSZ : Saturday November 2012 10:45:42.720+0100

### UnixTimeToString

This node converts Unix Time to String

#### Type

transform

#### Class

fire.nodes.etl.NodeUnixTimeToString

#### Fields

| Name | Title | Description |
|---|---|---|
| inputColName | Input Column Name | input column name |
| outputColName | Output Column Name | Output Column Name |
| outputColFormat | Output Column Format | Output Column Format (eg: yyyy-MM-dd HH:mm:ss) |

### Details

This node converts unix time (seconds) to string type.

### Examples

### Format Examples

dd-MM-yy : 31-01-12 dd-MM-yyyy : 31-01-2012 MM-dd-yyyy : 01-31-2012 yyyy-MM-dd : 2012-01-31 yyyy-MM-dd HH:mm:ss : 2012-01-31 23:59:59 yyyy-MM-dd HH:mm:ss.SSS : 2012-01-31 23:59:59.999 yyyy-MM-dd HH:mm:ss.SSSZ : 2012-01-31 23:59:59.999+0100 EEEEE MMMMM yyyy HH:mm:ss.SSSZ : Saturday November 2012 10:45:42.720+0100

Example: select an input column (long type), output column name and desired output column format. It will add one more column in string format.

If you input a date format like dd-MM-yy. It will add one column having value like 31-01-12.

## DateToString

This node converts a date/time column to string with given format

### Type

transform

### Class

fire.nodes.etl.NodeMultiDateToString

### Fields

| Name | Title | Description |
|---|---|---|
| inputColNames | Input Column Name | Input Column Name |
| outputColFormats | Output Column Formats | Output Column Formats. eg: yyyy-MM-dd yyyy-MM-dd HH:mm:ss |
| outputColNames | Output Column Names | Output Column Names |

### Details

This node converts date/time column to string type with given format.

#### Examples

#### Format Examples

dd-MM-yy : 31-01-12 dd-MM-yyyy : 31-01-2012 MM-dd-yyyy : 01-31-2012 yyyy-MM-dd : 2012-01-31 yyyy-MM-dd HH:mm:ss : 2012-01-31 23:59:59 yyyy-MM-dd HH:mm:ss.SSS : 2012-01-31 23:59:59.999 yyyy-MM-dd HH:mm:ss.SSSZ : 2012-01-31 23:59:59.999+0100 EEEEE MMMMM yyyy HH:mm:ss.SSSZ : Saturday November 2012 10:45:42.720+0100

### 07-String

#### StringFunctions

This node performs specified String function on a row

#### Input

It accepts a DataFrame as input from the previous Node

#### Type

transform

#### Class

fire.nodes.etl.NodeStringFunctions

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Input Column Name | input column name |
| stringFunction | String Function | String Function Name |
| replaceExistingCols | ReplaceExistingCols | replaceExistingCols |

#### StringFunctionsMultiple

String Functions Multiple

#### Type

transform

#### Class

fire.nodes.etl.NodeStringFunctionsMultiple

### Fields

| Name | Title | Description |
|---|---|---|
| description | Description | Description |
| inputCols | Columns | Columns |
| functions | Function | String Function to apply |
| replaceExistingCols | Replace Existing Cols | Replace Existing Columns (true or false) |

### TextCaseTransformer

This node converts text to upper or lower case

### Input

It accepts a DataFrame as input from the previous Node

### Type

transform

### Class

fire.nodes.etl.NodeTextCaseTransformer

### Fields

| Name | Title | Description |
|---|---|---|
| inputCol | Input Column Name | input column name |
| mode | Text Case Type | input to convert text to upper or lower case |
| outputCol | Output Column | Output Column |

## 05-DataCleaning

### DataWrangling

This node creates a new DataFrame by applying each of the Rules specified

### Input

It takes in a DataFrame as Input

### Output

It creates the output DataFrame by applying the data wrangling rules provided

### Type

transform

### Class

fire.nodes.etl.NodeDataWrangling

### Fields

| Name | Title | Description |
|---|---|---|
| rules | Rules | Rules to be applied on column and rows |

### Details

Rename one column to another rename col:c1 to c2;

Drop Column drop col:col4

Delete columns with some condition delete col:col3 > 44

Substring col:col2 0,3 get substring between 0 and 3rd column from the column col2

Trim Values : Removes leading and trailing whitespace from a string value.

set col:Name value: trim(Name)

Sets the new value of Name column to be trim(Name)

## RemoveUnwantedCharactersMult

This node removes unwanted characters

### Input

It accepts a DataFrame as input from the previous Node

### Type

transform

### Class

fire.nodes.etl.NodeRemoveUnwantedCharactersMultiple

### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Input Columns | Input columns |
| removeWhitespaces | Remove Whitespaces | Removes white space |
| removeLetters | Remove Letters | Removes letters |
| removeDigits | Remove Digits | Removes digits |
| removeSigns | Remove Signs | Removes signs |
| removeCommas | Remove Commas | Removes commas |

## ImputingWithMedian

Imputing with median

### Type

transform

### Class

fire.nodes.ml.NodeReplaceMissingValueWithMedian

### Fields

| Name | Title | Description |
|---|---|---|
| colNames | Input Columns | Input column of type - all numeric for median impute |

## DropRowsWithNull

This node creates a new DataFrame by dropping rows containing null values

### Input

It accepts DataFrame as input from the previous Node

### Output

This node drops rows containing null values

### Type

transform

#### Class

fire.nodes.etl.NodeDropRowsWithNull

#### Fields

### DropDuplicateRows

1>When user don't select any column, returns a new Dataset that contains only the unique rows from this Dataset. 2> Returns a new Dataset with duplicate rows removed, considering only the subset of columns.

#### Type

transform

#### Class

fire.nodes.etl.NodeDropDuplicateRows

#### Fields

| Name | Title | Description |
|---|---|---|
| colNames | Columns | Columns to be used in determining if any two rows are duplication. No columns indicate to use all the available columns. |

### FindAndReplaceUsingRegexMultiple

This node finds and replaces text in a column containing string

#### Input

It accepts a DataFrame as input from the previous Node

#### Type

transform

#### Class

fire.nodes.etl.NodeFindAndReplaceUsingRegexMultiple

### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Input Columns | Columns on which to apply Regex |
| searchPatterns | Find | Enter Search Pattern |
| replacePatterns | Replace | Enter replacement Value |

### FindAndReplaceUsingRegex

This node finds and replaces text in a column containing string

### Input

It accepts a DataFrame as input from the previous Node

### Type

transform

### Class

fire.nodes.etl.NodeFindAndReplaceUsingRegex

### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Input Columns | Columns on which to apply Regex |
| searchPattern | Find | Enter Search Pattern |
| replacePattern | Replace | Enter replacement Value |

### ImputingWithConstant

It imputes missing value with constant value. It fills missing values (None) in selected columns with given constant value for the corresponding column, in the incoming DataFrame.

### Type

transform

### Class

fire.nodes.ml.NodeReplaceMissingValueWithConstant

#### Fields

| Name | Title | Description |
|---|---|---|
| colNames | Columns | Columns to be processed for missing values |
| constants | Constants | Missing value will be replaced with constant |

### ImputingWithMeanValue

Imputing the continuous variables by mean.

#### Type

transform

#### Class

fire.nodes.ml.NodeReplaceMissingValueWithMean

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Column Names | Columns type should be continuous |

### RemoveDuplicateRows

This node take an array of fields, compare rows on those fields. If they full match then its a match. From the matches it would randomly take one row and drop the rest.

#### Input

It accepts a DataFrame as input from the previous Node

#### Output

The output Dataframe is the same as the input Dataframe with the duplicate rows removed

#### Type

transform

#### Class

fire.nodes.etl.NodeRemoveDuplicateRows

### Fields

| Name | Title | Description |
|---|---|---|
| order | Order | Whether to take the first or last matching record when removing duplicates |
| inputCols | Columns | The columns to be selected for match |

## Dedup

This node is used for problems like entity resolution or data matching. Entity resolution or Data matching is the problem of finding and linking different mentions of the same entity in a single data source or across multiple data sources.

### Input

It takes in a DataFrame as input

### Output

Dataframe with confidence score field and other selected scores for entities

### Type

transform

### Class

fire.nodes.ml.NodeDedup

### Fields

| Name | Title | Description |
|---|---|---|
| confidenceScore | Confidence Score | Confidence Score |
| lhsCols | LHS Variables | LHS columns for matching |
| rhsCols | RHS Variables | RHS columns for matching |
| matchingAlgorithms | Algorithm to use | Algorithm to use for matching |
| matchingWeights | Weights | Weights for matches |
| outputCols | Output Column | Output Column |

### Details

### Levenstein

The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character.

How many char you change to make two strings equal.

### JaroWinker

Jaro–Winkler distance for two strings is, the more similar the strings are. The Jaro–Winkler distance metric is designed and best suited for short strings such as person names. The score is normalized such that 0 equates to no similarity and 1 is an exact match.

Good for short words, typos and nikename.

Fullmatch

---

Fullmatch distance for two strings is, how two strings are match exactly. The score is assigned such that 1 is for exact match and 0 is for not match.

### Jaccard

The Jaccard similarity measures similarity between finite sample sets, and is defined as the cardinality of the intersection of sets divided by the cardinality of the union of the sample sets. Suppose you want to find jaccard similarity between two sets A and B it is the ration of cardinality of A  B and A  B.

Sparkflows provide default 3-gram Jaccard similarity measures.

Longest common subsequences(LCS): LCS distance between strings s1 and s2, computed as **|s1|** +|s2| - 2 * **|LCSfunction(s1, s2)|** and distance is normalized between 0 and 1.

LCSfunction returns the length of Longest Common Subsequence (LCS) between strings s1 and s2.

### Notional distance

Notional distance between two numbers X and Y, computed as abs(X - Y) / abs(x) + abs(Y).

### Date Difference

Date Difference gives number of days between two dates(yyyy-MM-dd).

### RemoveUnwantedCharacters

This node removes unwanted characters

### Input

It accepts a DataFrame as input from the previous Node

#### Type

transform

#### Class

fire.nodes.etl.NodeRemoveUnwantedCharacters

#### Fields

| Name | Title | Description |
|---|---|---|
| inputCols | Input Columns | Input columns |
| removeWhitespaces | Remove Whitespaces | Removes white space |
| removeLetters | Remove Letters | Removes letters |
| removeDigits | Remove Digits | Removes digits |
| removeSigns | Remove Signs | Removes signs |
| removeCommas | Remove Commas | Removes commas |

### ImputingWithModeValue

Imputing with most frequently observed value. It fills missing values (None) in selected columns with most frequently observed value in the corresponding column, in the incoming DataFrame.

#### Type

transform

#### Class

fire.nodes.ml.NodeReplaceMissingValueWithMode

#### Fields

| Name | Title | Description |
|---|---|---|
| colNames | Columns | Columns to be processed for imputing the missing values. |

## 24.1.12 04-DataValidation

### ValidateFieldsAdvanced

Validation Multiple Node

#### Type

transform

#### Class

fire.nodes.etl.NodeValidationMultiple

#### Fields

| Name | Title | Description |
|---|---|---|
| description | Description | Validations being Performed |
| measureValue | Validation Successful if Percent Good Records >= | Condition for Validation Passing |
| inputCols | Columns | Columns |
| functions1 | Function | Validation Function to apply |
| values1 | Values | Values |
| conditions1 | Condition | Validation Condition to apply |
| functions2 | Function | Validation Function to apply |
| values2 | Values | Values |
| conditions2 | Condition | Validation Condition to apply |
| functions3 | Function | Validation Function to apply |
| values3 | Values | Values |

### CompareDatasets

Validate the input datasets

#### Type

join

#### Class

fire.nodes.validation.NodeCompareDatasets

#### Fields

### ValidateAddress

This node validate the USA address

#### Input

It accepts a DataFrame as input from the previous Node

#### Output

A new column isValidAddress is added which contains valid or inValid values

#### Type

transform

#### Class

fire.nodes.etl.NodeValidateAddress

#### Fields

| Name | Title | Description |
|---|---|---|
| inputColName | Input Column Name | input column name |

#### ValidateFieldsSimple

Validation Node

#### Type

transform

#### Class

fire.nodes.etl.NodeValidation

#### Fields

| Name | Title | Description |
|---|---|---|
| description | Description | Validations being Performed |
| inputCols | Columns | Columns |
| functions | Function | Validation Function to apply |
| values | Values | Values |

### 24.1.13 CustomProcessors

#### pyspark

#### ScoreCard_Binning

#### Type

transform

#### Class

fire.nodes.etl.NodeCustomPySpark_dd281630-bf8f-4e04-8526-1cb555871c46

#### Fields

### 24.1.14 17-Documentation

#### StickyNote

Allows capturing Notes on the Workflow

#### Type

sticky

#### Class

fire.nodes.doc.NodeStickyNote

#### Fields

| Name | Title | Description |
|---|---|---|
| bgColor | Bg Color | Background of note |
| width | Width | Width of note |
| height | Height | Height of note |
| comment | Comment | Comments for the Workflow |

#### Notes

Allows capturing Notes on the Workflow

#### Type

doc

#### Class

fire.nodes.doc.NodeDocLarge

#### Fields

| Name | Title | Description |
|---|---|---|
| comment | Comment | Comments for the Workflow |

### 24.1.15 12-ML-H2O

#### H2OWord2Vec

H2O Word2Vec

#### Input

It takes in a DataFrame as input

#### Type

transform

#### Class

fire.nodes.h2o.NodeH2OWord2vec

#### Fields

| Name | Title | Description |
|---|---|---|
| min_word_freq | Min Word Freq | Specify an integer for the minimum word frequency. Word2vec will discard words that appear less than this number of times. |
| vec_size | Vec Size | Specify the size of word vectors. |
| window_size | Window Size | This specifies the size of the context window around a given word. |
| epochs | Epochs | Specify the number of training iterations to run. |
| init_learning_rate | Init Learning Rate | Set the starting learning rate. |
| sent_sample_rate | Sent Sample Rate | Set the threshold for the occurrence of words. Those words that appear with higher frequency in the training data will be randomly down-sampled. An ideal range for this option 0, 1e-5. |
| aggregateMethod | AggregateMethod | Specifies how to aggregate sequences of words. |

#### Details

The Word2vec algorithm takes a text corpus as an input and produces the word vectors as output. The algorithm first creates a vocabulary from the training text data and then learns vector representations of the words.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/word2vec.html#

### H2OScore

#### Type

join

#### Class

fire.nodes.h2o.NodeH2OScore

#### Fields

### H2OModelSave

Saves an H2O binary ML model

#### Type

ml-modelsave

#### Class

fire.nodes.h2o.NodeH2OModelSave

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Absolute Path for saving the H2O Mojo |

### H2OPCA

H2O PCA

#### Input

It takes in a DataFrame as input

#### Type

transform

#### Class

fire.nodes.h2o.NodeH2OPCA

#### Fields

#### Details

Principal Components Analysis (PCA) is closely related to Principal Components Regression. The algorithm is carried out on a set of possibly collinear features and performs a transformation to produce a new set of uncorrelated features.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/pca.html

### H2OGLM

H2O GLM

#### Input

It takes in a DataFrame as input

#### Type

transform

#### Class

fire.nodes.h2o.NodeH2OGlm

#### Fields

#### Details

Generalized Linear Models (GLM) estimate regression models for outcomes following exponential distributions. In addition to the Gaussian (i.e. normal) distribution, these include Poisson, binomial, and gamma distributions. Each serves a different purpose, and depending on distribution and link function choice, can be used either for prediction or classification.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/glm.html

### H2OScore

#### Type

ml-predict

#### Class

fire.nodes.h2o.NodeH2OScore

#### Fields

| Name | Title | Description |
|---|---|---|
| isTestData | isTestData | To enable the test metrics. |

### H2OMojoLoad

Loads an H2O Mojo ML model

#### Type

ml-modelload

#### Class

fire.nodes.h2o.NodeH2OMojoLoad

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Absolute Path for loading the H2O Mojo |

### H2OXGBoostScore

#### Type

ml-predict

#### Class

fire.nodes.h2o.NodeH2OXGBoostScore

#### Fields

| Name | Title | Description |
|---|---|---|
| isTestData | isTestData | To enable the test metrics. |

### H2O Model Load

This node load the H2O model.

#### Type

ml-modelload

#### Class

fire.nodes.h2o.NodeH2OModelLoad

#### Fields

### H2OXGBoostWithGridSearch

H2O XGBoost

#### Input

It takes in a DataFrame as input

#### Type

join

#### Class

fire.nodes.h2o.node_h2oxgboost_gridsearch

#### Fields

### H2OXGBoost

H2O XGBoost

#### Input

It takes in a DataFrame as input

### Type

join

### Class

fire.nodes.h2o.node_h2oxgboost

### Fields

| Name | Title | Description |
|---|---|---|
| responseCol | Response Column | |
| featureCols | Feature Columns | Specify the column or columns to be included for feature. |
| ntrees | NTrees | Specify the number of trees to build |
| tree_method | Tree Method | Specify the construction tree method to use. |
| grow_policy | Grow Policy | |
| max_depth | Max Depth | Specify the maximum tree depth (Setting this value to 0 specifies no limit) |
| max_leaves | Max Leaves | Specify the maximum number of leaves to include each tree |
| col_sample_rate_per_tree | Col Sample Rate Per Tree | |
| sample_rate | Sample rate | Specify the row sampling rate (x-axis). (Note that this method is sample without replacement) |
| learn_rate | Learn Rate | Specify the learning rate (The range is 0.0 to 1.0) |
| stopping_rounds | Stopping Rounds | |
| stopping_metric | Stopping Metric | Specify the construction tree method to use. |
| seed | Seed | |

## H2OXGBoost

H2O XGBoost

### Input

It takes in a DataFrame as input

### Type

transform

### Class

fire.nodes.h2o.NodeH2OXGBoost

#### Fields

#### Details

XGBoost is a supervised learning algorithm that implements a process called boosting to yield accurate models. Boosting refers to the ensemble learning technique of building many models sequentially, with each new model attempting to correct for the deficiencies in the previous model.

More details are available at : https://h2o-release.s3.amazonaws.com/h2o/rel-weierstrass/2/docs-website/h2o-docs/data-science/xgboost.html

### H2O Model Save

This node saves the H2O model at the specified path.

#### Input

It takes in a Model and DataFrame as input.

#### Output

The incoming dataframe is passed to the output.

#### Type

ml-modelsave

#### Class

fire.nodes.h2o.NodeH2OModelSave

#### Fields

### H2ONeuralNetwork

H2O Deep Learning/Neural Network

#### Input

It takes in a DataFrame as input

#### Type

transform

#### Class

fire.nodes.h2o.NodeH2ONeuralNetwork

#### Fields

#### Details

H2O's Deep Learning is based on a multi-layer feedforward artificial neural network that is trained with stochastic gradient descent using back-propagation. The network can contain a large number of hidden layers consisting of neurons with tanh, rectifier, and maxout activation functions.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/deep-learning.html

### H2ONaiveBayes

H2O Naive Bayes

#### Input

It takes in a DataFrame as input

#### Type

transform

#### Class

fire.nodes.h2o.NodeH2ONaiveBayes

#### Fields

#### Details

Naïve Bayes is a classification algorithm that relies on strong assumptions of the independence of covariates in applying Bayes Theorem. The Naïve Bayes classifier assumes independence between predictor variables conditional on the response, and a Gaussian distribution of numeric predictors with mean and standard deviation computed from the training dataset.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/naive-bayes.html

### H2OGLRM

H2O GLRM

#### Input

It takes in a DataFrame as input

#### Type

transform

#### Class

fire.nodes.h2o.NodeH2OGlrm

#### Fields

#### Details

Generalized Low Rank Models (GLRM) is an algorithm for dimensionality reduction of a dataset. It is a general, parallelized optimization algorithm that applies to a variety of loss and regularization functions. Categorical columns are handled by expansion into 0/1 indicator columns for each level. With this approach, GLRM is useful for reconstructing missing values and identifying important features in heterogeneous data.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/glrm.html

### H2OGBM

H2O GBM

#### Input

It takes in a DataFrame as input

#### Type

transform

#### Class

fire.nodes.h2o.NodeH2OGbm

#### Fields

#### Details

Gradient Boosting Machine (for Regression and Classification) is a forward learning ensemble method. The guiding heuristic is that good predictive results can be obtained through increasingly refined approximations. H2O's GBM sequentially builds regression trees on all the features of the dataset in a fully distributed way - each tree is built in parallel.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/gbm.html

### H2OKMeans

H2O KMeans

#### Input

It takes in a DataFrame as input

#### Type

ml-estimator

#### Class

fire.nodes.h2o.NodeH2OKMeans

#### Fields

#### Details

K-Means falls in the general category of clustering algorithms. Clustering is a form of unsupervised learning that tries to find structures in the data without using any labels or target values. Clustering partitions a set of observations into separate groupings such that an observation in a given group is more similar to another observation in the same group than to another observation in a different group.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/k-means.html

### H2OIsolationForest

Isolation Forest is similar in principle to Random Forest and is built on the basis of decision trees.

#### Input

It takes in a DataFrame as input

#### Type

transform

#### Class

fire.nodes.h2o.NodeH2OIsolationForest

#### Fields

#### Details

Isolation Forest is similar in principle to Random Forest and is built on the basis of decision trees. Isolation Forest, however, identifies anomalies or outliers rather than profiling normal data points. Isolation Forest isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that selected feature. This split depends on how long it takes to separate the points.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/if.html

### H2ODRF

H2O DRF

#### Input

It takes in a DataFrame as input

#### Type

transform

#### Class

fire.nodes.h2o.NodeH2ODrf

#### Fields

#### Details

Distributed Random Forest (DRF) is a powerful classification and regression tool. When given a set of data, DRF generates a forest of classification or regression trees, rather than a single classification or regression tree. Each of these trees is a weak learner built on a subset of rows and columns. More trees will reduce the variance. Both classification and regression take the average prediction over all of their trees to make a final prediction, whether predicting for a class or numeric value.

More details are available at : http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html

### H2OMojoSave

Saves an H2O MOJO ML model

#### Type

ml-modelsave

#### Class

fire.nodes.h2o.NodeH2OMojoSave

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path for saving the H2O Mojo |

### H2OModelLoad

Loads an H2O binary ML model

#### Type

ml-modelload

#### Class

fire.nodes.h2o.NodeH2OModelLoad

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path for loading the H2O Mojo |

## 24.1.16 13-ML-AWSSagemaker

### KMeansSageMakerEstimator

#### Type

ml-estimator

#### Class

fire.nodes.sagemaker.NodeKMeansSageMakerEstimator

#### Fields

| Name | Title | Description |
|---|---|---|
| roleArn | Role Arn | Role arn to use sagemaker |
| trainingInstanceType | Training Instance Type | InstanceType for training |
| trainingInstanceCount | Training Instance Count | Number of Instance for training |
| endpointInstanceType | Endpoint Instance Type | InstanceType for Endpoint |
| endpointInitialInstanceCount | Endpoint Initial Instance Count | Number of Instance for Endpoint |
| k | K | The number of clusters to create. |
| featureDim | Feature Dim | The number of dimensions in dataset |

### XGBoostSageMakerEstimator

#### Type

ml-estimator

#### Class

fire.nodes.sagemaker.NodeXGBoostSageMakerEstimator

#### Fields

| Name | Title | Description |
|---|---|---|
| roleArn | Role Arn | Role arn to use sagemaker |
| trainingInstanceType | Training Instance Type | InstanceType for training |
| trainingInstanceCount | Training Instance Count | Number of Instance for training |
| endpointInstanceType | Endpoint Instance Type | InstanceType for Endpoint |
| endpointInitialInstanceCount | Endpoint Initial Instance Count | Number of Instance for Endpoint |
| booster | Booster | Select the type of model to run at each iteration. It has 2 options: gbtree: tree-based models & gblinear: linear models |
| silent | Silent | Silent mode is activated is set to 1, i.e. no running messages will be printed |
| nthread | NThread | If you wish to run on all cores, value should not be entered and algorithm will detect automatically |
| objective | Objective | This defines the loss function to be minimized |
| numTrees | Num Trees | The number of rounds for boosting |
| numClasses | Num Classes | For Objective: multi:softmax, you also need to set an additional num_class (number of classes) parameter defining the number of unique classes |
| seed | Seed | Can be used for generating reproducible results and also for parameter tuning |

### PCASageMakerEstimator

#### Type

ml-estimator

#### Class

fire.nodes.sagemaker.NodePCASageMakerEstimator

#### Fields

| Name | Title | Description |
|---|---|---|
| roleArn | Role Arn | Role arn to use sagemaker |
| trainingInstanceType | Training Instance Type | InstanceType for training |
| trainingInstanceCount | Training Instance Count | Number of Instance for training |
| endpointInstanceType | Endpoint Instance Type | InstanceType for Endpoint |
| endpointInitialInstanceCount | Endpoint Initial Instance Count | Number of Instance for Endpoint |
| numComponents | Num Components | The number of principal components to find. |
| featureDim | Feature Dim | The number of dimensions in dataset |

### SageMakerLinearLearnerBinaryClassifier

#### Type

ml-estimator

#### Class

fire.nodes.sagemaker.NodeLinearLearnerBinaryClassifier

#### Fields

| Name | Title | Description |
|---|---|---|
| roleArn | Role Arn | Role arn to use sagemaker |
| trainingInstanceType | Training Instance Type | InstanceType for training |
| trainingInstanceCount | Training Instance Count | Number of Instance for training |
| endpointInstanceType | Endpoint Instance Type | InstanceType for Endpoint |
| endpointInitialInstanceCount | Endpoint Initial Instance Count | Number of Instance for Endpoint |

### SageMakerLinearLearnerRegressor

#### Type

ml-estimator

#### Class

fire.nodes.sagemaker.NodeLinearLearnerRegressor

#### Fields

| Name | Title | Description |
|---|---|---|
| roleArn | Role Arn | Role arn to use sagemaker |
| trainingInstanceType | Training Instance Type | InstanceType for training |
| trainingInstanceCount | Training Instance Count | Number of Instance for training |
| endpointInstanceType | Endpoint Instance-Type | InstanceType for Endpoint |
| endpointInitialInstanceCount | Endpoint Initial Instance Count | Number of Instance for Endpoint |

### SaveSageMakerFormat

Saves the DataFrame into the specified location in Sagemaker Format

#### Type

transform

#### Class

fire.nodes.sagemaker.NodeSaveSagemaker

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path where to save the Sagemaker files |
| saveMode | Save Mode | Whether to Append, Overwrite or Error if the path Exists |
| labelColumnName | Label Column Name | label column name |
| featuresColumnName | Features Column Name | features column name |

## 24.1.17 14-ML-Sklearn

### SklearnPredict

Predict node takes in a DataFrame and Model and makes predictions

#### Input

It takes in a DataFrame and Model as input

#### Output

A new column containing the predictions is added to the input DataFrame

#### Type

ml-predict

#### Class

fire.nodes.sklearn.NodeSklearnPredict

#### Fields

### SklearnRegressionEvaluator

Evaluator for regression, which expects two input columns: prediction and label.

#### Input

It takes in a DataFrame as input

#### Output

The incoming DataFrame is passed to the output

#### Type

transform

#### Class

fire.nodes.sklearn.NodeSklearnRegressionEvaluator

#### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Label Column | The label column for model fitting. |
| predictCol | Prediction Column | The prediction column. |

### Sklearn Model Load

This node load the Sklearn model stored in the pickel file.

#### Type

ml-modelload

#### Class

fire.nodes.sklearn.NodeModelLoad

#### Fields

### CustomMetrics

#### Type

transformer

#### Class

fire.nodes.sklearn.NodeCustomMetrics

#### Fields

| Name | Title | Description |
|---|---|---|
| actualCol | ActualCol | |
| predictedCol | PredictedCol | |
| aggregatedAt | AggregatedAt | |
| metricsType | metricsType | Window Function Name |

### SkLearnRidgeRegression

#### Type

ml-estimator

#### Class

fire.nodes.sklearn.NodeSklearnRidgeRegression

#### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Target Column | The label column for model fitting |
| alpha | Alpha | |
| fitintercept | Fitintercept | |
| normalize | Normalize | |
| maxiter | Maxiter | |
| tol | Tolerence | |
| solver | Solver | |
| randomstate | randomstate | Random state |

### SklearnRandomForestClassifier

#### Type

ml-estimator

#### Class

fire.nodes.sklearn.NodeSklearnRandomForestClassifier

#### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Target Column | The label column for model fitting |
| n_estimators | NEstimators | |
| criterion | Criterion | |
| max_depth | MaxDepth | Default value is None i.e -1 |
| min_samples_split | MinSamplesSplit | |
| min_samples_leaf | MinSamplesLeaf | |
| min_weight_fraction_leaf | MinWeightFractionLeaf | |
| max_features | MaxFeatures | |
| max_leaf_nodes | MaxLeafNodes | Default value is None i.e -1 |
| min_impurity_decrease | MinImpurityDecrease | |
| min_impurity_split | MinImpuritySplit | |
| bootstrap | Bootstrap | |
| oob_score | OobScore | |
| random_state | RandomState | Default value is None i.e -1 |
| warm_start | WarmStart | |

### SklearnRandomForestRegression

#### Type

ml-estimator

### Class

fire.nodes.sklearn.NodeSklearnRandomForestRegression

### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Target Column | The label column for model fitting |
| n_estimators | NEstimators | |
| criterion | Criterion | |
| max_depth | MaxDepth | Default value is None i.e -1 |
| min_samples_split | MinSamplesSplit | |
| min_samples_leaf | MinSamplesLeaf | |
| min_weight_fraction_leaf | MinWeightFractionLeaf | |
| max_features | MaxFeatures | |
| max_leaf_nodes | MaxLeafNodes | Default value is None i.e -1 |
| min_impurity_decrease | MinImpurityDecrease | |
| min_impurity_split | MinImpuritySplit | |
| bootstrap | Bootstrap | |
| oob_score | OobScore | |
| random_state | RandomState | Default value is None i.e -1 |
| warm_start | WarmStart | |

## SklearnGradientBoostingRegression

### Type

ml-estimator

### Class

fire.nodes.sklearn.NodeSklearnGradientBoostingRegressor

### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Target Column | The label column for model fitting |
| loss | Loss | |
| learning_rate | LearningRate | |
| n_estimators | NEstimators | |
| subsample | Subsample | |
| criterion | Criterion | |
| min_samples_split | MinSamplesSplit | |
| min_samples_leaf | MinSamplesLeaf | |
| min_weight_fraction_leaf | MinWeightFractionLeaf | |
| max_depth | MaxDepth | Default value is None i.e -1 |
| min_impurity_decrease | MinImpurityDecrease | |
| min_impurity_split | MinImpuritySplit | |
| random_state | RandomState | Default value is None i.e -1 |
| max_features | MaxFeatures | |
| alpha | Alpha | |
| verbose | Verbose | |
| max_leaf_nodes | MaxLeafNodes | Default value is None i.e -1 |
| warm_start | WarmStart | |
| presort | Presort | |
| validation_fraction | ValidationFraction | |
| n_iter_no_change | NIterNoChange | Default value is None i.e -1 |
| tol | Tol | |

### SklearnGradientBoostingClassifier

### Type

ml-estimator

### Class

fire.nodes.sklearn.NodeSklearnGradientBoostingClassifier

### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Target Column | The label column for model fitting |
| loss | Loss | |
| learning_rate | LearningRate | |
| n_estimators | NEstimators | |
| subsample | Subsample | |
| criterion | Criterion | |
| min_samples_split | MinSamplesSplit | |
| min_samples_leaf | MinSamplesLeaf | |
| min_weight_fraction_leaf | MinWeightFractionLeaf | |
| max_depth | MaxDepth | |
| min_impurity_decrease | MinImpurityDecrease | |
| min_impurity_split | MinImpuritySplit | |
| random_state | RandomState | Default value is None i.e -1 |
| max_features | MaxFeatures | |
| verbose | Verbose | |
| max_leaf_nodes | MaxLeafNodes | Default value is None i.e -1 |
| warm_start | WarmStart | |
| presort | Presort | |
| validation_fraction | ValidationFraction | |
| n_iter_no_change | NIterNoChange | Default value is None i.e -1 |
| tol | Tol | |

### SkLearnLassoRegression

### Type

ml-estimator

### Class

fire.nodes.sklearn.NodeSklearnLassoRegression

#### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Target Column | The label column for model fitting |
| alpha | Alpha | |
| fit_intercept | Fitintercept | |
| normalize | Normalize | |
| precompute | Precompute | |
| max_iter | Maxiter | |
| tol | Tol | |
| warm_start | WarmStart | |
| positive | Positive | |
| random_state | RandomState | Default value is None i.e -1 |
| selection | Selection | |

### SklearnLogisticRegression

#### Type

ml-estimator

#### Class

fire.nodes.sklearn.NodeSklearnLogisticRegression

#### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Target Column | The label column for model fitting |
| penalty | Penalty | |
| dual | Dual | |
| tol | Tol | |
| C | C | |
| fit_intercept | Fitintercept | |
| intercept_scaling | InterceptScaling | |
| class_weight | ClassWeight | |
| random_state | RandomState | |
| solver | Solver | |
| max_iter | Maxiter | |
| multi_class | MultiClass | |
| verbose | Verbose | |
| warm_start | WarmStart | |
| l1_ratio | L1Ratio | |

### Sklearn Model Save

This node saves the Sklearn model generated at the specified path in pickle file.

#### Input

It takes in a Model and DataFrame as input.

#### Output

The incoming dataframe is passed to the output.

#### Type

ml-modelsave

#### Class

fire.nodes.sklearn.NodeModelSave

#### Fields

### Sklearn Model Load From S3

This node load the Sklearn model stored in the pickel format in S3.

#### Input

It takes in a Model and DataFrame as input.

#### Output

The incoming dataframe is passed to the output.

#### Type

ml-modelsave

#### Class

fire.nodes.sklearn.NodeSklearnModelLoadFromS3

#### Fields

### SklearnClassificationEvaluator

Evaluator for classification, which expects two input columns: prediction and label.

### Input

It takes in a DataFrame as input

### Output

The incoming DataFrame is passed to the output

### Type

transform

### Class

fire.nodes.sklearn.NodeSklearnClassificationEvaluator

### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Label Column | The label column for model fitting. |
| predictCol | Prediction Column | The prediction column. |

## Sklearn Model Save To S3

This node saves the Sklearn model generated at the specified path in S3 in pickle format.

### Input

It takes in a Model and DataFrame as input.

### Output

The incoming dataframe is passed to the output.

### Type

ml-modelsave

### Class

fire.nodes.sklearn.NodeSklearnModelSaveToS3

#### Fields

#### CategoryEncoders

##### Type

ml-transformer

##### Class

fire.nodes.sklearn.NodeCategoryEncoders

##### Fields

| Name | Title | Description |
|---|---|---|
| category_features_column | Category Features Column | |

### 24.1.18 08-Group

#### GroupBy

Grouper Node

##### Type

transform

##### Class

fire.nodes.etl.NodeGroupBy

##### Fields

#### Cube

Cube Node generates a result set that shows aggregates for all combinations of values in the selected columns.

##### Type

transform

#### Class

fire.nodes.etl.NodeCube

#### Fields

| Name | Title | Description |
|---|---|---|
| cubeCols | Cube Columns | |
| aggregateCols | Aggregate Columns | Aggregate Columns |
| aggregateOperations | Aggregate Operation to use | Aggregate Operation |

### Rollup

Rollup Node generates a result set that shows aggregates for a hierarchy of values in the selected columns.

#### Type

transform

#### Class

fire.nodes.etl.NodeRollup

#### Fields

| Name | Title | Description |
|---|---|---|
| rollupCols | Rollup Columns | |
| aggregateCols | Aggregate Columns | Aggregate Columns |
| aggregateOperations | Aggregate Operation to use | Aggregate Operation |

### PivotBy

Pivot Node

#### Type

transform

#### Class

fire.nodes.etl.NodePivotBy

#### Fields

### 24.1.19 06-Code

#### SQLExecuter

This node runs the given SQL query

##### Input

This type of node takes the sql query of any statement type

##### Output

This node execute the given SQL query

##### Type

dataset

##### Class

fire.nodes.runrdbmssql.NodeSqlExecuter

##### Fields

| Name | Title | Description |
|---|---|---|
| url | Db Url | Url of SQL |
| driver | driver class name | driver class name for SQL |
| user name | User Name | User name of SQL |
| password | password | password of SQL |
| statementType | Statement Type | statementType of SQL |
| query | query | write query to wxecute |

#### PipePython2

This node runs any given Python code. It pipes the incoming DataFrame through pipe to the Python Script. Output back to Spark has to be written out using print.

##### Input

It pipes the incoming DataFrame through pipe to the Python Script. It also passes the Schema of the DataFrame to the Python script through the command line argument - argv[1]

### Output

Output back to Spark has to be written out using print.

### Type

transform

### Class

fire.nodes.etl.NodePipePython2

### Fields

| Name | Title | Description |
|---|---|---|
| codeHeader | Pipe Header Code | Header part of the Python code to be run. It receives each record as a string |
| codeBody | Pipe Body Code | Body part of the Python code to be run. |
| codeFooter | Pipe Footer Code | Footer part of the Python code to be run. It should write out each resulting record back as a string. |
| outputColNames | Output Column Names | Output Schema of Pipe Python Processor |
| outputColTypes | Output Column Types | Data Type of the Output Columns |
| outputColFormats | Output Column Formats | Format of the Output Columns |

## ScalaUDF

This node runs any given Scala code for UDFs

### Input

.

### Type

scala

### Class

fire.nodes.etl.NodeUDFScala

### Fields

| Name | Title | Description |
|---|---|---|
| code | Scala | Scala code to be run. |

### Jython

This node runs any given Jython code. The input dataframe is passed in the variable inDF. The output dataframe should be placed in the variable outDF

#### Input

The input dataframe is passed in the variable in DF

#### Output

The output dataframe should be placed in the variable outDF

#### Type

transform

#### Class

fire.nodes.etl.NodeJython

#### Fields

#### Details

This node runs any given Jython code.

Below is an example jython code. It takes the input dataframe 'inDF', and returns the new dataframe 'outDF'

outDF = inDF.groupBy("c2").count()

### UnixShellCommands

This node execute shell command

#### Type

shellcommand

### Class

fire.nodes.etl.NodeShellCommand

### Fields

| Name | Title | Description |
|---|---|---|
| shellCommand | shell Command | Unix Shell Command |

## SQL

This node runs the given SQL on the incoming DataFrame

### Input

This type of node takes in a DataFrame and transforms it to another DataFrame

### Output

This node runs the given SQL on the incoming DataFrame to generate the output DataFrame

### Type

transform

### Class

fire.nodes.etl.NodeSQL

### Fields

## Scala

This node runs any given Scala code. The input dataframe is passed in the variable inDF. The output dataframe is passed back by registering it as a temporary table.

### Input

The input dataframe is passed in the variable inDF.

### Output

The output dataframe is passed back by registering it as a temporary table

#### Type

scala

#### Class

fire.nodes.etl.NodeScala

#### Fields

### PipePython

This node runs any given Python code. It pipes the incoming DataFrame through pipe to the Python Script. Output back to Spark has to be written out using print.

#### Input

It pipes the incoming DataFrame through pipe to the Python Script. It also passes the Schema of the DataFrame to the Python script through the command line argument - argv[1]

#### Output

Output back to Spark has to be written out using print.

#### Type

transform

#### Class

fire.nodes.etl.NodePipePython

#### Fields

| Name | Title | Description |
|---|---|---|
| code | Pipe Python | Python code to be run. It receives each record as a string and outputs records back as a string. |
| outputColNames | Output Column Names | Output Schema of Pipe Python Processor |
| outputColTypes | Output Column Types | Data Type of the Output Columns |
| outputColFormats | Output Column Formats | Format of the Output Columns |

### PySpark

This node runs any given PySpark code. The input dataframe is passed in the variable inDF. The output dataframe is passed back by registering it as a temporary table.

#### Input

The input dataframe is passed in the variable inDF.

#### Output

The output dataframe is passed back by registering it as a temporary table

#### Type

pyspark

#### Class

fire.nodes.etl.NodePySpark

#### Fields

### RunHIVEQL

This node runs the given SQL on the incoming DataFrame

#### Input

This type of node takes in a DataFrame and transforms it to another DataFrame

#### Output

This node runs the given SQL on the incoming DataFrame to generate the output DataFrame

#### Type

transform

#### Class

fire.nodes.etl.NodeRunHiveQL

#### Fields

| Name | Title | Description |
|---|---|---|
| hql | HiveQL - HIVE Query Language | HiveQL |

### 24.1.20 10-Visualization

#### GraphRegionGeo

This node displays values on a Map

#### Type

transform

#### Class

fire.nodes.graph.NodeGraphRegionGeo

#### Fields

#### PrintNRows

Prints the specified number of records in the DataFrame. It is useful for seeing intermediate output

#### Type

transform

#### Class

fire.nodes.util.NodePrintFirstNRows

#### Fields

#### GraphValues

#### Type

transform

#### Class

fire.nodes.graph.NodeGraphValues

#### Fields

### GraphGroupByColumn

Groups the data by the given column and plots the number of records in each group

#### Type

transform

#### Class

fire.nodes.graph.NodeGraphGroupByColumn

#### Fields

### Sample PrintNRows

Prints the specified number of records in the DataFrame. It is useful for seeing intermediate output

#### Type

transform

#### Class

fire.nodes.util.NodeSamplePrintFirstNRows

#### Fields

## 24.1.21 19-Deprecated

### StringToDate

This node converts a string column to date using the given date/time format

#### Type

transform

#### Class

fire.nodes.etl.NodeStringToDate

#### Fields

| Name | Title | Description |
|---|---|---|
| inputColName | Input Column Name | Input Column Name |
| inputColFormat | Input Column Format | Input Column Format. eg: yyyy-MM-dd yyyy-MM-dd HH:mm:ss |
| outputColName | Output Column Name | Output Column Name |
| outputColType | Output Column Type | Output Column Type |

#### Examples

#### Format Examples

dd-MM-yy : 31-01-12 dd-MM-yyyy : 31-01-2012 MM-dd-yyyy : 01-31-2012 yyyy-MM-dd : 2012-01-31 yyyy-MM-dd HH:mm:ss : 2012-01-31 23:59:59 yyyy-MM-dd HH:mm:ss.SSS : 2012-01-31 23:59:59.999 yyyy-MM-dd HH:mm:ss.SSSZ : 2012-01-31 23:59:59.999+0100 EEEEE MMMMM yyyy HH:mm:ss.SSSZ : Saturday November 2012 10:45:42.720+0100

**OUTPUT COLUMN NAME: - If user inputs an existing column name, it overrides the column** otherwise it will add a new column.

### 24.1.22 15-Streaming

#### StreamingSocketTextStream

Reads in streaming text from a socket

#### Input

It does not take any DataFrame as input

#### Output

It creates DataFrame from reading data from a socket. This DataFrame is passed to the output Nodes.

#### Type

sparkstreaming

### Class

fire.nodes.streaming.NodeStreamingSocketTextStream

### Fields

| Name | Title | Description |
|---|---|---|
| batchDuration | Batch Duration in Seconds | Batch Duration in Seconds |
| hostname | Hostname | Host to connect to for listening |
| port | Port | Port to connect to |

### Details

This Processor reads in messages from a Socket

### Key Fields

Below are the key fields of this Processor.

- hostname: this is the name of the host from where to read in the messages
- port: this is the port number from where to read in the messages

### Examples

Below is an example of the fields:

- hostname: localhost
- port: 8099

### StreamingKafka

Reads in streaming text from topics in Apache Kafka

### Input

It does not take any DataFrame as input

### Output

It reads events from Kafka and creates DataFrame from the resulting rows. This DataFrame is passed to the output Nodes.

#### Type

sparkstreaming

#### Class

fire.nodes.streaming.NodeStreamingKafka

#### Fields

| Name | Title | Description |
|---|---|---|
| batchDuration | Batch Duration in Seconds | Batch Duration in Seconds |
| brokers | Kafka Brokers | Kafka Brokers |
| group | Consumer Group | Consumer Group |
| topics | Kafka Topics | List of Topics separated by , (comma) |
| autoOffsetReset | auto.offset.reset | Auto Offset Reset |
| enableAutoCommit | enable.auto.commit | Enable Auto Commit |
| kafkaParamsKeys | Params Key/Value Pairs | More Config Values |
| kafkaParamsValues | Parms Key/Value Pairs | More Config Values |

### StreamingTextFileStream

It monitors a specified directory for new files. It keeps reading in any new files created in the directory.

#### Input

It does not take any DataFrame as input

#### Output

It reads the new files and creates DataFrame from the content of the text files. This DataFrame is passed to the output Nodes.

#### Type

sparkstreaming

#### Class

fire.nodes.streaming.NodeStreamingTextFileStream

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Directory from where to pick up files from |
| batchDuration | Batch Duration in Seconds | Batch Duration in Seconds |
| outputCol | Output Column | Output Column |

### 24.1.23 15-StructuredStreaming

#### StructuredStreamingCSV

It monitors a specified directory for new files. It keeps reading in any new files created in the directory.

#### Input

It does not take any DataFrame as input

#### Output

It reads the new files and creates DataFrame from the content of the text files. This DataFrame is passed to the output Nodes.

#### Type

sparkstreaming

#### Class

fire.nodes.structuredstreaming.NodeStructuredStreamingCSV

#### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path of the Text file/directory |
| separator | Separator | CSV Separator |
| outputColNames | Column Names for the CSV | Output Column Names |
| outputColTypes | Column Types for the CSV | Output Column Types |
| outputColFormats | Column Formats for the CSV | Output Column Formats |

### StructuredStreamingHiveSink2

Saves the streaming data into an Apache HIVE Table

#### Type

transform

#### Class

fire.nodes.structuredstreaming.NodeStructuredStreamingHiveSink2

#### Fields

| Name | Title | Description |
|---|---|---|
| databaseName | HIVE Database | Name of the HIVE Database |
| tableName | HIVE Table | Name of the HIVE table |

### StructuredStreamingFileSink

It writes the DataFrame to files with Structured Streaming

#### Input

It takes in DataFrame as input

#### Output

It writes the incoming DataFrame to files.

#### Type

transform

#### Class

fire.nodes.structuredstreaming.NodeStructuredStreamingFileSink

### Fields

| Name | Title | Description |
|---|---|---|
| path | Path | Path where to write the files |
| outputMode | Output Mode | Output Mode for saving to Files |
| checkpointLocation | Checkpoint Location | Checkpoint Location on HDFS compatible file system for Streaming |
| format | Format | File Format |
| partitionBy | Partition By Columns | Partition By Columns separated by space (can be empty in which case partitionBy would not be applied) |

## StructuredStreamingSocket

Reads in streaming text from a socket

### Input

It does not take any DataFrame as input

### Output

It reads events a socket and creates DataFrame from the resulting rows. This DataFrame is passed to the output Nodes.

### Type

sparkstreaming

### Class

fire.nodes.structuredstreaming.NodeStructuredStreamingSocket

### Fields

| Name | Title | Description |
|---|---|---|
| host | Hostname | Host to connect to for listening |
| port | Port | Port to connect to |

## StructuredStreamingHiveSink

Saves the streaming data into a HIVE Table

#### Type

transform

#### Class

fire.nodes.structuredstreaming.NodeStructuredStreamingHiveSink

#### Fields

| Name | Title | Description |
|---|---|---|
| databaseName | HIVE Database | Name of the HIVE Database |
| tableName | HIVE Table | Name of the HIVE table |

### StructuredStreamingKinesis

Reads in streaming text from Kinesis stream

#### Input

It does not take any DataFrame as input

#### Output

It reads events from Kinesis and creates DataFrame from the resulting rows. This DataFrame is passed to the output Nodes.

#### Type

sparkstreaming

#### Class

fire.nodes.structuredstreaming.NodeStructuredStreamingKinesis

#### Fields

| Name | Title | Description |
|---|---|---|
| streamName | Stream Name | Kinesis Stream Name |
| endpointUrl | Endpoint Url | Kinesis Endpoint Url |
| editorData | Editor Data | Data to be used for testing in the Workflow Editor |

### StructuredStreamingKafka

Reads in streaming text from topics in Apache Kafka

#### Input

It does not take any DataFrame as input

#### Output

It reads events from Kafka and creates DataFrame from the resulting rows. This DataFrame is passed to the output Nodes.

#### Type

sparkstreaming

#### Class

fire.nodes.structuredstreaming.NodeStructuredStreamingKafka

#### Fields

| Name | Title | Description |
|---|---|---|
| batchDuration | Batch Duration in Seconds | Batch Duration in Seconds |
| brokers | Kafka Brokers | Kafka Brokers |
| group | Consumer Group | Consumer Group |
| topics | Kafka Topics | List of Topics separated by , (comma) |
| autoOffsetReset | auto.offset.reset | Auto Offset Reset |
| enableAutoCommit | enable.auto.commit | Enable Auto Commit |
| kafkaParamsKeys | Params Key/Value Pairs | More Config Values |
| kafkaParamsValues | Parms Key/Value Pairs | More Config Values |

### StructuredStreamingConsoleSink

It output the DataFrame to the console

#### Input

It takes in DataFrame as input

#### Output

It writes the incoming DataFrame to the console.

#### Type

transform

#### Class

fire.nodes.structuredstreaming.NodeStructuredStreamingConsoleSink

#### Fields

| Name | Title | Description |
|---|---|---|
| outputMode | Output Mode | Output Mode for saving to Files |

### 24.1.24 14-DL

#### KerasModelFit

#### Type

ml-estimator

#### Class

fire.nodes.dl.NodeModelFit

### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Target Column | The label column for model fitting |
| batch_size | BatchSize | Default value is None i.e -1 |
| epochs | Epochs | |
| verbose | Verbose | |
| callbacks | Callbacks | Default value is None i.e -1 |
| validation_split | ValidationSplit | |
| validation_data | ValidationData | Default value is None i.e -1 |
| shuffle | Shuffle | |
| class_weight | ClassWeight | Default value is None i.e -1 |
| sample_weight | SampleWeight | Default value is None i.e -1 |
| initial_epoch | InitialEpoch | |
| steps_per_epoch | StepsPerEpoch | Default value is None i.e -1 |
| validation_steps | ValidationSteps | Default value is None i.e -1 |
| validation_freq | ValidationFreq | |
| max_queue_size | MaxQueueSize | |
| workers | Workers | |
| use_multiprocessing | UseMultiprocessing | |

## KerasPredict

### Type

ml-predict

### Class

fire.nodes.dl.NodePredict

### Fields

| Name | Title | Description |
|---|---|---|
| targetCol | Target Column | The label column for model fitting |
| batch_size | BatchSize | Default value is None i.e -1 |
| verbose | Verbose | |
| steps | Steps | Default value is None i.e -1 |
| callbacks | Callbacks | Default value is None i.e -1 |
| max_queue_size | ValidationFreq | |
| workers | Workers | |
| use_multiprocessing | UseMultiprocessing | |

## KerasModelCompile

#### Type

transform

#### Class

fire.nodes.dl.NodeModelCompile

#### Fields

| Name | Title | Description |
|---|---|---|
| optimizer | Optimizer | |
| loss | Loss | |
| metrics | Metrics | |
| loss_weights | LossWeights | |
| sample_weight_mode | SampleWeightMode | |
| weighted_metrics | WeightedMetrics | |
| target_tensors | TargetTensors | |

### DenseLayer

#### Type

transform

#### Class

fire.nodes.dl.NodeDense

#### Fields

| Name | Title | Description |
|---|---|---|
| units | Units | |
| activation | Activation | |
| use_bias | Use Bias | |
| kernel_initializer | Kernel Initializer | |
| bias_initializer | Bias Initializer | |
| kernel_regularizer | Kernel Regularizer | |
| bias_regularizer | Bias Regularizer | |
| activity_regularizer | Activity Regularizer | |
| kernel_constraint | Kernel Constraint | |
| bias_constraint | Bias Constraint | |

#### KerasModelSequential

##### Type

transform

##### Class

fire.nodes.dl.NodeModelSequential

##### Fields

| Name | Title | Description |
|---|---|---|
| layers | Layers | |

### 24.1.25 07-JoinUnion

#### UnionAll

This node creates a new DataFrame by merging all the rows without removing the duplicates

##### Input

It accepts a DataFrame as input from the previous Node

##### Output

This node does union of all the rows without removing the duplicates

##### Type

join

##### Class

fire.nodes.etl.NodeUnionAll

##### Fields

#### GeoJoin

This node joins the incoming dataframes

#### Input

This node takes in 2 DataFrames as input and produces one DataFrame as output

#### Type

join

#### Class

fire.nodes.etl.NodeGeoJoin

#### Fields

| Name | Title | Description |
|---|---|---|
| latitudeCol | Latitude Column | Latitude Column from first DataFrame |
| longitudeCol | Longitude Column | Longitude Column from first DataFrame |
| polygonCol | Polygon Column | Polygon Column from second DataFrame |

### JoinOnCommonColumns

This node joins the incoming dataframes on 1 or more columns

#### Input

It takes in 2 DataFrames as input and produces one DataFrame as output by joining on the specified columns

#### Output

The output DataFrame produced as a result of joining the incoming DataFrames on the specified columns

#### Type

join

#### Class

fire.nodes.etl.NodeJoinUsingColumns

### Fields

| Name | Title | Description |
|---|---|---|
| joinCols | Common Join Columns | Space separated list of columns on which to join |
| joinType | Join Type | Type of Join |
| outputColNames | Output Column Names | Name of the Output Columns |
| outputColTypes | Output Column Types | Data Type of the Output Columns |
| outputColFormats | Output Column Formats | Format of the Output Columns |
| whereClause | Where Clause | where condition after join function |

## JoinOnColumns

### Type

join2inputs

### Class

fire.nodes.etl.JoinOnColumns

### Fields

| Name | Title | Description |
|---|---|---|
| joinType | Join Type | Type of Join |
| leftTableJoinColumn | LeftTableJoinColumn | |
| rightTableJoinColumn | RightTableJoinColumn | |

## JoinUsingSQL

This node registers the incoming DataFrames as temporary tables and executes the SQL provided

### Input

It takes in 2 DataFrames as input and produces one DataFrame as output by executing the provided SQL.

### Output

The DataFrame created as a result of executing the join SQL

#### Type

join

#### Class

fire.nodes.etl.NodeJoinUsingSQL

#### Fields

### UnionDistinct

This node creates a new DataFrame by performing a DISTINCT on the result set, eliminating any duplicate rows

#### Input

It takes in multiple DataFrames as input

#### Output

This node does union of all the rows from the incoming DataFrames to generate the output DataFrame

#### Type

join

#### Class

fire.nodes.etl.NodeUnionDistinct

#### Fields

### JoinOnCommonColumn

This node joins the incoming dataframes on a joinCol

#### Input

This node takes in 2 DataFrames as input and produces one DataFrame as output

#### Output

The output DataFrame is the result of joining the 2 incoming DataFrames on the join column

### Type

join

### Class

fire.nodes.etl.NodeJoinUsingColumn

### Fields

| Name | Title | Description |
|---|---|---|
| joinCol | Common Join Column | column on which to join |

CHAPTER 25

# Release Notes

## 25.1 Release Notes

### 25.1.1 Upcoming Features

Below are the upcoming features in Fire Insights.

#### Installer

A one-click installer and update for Fire Insights.

Users would be able to install and update Fire Insights on their laptops with one click.

### 25.1.2 Aug 2020

#### New Features

- Time Series Modeling with Prophet
- Time Series Modeling with Arima
- Building Custom Nodes in Python

#### UI Improvements

- Upgraded look and feel

### 25.1.3 May 2020

#### New Features

- Added viewing of Fire Insights logs under Administration Menu
- Added more details to Data Profiling
- Added file upload and delete capabilities in DBFS browser
- Added ability to create datasets for data on AWS S3.
- Added configurations for AWS Home Directory to restrict access of other bucket or folder
- Added interactive dashboards
- Added ability to view workflows by type : Normal, Data Profiling, Dataset Cleaning

#### UI Improvements

- Each workflow list page now displays up to 50 workflows

### 25.1.4 April 2020

#### New Features

- Added browsing of AWS S3 file system under Data Browser
- Added uploading files to S3
- Added creating folder on S3
- Added deleting files on S3
- Added reporting for Total Users, Groups, Projects, Workflows & Workflows Executions

#### UI Improvements

- Autocomplete feature added to SQL editor in workflows

### 25.1.5 March 2020

#### New Features

- Integration with Databricks
- Added Browsing Databricks DB, Databricks Cluster & DBFS
- Added Scheduling in Standalone Mode
- Compatible with Amazon Aurora Database

#### UI Improvements

- Improvement of Metrics which include stage information
- Improvement in JOIN USING SQL Processors

### 25.1.6 February 2020

#### New Features

- Job Metrics Integration with improvements.
- SUPERUSER to have more rights when elevated access is enabled.
- If user is inactive, he is unable to login.
- Added Runtime Statistics.
- Added Compare Model.

#### UI Improvment

- Improvement to Connection page.

### 25.1.7 January 2020

#### New Features

- Integrated with yarn which enable us to see detail information of job submitted to cluster
- Integrated with Job Metrics
- Added plugins for GoogleRestApiKey in Configurations
- Added Geo chart: Country & Geo chart: Lat, Lon features in Interactive Dahboard
- Integrated with Model List and Summary Page for viewing detail information about the model
- Added Reload Sample Application Features

### 25.1.8 September 2019

#### New Processors Added For Scala Engine

- MultiWindowAnalytics
- MultiWindowRanking

#### New Processors Added For Pyspark Engine

- SaveAvro
- SaveJSON

#### Improvement of RESTAPI

#### New Features

- Integrated File Watcher with AWS
- Database Cleanup for workflow execution & workflow execution results

- Export of all users implemented
- Added search help with search option to Quickstart Guide, Tutorials & FAQ

#### Upgrades for Security Vulnerabilties

- All the dependencies have been upgraded to handle security vulnerabilities.

#### UI Improvement

- Improvement of WorkflowEeditor Page to make it easy to add the workflow parameters.

### 25.1.9 August 2019

#### New Processors Added For Scala Engine

- WindowAnalytics
- WindowRanking
- H2OGLRM
- H2OWord2Vec

#### New Processors Added For Pyspark Engine

- ZipWithIndex
- ReadAvro
- ReadJSON
- ReadParquet

#### UI Improvements

- Drag and drop function for node in workflow editor
- Improvement of workflow editor page look & feel.

### 25.1.10 July 2019

#### Integration of H2O

- The following New H2O Processors have been added :
  - H2ODRF
  - H2OGBM
  - H2OGLM
  - H2OIsolationForest
  - H2OKMeans

- H2OModelLoad
- H2OModelSave
- H2OMojoLoad
- H2OMojoSave
- H2ONaiveBayes
- H2ONeuralNetwork
- H2OPCA
- H2OScore

#### Improvements in UI

- Login Page of Fire Insights has been upgraded.
- Scatter Plot look and feel has been upgraded.

#### Improvements to HDFS Browser

- Ability to edit files and directories.

#### Improvements in Home Dashboard Page

#### Added New Features

- Added Search Box to Search Workflow, Node, Dataset & Dashboard available in an application.
- Added Self-Registration to create a user directly from Login page.

#### Upgradation of Running Server on Ports

- Fire Insights now enable us to run Fire & Pspark server on different ports.

### 25.1.11 June 2019

The following features have been released in June 2019.

#### Improvements in UI

- Displaying text in Workflow Execution Page with more details visible.
- CSV and other read file nodes, now display the name of the file.
- When cloning a node in the editor, the cloned node is created close to the original node.

#### Improvements to HDFS Browser

- Fire Insights now allows moving multiple files from one directory to another.

#### Support Of Authentication Using Token

- Fire now supports two methods Of getting tokens to access Fire

Grant Types – Password.

Grant Types – Authorization code.

#### Improvements in Dataset

- Look and feel of the edit Dataset page has been upgraded.

#### Running Applications Locally

- Workflows when running locally are now executed as separate Java or Python processes.

#### Node Updates

- JoinUsingSQL now allows joining multiple datasets at a time.

### 25.1.12 May 2019

The following features have been released in May 2019.

#### PySpark Engine

- New Engine for running PySpark

#### New Processors

#### Outlier Detection

- New Node for Outlier Detection

#### Improvements to HDFS Browser

- Displaying user permission for each file/directory
- Displaying an icon indicating whether it is file or directory
- Better display of error messages

#### Applications

- Datasets tab is the first tab now

#### Datasets

- Better display of the Create/Edit dataset page
- Do not display JDBC passwords

#### Workflow Editor

- Ability to create DataSet Nodes by browsing the list of datasets
- HIVE DB Browser on the LHS
- Better display of the processors
- Fix for tabs in dialogs not showing up (eg. in Logistic Regression Processor)

### 25.1.13 April 2019

#### New Processors Added For Scala Engine

- MultiWindowAnalytics
- MultiWindowRanking

#### New Processors Added For Pyspark Engine

- SaveAvro
- SaveJSON

#### Improvement of RESTAPI

#### New Features

- Integrated File Watcher with AWS
- Database Cleanup for workflow execution & workflow execution results
- Export of all users
- Added search help with search option to quickstart guide, tutorials & FAQ

#### Upgrades for Security Vulnerabilties

- All the dependencies have been upgraded to handle security vulnerabilities.

#### UI Improvement

- Improvement of workflow editor page to make it easy to add the workflow parameters.

### 25.1.14 February 2019

The following features have been released in Feb 2019.

#### Correlation Node Output

In Heatmap the colors are not repeated.

#### Scheduled Workflow Edit

Fire now enables editing of already scheduled workflows for executions.

#### Multiple users in a Group

Fire now enables you to add multiple users to a group.

#### SaveMongoDB Node

Fire now enables you to save your data to MongoDB using this node.

#### Interactive Dashboard Improvements

- Allows 2 items or more in y-axis in Histogram Chart.
- When there are 2 items on x-axis, only one item is allowed on the y-axis.

### 25.1.15 January 2019

The following features have been released in Jan 2019.

#### Interactive Dashboards

Fire now enables you to create Interactive Dashboards. Interactive Dashboards pull data from JDBC sources.

#### Workflow Wizard

Workflow Wizard enables you to quickly create workflows of various kinds. These could be data cleaning, reporting, spam detection, churn prediction etc.

#### Pipelines

Fire now supports Pipelines. Pipelines allow creating a DAG of workflows. In the future it would allow adding more types of nodes to the DAG.

#### Charts Improvements

- Ability to display more than 1 heatmap in a workflow
- Display of X-values and X-axis in the Charts

#### Processor Improvements

- In RowFilter Processor, the size of conditional expression textfield has been increased.

#### Support for Uploading Large Files

Fire now supports uploading very large files.

### 25.1.16 November 2018

The following features have been released in Nov 2018.

#### Support for Applications

You can now create Applications in Fire. Applications can contain:

- Datasets
- Workflows
- Dashboards
- Sharing information

This allows you to easily create complex Big Data and ML Applications and work in groups.

#### Structured Streaming

Fire now supports Structured Streaming. It provides a number of Processors for Structured Streaming. These include Processors for reading from Kafka, reading from files etc. There are also a number of Processors for writing to files etc.

### 25.1.17 3.1.0 Release Notes

- Release Date: 09/01/2018
- Download TGZ name: sparkflows-fire-3.1.0.tgz
- TGZ Size: 505 MB

#### Contents of this release

- **New Processors Added**
  - Decision Node Processor
  - JSON Parser Processor
  - SortBy Processor
  - Empty Dataset Processor
  - Multi Validation Processor
  - String Function Multiple Processor

    - Math Function Multiple Processor
    - Case When Processor
    - Remove Duplicate Processor

- Support for uploading files to HDFS
- Support for LDAP
- **Support for running the workflows in debug mode.**
    - In debug mode, the number of records processed at each Node are printed.
    - SQL executed is printed where relevant
- **Various Workflow Editor Upgrades**
    - Ability to rename the Nodes
    - Richer support in JDBC Processor for interactive execution
    - Save Warning when moving away from the Workflow Editor
    - Rich widget support for Multi-Validations Processor
- Support for Caching Datasets in any Processor
- Support for Workflow Cloning
- Richer support in Dashboard Editor for drag and drop of Processors

### 25.1.18 2.1.0 Release Notes

- Release Date: 04/01/2018
- Download TGZ name: sparkflows-fire-2.1.0.tgz
- TGZ Size: 508 MB

#### Contents of this release

- Separation of Workflow Server from Workflow Engine
- Support for HDFS File Upload
- **New Processors**
    - HBase Read Processor
    - HBase Write Processor
    - Split by Multiple Expressions Processor
    - Fixes to Node Correlation
- Support for Rich REST API's

### 25.1.19 1.4.0 Release Notes

- Release Date: 11/29/2017
- Download TGZ name: sparkflows-fire-1.4.3.tgz
- TGZ Size: 485 MB

#### Contents of this release

- Scheduling Workflows
- Support for ORC files
- Support for ElasticSearch
- Running in YARN Cluster Mode
- Better browsing experience
- Support for more widget types
- Fixes to Node Correlation Matrix
- Elastic Search Integration
- Support for OpenNLP

### 25.1.20 1.3.0 Release Notes

- Release Date: 1/8/2017
- Download TGZ name: sparkflows-fire-1.3.0.tgz
- TGZ Size: 485 MB

#### Contents of this release

- Interactive Workflow Execution
- Streaming Workflow Engine
- Saving & Loading Models
- Support for Jython Nodes
- Many new Machine Learning Nodes added
- Many User Interface Improvements

# CHAPTER 26

## REST API Authentication

## 26.1 REST API Authentication

Sparkflows provides REST API for interacting with it.

Swagger is also enabled and is available at http://<machine-name>:8080/swagger-ui.html

To authenticate and access Fire Insights REST APIs, you can use personal access tokens or passwords. We strongly recommend that you use tokens. Like passwords, tokens should be treated with care. Unlike passwords, tokens expire and can be revoked.

Tokens can be generated using Postman.

You can also log in with your username/password, get a session cookie, store it into a file and use it in subsequent requests.

### 26.1.1 Acquire Session Cookie Using CURL

When invoking the REST APIs of Fire Insights with curl, the first step is to log in and save the incoming cookie into a text file. This file would then be used in making subsequent REST calls via curl.

Save the incoming cookies using the `-c` option of `curl` into a file.

In the below example, the Fire Insights web server is running on the local machine at : `localhost:8080`

You can replace it with your machine name and port.

**CURL**:

```
curl -i -X POST -d username=admin -d password=admin -c /tmp/cookies.txt␣
↪localhost:8080/login
```

In the above:

- username = admin
- password = admin

- Incoming cookie gets saved into : /tmp/cookies.txt
- REST API endpoint : localhost:8080/login

### 26.1.2 Acquire Session Cookie in Python

Fire Insights REST API's can be accessed with Python. Session Cookie can be acquired using username and password and used in the subsequent calls.

#### Get List of Processors

The below code in Python logs in the user and acquires the session cookie via the Fire Insights REST API.

It then gets the list of Processors in Fire Insights using the REST API and prints them.

```
#!/usr/bin/python

# This python script logs into an instance of sparkflows, and then gets the list of Processors/Operators supported

# -*- coding: utf-8 -*-
import json
import requests

payload = {'username':'admin', 'password':'admin'}

# login url
urllogin = 'http://localhost:8080/login'

# get list of processors url
urlprocessors = 'http://localhost:8080/nodeList'

with requests.session() as s:

  # log into sparkflows
  r = s.post(urllogin, data=payload)

  # get list of processors
  resp = s.get(urlprocessors)

  parsed_resp = json.loads(resp.text)

  for i in parsed_resp:
    print (i['name'])
```

### 26.1.3 Acquire Token Using CURL

Tokens can be acquired from Fire Insights using curl. They would then be used in making subsequent curl requests.

This page work is in progress…

### 26.1.4 Acquire Token using Postman and Grant Type - Password

This document describes the steps to obtain and use OAuth 2.0 access tokens using Postman.

### Overview of Grant Type – Password

The Password grant is used when the application presents a traditional username and password login form to collect the user's credentials and makes a POST request to the server to exchange the password for an access token. The POST requests that the application made looks like the example below.

### Form the Post Request

The POST Request method requests that a web server accepts the data enclosed in the body of the request message, most likely for storing it

Table 1: Below are the Relevant Request

| Title | URL |
|---|---|
| POST | http://hostname:port/oauth/token?grant_type=password&username=<username>&password=<password> |

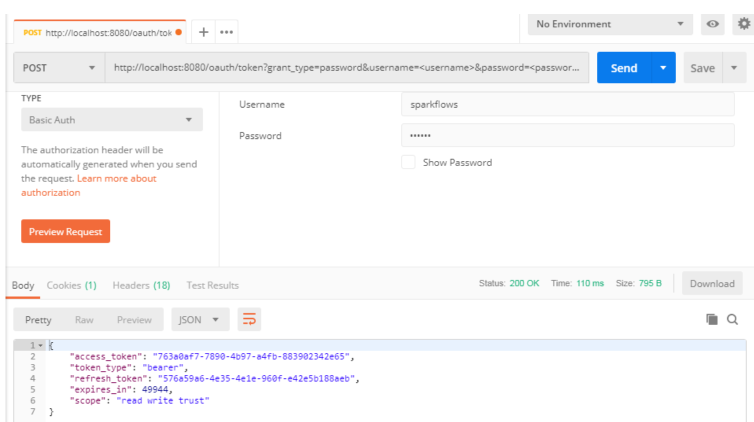
Update the username and password in URL and use as request header.

### Click on Authorization tab and select Type - Basic Auth

Basic Auth is an authorization type that requires a verified username and password to access a data resource.

Use default Username `sparkflows` and Password `secret` for client authentication. Click on `Send` to authorize the user and get the access token.

### Example

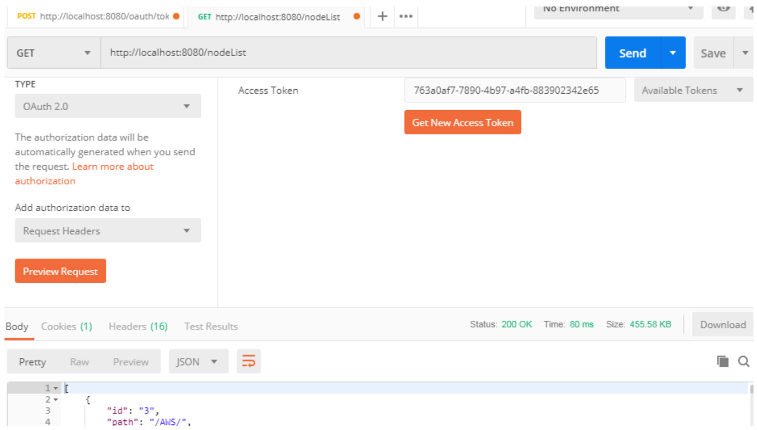



### Now use access_token from previous step to access the REST API

An Access Token is a credential that can be used by an application to access an API. Below is an example to invoke the `nodeList` REST API of Fire Insights.

## 26.1.5 Acquire token using Postman - Authorization code

The Authorization Code grant type is used to exchange an authorization code for an access token.

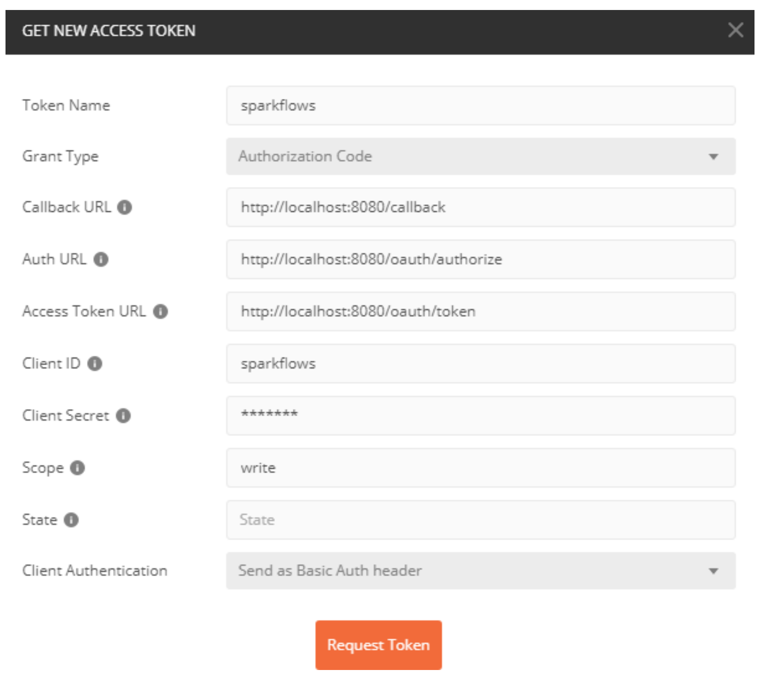



## Get the access token

The app can obtain an access token that provides temporary, secure access to it. Below are steps involved to Request an Access_token

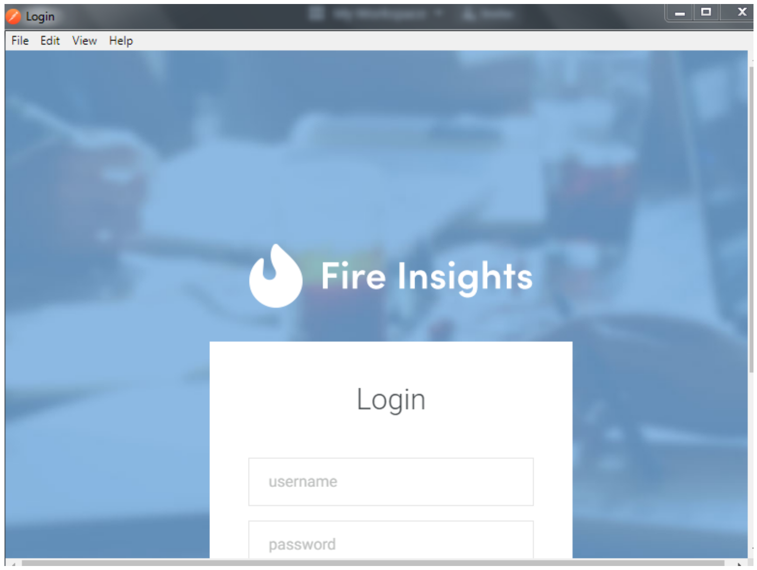
## Click on Authorization tab

- Select Type OAuth 2.0




## Click on Request Token

It will redirect to sparkflows login URL Page.

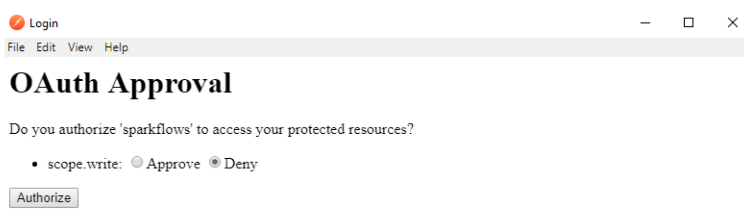



### Fill the username and password and click on signIn

It will then display the OAuth Approval page.

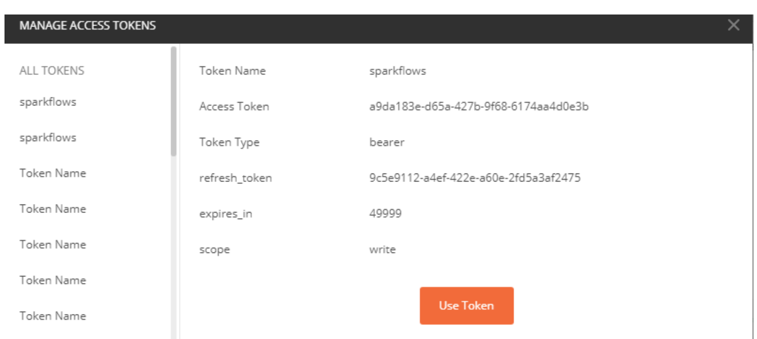
### OAuth Approval

OAuth is an authentication protocol that allows you to approve one application interacting with another on your behalf without giving away your password. Below is the Screenshot for updating the Oauth approval.



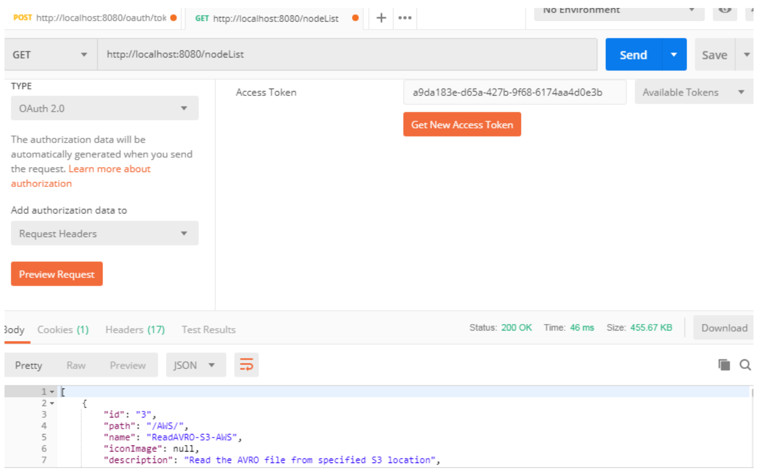

### Click on Use token

A security token (sometimes called an authentication token) is a small hardware device that the owner carries to authorize access to a network service.



### Using tokens for accessing REST API

Using above token we can access the REST API.




## 26.1.6 Acquire Token in Python - Grant Type Password

Below are examples of Python code for accessing the Fire REST API using Python.

### Get Processor Count

The below code in Python does the following:

- Acquires the token using Grant Type Password
- Invokes the Fire Insights REST API to get the number of processors list available in Fire Insights.

```
#!/usr/bin/python

import requests

import json

import getpass

token_url = "http://hostname:8080/oauth/token"

processor_count_api_url = "http://hostname:8080/getNodeCount" # processor list count api of sparkflows

#Step A - resource owner supplies credentials
#Resource owner (enduser) credentials

RO_user = 'admin'
RO_password = 'admin'

#client (application) credentials
client_id = 'sparkflows'
client_secret = 'secret'
```

(continues on next page)

(continued from previous page)

```
#step B, C - single call with resource owner credentials in the body and client␣
↪credentials as the basic auth header will return #access_token

data = {'grant_type': 'password','username': RO_user, 'password': RO_password}

access_token_response = requests.post(token_url, data=data, verify=False, allow_
↪redirects=False, auth=(client_id, client_secret))

print(access_token_response.headers)
print(access_token_response.text)

tokens = json.loads(access_token_response.text)
print( "access token: " + tokens['access_token'])

# Step C - now we can use the access_token to make another rest api call to get␣
↪the processor count

api_call_headers = {'Authorization': 'Bearer ' + tokens['access_token']}

print( api_call_headers)

api_call_response = requests.get(processor_count_api_url, headers=api_call_
↪headers, verify=False)

print(api_call_response.text)
```

After running above REST API code in Python, we get the below results.

### Infer Hadoop Cluster Configurations

The below code in Python invokes the Fire Insights REST API to infer Hadoop cluster configurations. It then saves the infer cluster Hadoop configurations as updated values.

```
#!/usr/bin/python

import requests

import json

token_url = "http://hostname:8080/oauth/token"

infer_configuration_api_url = "http://hostname:8080/api/v1/configurations/infer"

save_configuration_api_url = "http://hostname:8080/api/v1/configurations"

#Step A - resource owner supplies credentials
#Resource owner (enduser) credentials

RO_user = 'admin' #input your own username
RO_password = 'admin' #input your own password

#client (application) credentials
```

(continues on next page)

(continued from previous page)

```python
client_id = 'sparkflows'
client_secret = 'secret'

#step B, C - single call with resource owner credentials in the body and client credentials as the basic auth header will return #access_token

data = {'grant_type': 'password','username': RO_user, 'password': RO_password}

access_token_response = requests.post(token_url, data=data, verify=False, allow_redirects=False, auth=(client_id, client_secret))

print(access_token_response.headers)
print(access_token_response.text)

tokens = json.loads(access_token_response.text)
print( "access token: " + tokens['access_token'])

#Step- now use the access_token to call infer configuration api and its save api.

api_call_headers = {'Authorization': 'Bearer ' + tokens['access_token']}

print( api_call_headers)

#infer the hadoop configuration

infer_configuration_api_response = requests.get(infer_configuration_api_url, headers=api_call_headers, verify=False)
print(" infer configuration response : "+ infer_configuration_api_response.text)

#save the hadoop configuration

save_configuration_api_response = requests.post(save_configuration_api_url,json=infer_configuration_api_response.json(), headers=api_call_headers, verify=False)

print(" configuration after save : "+save_configuration_api_response.text)
```

After running above REST API code using Python, Will get the results as below

CHAPTER 27

# REST API's using Python

## 27.1 REST API Examples using Python

Sparkflows provides REST API for interacting with it.

Below are examples using tokens. The first step is to log in with your username and password and acquire the token.

Swagger is also enabled and is available at http://<machine-name>:8080/swagger-ui.html

### 27.1.1 Accessing REST API using Python & Session

Fire Insights REST APIs can be accessed with Python. This page provides 2 examples of accessing the REST API's with Python.

#### Get List of Processors

The below code in Python gets the list of Processors in Fire Insights using the REST API and prints them.

```
#!/usr/bin/python

# This python script logs into an instance of sparkflows, and then gets the list of␣
↪Processors/Operators supported

# -*- coding: utf-8 -*-
import json
import requests

payload = {'username':'admin', 'password':'admin'}

# login url
urllogin = 'http://localhost:8080/login'

```

(continues on next page)

(continued from previous page)

```
# get list of processors url
urlprocessors = 'http://localhost:8080/nodeList'

with requests.session() as s:

  # log into sparkflows
  r = s.post(urllogin, data=payload)

  # get list of processors
  resp = s.get(urlprocessors)

  parsed_resp = json.loads(resp.text)

  for i in parsed_resp:
    print (i['name'])
```

### Create a New Workflow

The Workflow JSON is saved in a file called `workflow.json`.

The below code in Python creates a new Workflow in the Project with id 1.

```
#!/usr/bin/python

# This python script logs into an instance of sparkflows, and then gets the list of
↪Processors/Operators supported

# -*- coding: utf-8 -*-
import json
import requests

payload = {'username':'admin', 'password':'admin'}

# login url
urllogin = 'http://localhost:8080/login'

# save workflow url
urlsaveworkflow = 'http://localhost:8080/saveWorkflow'

# read workflow json
wf = open("workflow.json","r", encoding='utf8')
workflow = wf.read()

# define other parameters
analysisFlowId = "null"
projectId = "1"
engine = "scala"

with requests.session() as s:

  # log into sparkflows
  s.get(urllogin)

  r = s.post(urllogin, data=payload)

```

(continues on next page)

(continued from previous page)

```
# save workflow
headers = {'Content-type': 'application/json', 'Accept': 'text/plain', 'analysisFlowId': analysisFlowId, 'projectId': projectId, 'engine':engine }
resp = s.post(urlsaveworkflow, data=workflow, headers=headers)

print(resp)
```

### 27.1.2 Accessing REST API using Python & Tokens

Below are examples of Python code for accessing the Fire REST API using Python.

#### Get Processor Count

The below code in Python invokes the Fire Insights REST API to calculate number of processors list available in Fire Insight.

```
#!/usr/bin/python

import requests

import json

import getpass

token_url = "http://localhost:8080/oauth/token"

processor_count_api_url = "http://localhost:8080/getNodeCount" # processor list count api of sparkflows

#Step A - resource owner supplies credentials
#Resource owner (enduser) credentials

RO_user = 'admin'
RO_password = 'admin'

#client (application) credentials
client_id = 'sparkflows'
client_secret = 'secret'

#step B, C - single call with resource owner credentials in the body and client credentials as the basic auth header will return #access_token

data = {'grant_type': 'password','username': RO_user, 'password': RO_password}

access_token_response = requests.post(token_url, data=data, verify=False, allow_redirects=False, auth=(client_id, client_secret))

print(access_token_response.headers)
print(access_token_response.text)

tokens = json.loads(access_token_response.text)
print( "access token: " + tokens['access_token'])
```

(continues on next page)

(continued from previous page)

```
# Step C - now we can use the access_token to make as many calls as we want.

api_call_headers = {'Authorization': 'Bearer ' + tokens['access_token']}

print( api_call_headers)

api_call_response = requests.get(processor_count_api_url, headers=api_call_
↪headers, verify=False)

print(api_call_response.text)
```

After running above REST API code using Python, will get the results as below:

### Infer Hadoop Cluster Configurations

The below code in Python invokes the Fire Insights REST API to infer Hadoop cluster configurations. It then saves the infer cluster Hadoop configurations as updated values.

```
#!/usr/bin/python

import requests

import json

token_url = "http://localhost:8080/oauth/token"

infer_configuration_api_url = "http://localhost:8080/api/v1/configurations/infer"

save_configuration_api_url = "http://localhost:8080/api/v1/configurations"

#Step A - resource owner supplies credentials
#Resource owner (enduser) credentials

RO_user = 'admin' #input your own username
RO_password = 'admin' #input your own password

#client (application) credentials

client_id = 'sparkflows'
client_secret = 'secret'

#step B, C - single call with resource owner credentials in the body and client
↪credentials as the basic auth header will return #access_token

data = {'grant_type': 'password','username': RO_user, 'password': RO_password}

access_token_response = requests.post(token_url, data=data, verify=False, allow_
↪redirects=False, auth=(client_id, client_secret))

print(access_token_response.headers)
print(access_token_response.text)
```

(continues on next page)

(continued from previous page)

```
tokens = json.loads(access_token_response.text)
print( "access token: " + tokens['access_token'])

#Step- now use the access_token to call infer configuration api and its save api.

api_call_headers = {'Authorization': 'Bearer ' + tokens['access_token']}

print( api_call_headers)

#infer the hadoop configuration

infer_configuration_api_response = requests.get(infer_configuration_api_url, headers=api_call_headers, verify=False)
print(" infer configuration response : "+ infer_configuration_api_response.text)

#save the hadoop configuration

save_configuration_api_response = requests.post(save_configuration_api_url,json=infer_configuration_api_response.json(), headers=api_call_headers, verify=False)

print(" configuration after save : "+save_configuration_api_response.text)
```

After running above REST API code using Python, will get the results as below

CHAPTER 28

# REST API's using Java

## 28.1 REST API Examples using Java

Fire Insighs provides REST API for interacting with it.

Below are examples using tokens. The first step is to log in with your username and password and acquire the token.

Swagger is also enabled and is available at http://<machine-name>:8080/swagger-ui.html

CHAPTER 29

# REST API's using curl

## 29.1 REST API Examples using curl

This topic contains a range of examples that demonstrate how to use the Fire Insights API using curl.

**Acquire Session Cookie Using Curl**

When invoking the REST APIs of Fire Insights with curl, the first step is to log in and save the incoming cookie into a text file. This file would then be used in making subsequent REST calls via curl.

Save the incoming cookies using the `-c` option of `curl` into a file.

In the below examples, the Fire Insights web server is running on the local machine at : `localhost:8080`

You can replace it with your machine name and port.

*Login and save the session cookie into /tmp/cookies.txt*:

```
curl -i -X POST -d username=admin -d password=admin -c /tmp/cookies.txt␣
↪localhost:8080/login
```

In the above:

- username = admin
- password = admin
- Incoming cookie gets saved into : /tmp/cookies.txt
- REST API endpoint : localhost:8080/login

There are various categories of REST API's available:

### 29.1.1 Processors REST API's

### Overview

The Processors REST APIs, allow you to get the list of available Processors and details regarding each Processor.

Below are the various Processor APIs available in Fire Insights.

They should be executed after you have logged into Fire Insights. Use the -b option to use the cookies previously saved.

### GET Processors List

Gets the list of processors available.

An example request for getting list of processors:

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/nodes' -
↪b /tmp/cookies.txt
```

An example response:


```
  [
{
  "id": "3",
  "path": "/01-Connectors/",
  "name": "ReadCassandra",
  "iconImage": null,
  "description": "This node reads data from Apache Cassandra",
  "details": "",
  "examples": "",
  "type": "dataset",
  "nodeClass": "fire.nodes.cassandra.NodeReadCassandra",
  "x": null,
  "y": null,
  "fields": [
    {
      "name": "storageLevel",
      "value": "DEFAULT",
      "widget": "array",
      "title": "Output Storage Level",
      "description": "Storage Level of the Output Datasets of this Node",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": [
        "DEFAULT",
        "NONE",
        "DISK_ONLY",
        "DISK_ONLY_2",
        "MEMORY_ONLY",
        "MEMORY_ONLY_2",
        "MEMORY_ONLY_SER",
        "MEMORY_ONLY_SER_2",
        "MEMORY_AND_DISK",
        "MEMORY_AND_DISK_2",
        "MEMORY_AND_DISK_SER",
        "MEMORY_AND_DISK_SER_2",
        "OFF_HEAP"
      ],
      "required": false,
```


(continues on next page)

(continued from previous page)


```
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "table",
      "value": "",
      "widget": "textfield",
      "title": "Cassandra Table",
      "description": "Cassandra Table from which to read the data",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": true,
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "keyspace",
      "value": "",
      "widget": "textfield",
      "title": "Cassandra Keyspace",
      "description": "Cassandra Keyspace",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": true,
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "cluster",
      "value": "",
      "widget": "textfield",
      "title": "Cassandra Cluster",
      "description": "The group of the Cluster Level ",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": false,
      "display": true,
      "editable": true,
      "disableRefresh": false
    }
  ],
  "engine": "scala"
},
```


### GET Node Count

Gets the count of the processors.

An example request for getting count of the processors:

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/nodes/
↪count' -b /tmp/cookies.txt
```

An example response:

```
266
```

### GET Processors list for Engine

Gets the list of processors for the specified engine(scala or pyspark or empty-field for all).

An example request for getting list of processors for scala

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/nodes?
↪engine=scala' -b /tmp/cookies.txt
```

An example response:


```
  [
{
  "id": "3",
  "path": "/01-Connectors/",
  "name": "ReadCassandra",
  "iconImage": null,
  "description": "This node reads data from Apache Cassandra",
  "details": "",
  "examples": "",
  "type": "dataset",
  "nodeClass": "fire.nodes.cassandra.NodeReadCassandra",
  "x": null,
  "y": null,
  "fields": [
    {
      "name": "storageLevel",
      "value": "DEFAULT",
      "widget": "array",
      "title": "Output Storage Level",
      "description": "Storage Level of the Output Datasets of this Node",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": [
        "DEFAULT",
        "NONE",
        "DISK_ONLY",
        "DISK_ONLY_2",
        "MEMORY_ONLY",
        "MEMORY_ONLY_2",
        "MEMORY_ONLY_SER",
        "MEMORY_ONLY_SER_2",
        "MEMORY_AND_DISK",
        "MEMORY_AND_DISK_2",
        "MEMORY_AND_DISK_SER",
        "MEMORY_AND_DISK_SER_2",
        "OFF_HEAP"
      ],
      "required": false,
```


(continues on next page)

(continued from previous page)

```
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "table",
      "value": "",
      "widget": "textfield",
      "title": "Cassandra Table",
      "description": "Cassandra Table from which to read the data",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": true,
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "keyspace",
      "value": "",
      "widget": "textfield",
      "title": "Cassandra Keyspace",
      "description": "Cassandra Keyspace",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": true,
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "cluster",
      "value": "",
      "widget": "textfield",
      "title": "Cassandra Cluster",
      "description": "The group of the Cluster Level ",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": false,
      "display": true,
      "editable": true,
      "disableRefresh": false
    }
  ],
  "engine": "scala"
},
```

### GET Processor Details by Name

Gets Processor Details by Name

An example request for getting Processor Details by Name:

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/nodes/
↪names/ReadCSV' -b /tmp/cookies.txt
```

An example response:


```
{
 "id": "17",
 "path": "/02-ReadStructured/",
"name": "ReadCSV",
"iconImage": null,
"description": "It reads in CSV files and creates a DataFrame from it",
"details": "",
"examples": "",
"type": "dataset",
"nodeClass": "fire.nodes.dataset.NodeDatasetCSV",
"x": null,
"y": null,
"fields": [
{
  "name": "storageLevel",
  "value": "DEFAULT",
  "widget": "array",
  "title": "Output Storage Level",
  "description": "Storage Level of the Output Datasets of this Node",
  "optionsMap": null,
  "datatypes": null,
  "optionsArray": [
    "DEFAULT",
    "NONE",
    "DISK_ONLY",
    "DISK_ONLY_2",
    "MEMORY_ONLY",
    "MEMORY_ONLY_2",
    "MEMORY_ONLY_SER",
    "MEMORY_ONLY_SER_2",
    "MEMORY_AND_DISK",
    "MEMORY_AND_DISK_2",
    "MEMORY_AND_DISK_SER",
    "MEMORY_AND_DISK_SER_2",
    "OFF_HEAP"
  ],
  "required": false,
  "display": true,
  "editable": true,
  "disableRefresh": false
},
{
  "name": "path",
  "value": "",
  "widget": "textfield",
  "title": "Path",
  "description": "Path of the Text file/directory",
  "optionsMap": null,
  "datatypes": null,
  "optionsArray": null,
  "required": true,
  "display": true,
```


(continues on next page)

(continued from previous page)


```
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "separator",
      "value": ",",
      "widget": "textfield",
      "title": "Separator",
      "description": "CSV Separator",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": false,
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "header",
      "value": "false",
      "widget": "array",
      "title": "Header",
      "description": "Does the file have a header row",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": [
        "true",
        "false"
      ],
      "required": false,
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "dropMalformed",
      "value": "false",
      "widget": "array",
      "title": "Drop Malformed",
      "description": "Whether to drop Malformed records or error",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": [
        "true",
        "false"
      ],
      "required": false,
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "outputColNames",
      "value": "[]",
      "widget": "schema_col_names",
      "title": "Column Names for the CSV",
      "description": "New Output Columns of the SQL",
```


(continues on next page)

(continued from previous page)


```
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": false,
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "outputColTypes",
      "value": "[]",
      "widget": "schema_col_types",
      "title": "Column Types for the CSV",
      "description": "Data Type of the Output Columns",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": false,
      "display": true,
      "editable": true,
      "disableRefresh": false
    },
    {
      "name": "outputColFormats",
      "value": "[]",
      "widget": "schema_col_formats",
      "title": "Column Formats for the CSV",
      "description": "Format of the Output Columns",
      "optionsMap": null,
      "datatypes": null,
      "optionsArray": null,
      "required": false,
      "display": true,
      "editable": true,
      "disableRefresh": false
    }
  ],
  "engine": "all"
}
```


### Node Rules

Gets the node rules used in the workflow editor.

An example request for getting the node rules:

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/node-
↪rules' -b /tmp/cookies.txt
```

An example response:


```
  [
{
  "nodeType": "dataset",
  "possibleSources": [
```


(continues on next page)

(continued from previous page)


```
    "shellcommand"
  ],
  "minNumOfInputs": 0,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#F0F1F9",
  "nodeIcon": "fa-th-list",
  "nodeShape": "rectangle"
},
{
  "nodeType": "shellcommand",
  "possibleSources": [
    "dataset",
    "scala",
    "pyspark",
    "transform",
    "join",
    "ml-transformer",
    "ml-predict",
    "sparkstreaming"
  ],
  "minNumOfInputs": 0,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#F0F1F9",
  "nodeIcon": "fa-th-list",
  "nodeShape": "rectangle"
},
{
  "nodeType": "sparkstreaming",
  "possibleSources": [],
  "minNumOfInputs": 0,
  "maxNumOfInputs": 0,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#FFEB94",
  "nodeIcon": "fa-external-link",
  "nodeShape": "rectangle"
},
{
  "nodeType": "transform",
  "possibleSources": [
    "dataset",
    "scala",
    "pyspark",
    "transform",
    "join",
    "ml-transformer",
    "ml-predict",
    "sparkstreaming",
    "shellcommand"
  ],
  "minNumOfInputs": 1,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": null,
```


(continues on next page)

(continued from previous page)


```
  "sourceRestrictions": [],
  "backgroundColor": "#AFD4F0",
  "nodeIcon": "fa-tumblr-square",
  "nodeShape": "rectangle"
},
{
  "nodeType": "scala",
  "possibleSources": [
    "dataset",
    "transform",
    "join",
    "ml-transformer",
    "ml-predict",
    "sparkstreaming",
    "shellcommand"
  ],
  "minNumOfInputs": 0,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#AFD4F0",
  "nodeIcon": "fa-tumblr-square",
  "nodeShape": "rectangle"
},
{
  "nodeType": "pyspark",
  "possibleSources": [
    "dataset",
    "transform",
    "join",
    "ml-transformer",
    "ml-predict",
    "sparkstreaming",
    "shellcommand"
  ],
  "minNumOfInputs": 0,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#AFD4F0",
  "nodeIcon": "fa-tumblr-square",
  "nodeShape": "rectangle"
},
{
  "nodeType": "join",
  "possibleSources": [
    "dataset",
    "transform",
    "join",
    "shellcommand",
    "sparkstreaming"
  ],
  "minNumOfInputs": 2,
  "maxNumOfInputs": 8,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#D4A190",
```


(continues on next page)

(continued from previous page)


```
  "nodeIcon": "fa-stumbleupon",
  "nodeShape": "rectangle"
},
{
  "nodeType": "ml-transformer",
  "possibleSources": [
    "dataset",
    "transform",
    "ml-transformer",
    "join",
    "shellcommand"
  ],
  "minNumOfInputs": 1,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": "2",
  "sourceRestrictions": [],
  "backgroundColor": "#dfe166",
  "nodeIcon": "fa-qrcode",
  "nodeShape": "rectangle"
},
{
  "nodeType": "ml-estimator",
  "possibleSources": [
    "dataset",
    "transform",
    "ml-transformer",
    "join",
    "shellcommand"
  ],
  "minNumOfInputs": 1,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": "2",
  "sourceRestrictions": [],
  "backgroundColor": "#F7EFE2",
  "nodeIcon": "fa-qrcode",
  "nodeShape": "rectangle"
},
{
  "nodeType": "ml-predict",
  "possibleSources": [
    "dataset",
    "transform",
    "join",
    "ml-estimator",
    "ml-transformer",
    "ml-pipeline",
    "ml-crossvalidator",
    "ml-modelload"
  ],
  "minNumOfInputs": 1,
  "maxNumOfInputs": 2,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#D7CFC2",
  "nodeIcon": "fa-qrcode",
  "nodeShape": "rectangle"
},
```


(continues on next page)

(continued from previous page)


```
{
  "nodeType": "ml-evaluator",
  "possibleSources": [
    "ml-predict",
    "ml-estimator",
    "ml-pipeline"
  ],
  "minNumOfInputs": 1,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": "1",
  "sourceRestrictions": [],
  "backgroundColor": "#ff9900",
  "nodeIcon": "fa-qrcode",
  "nodeShape": "rectangle"
},
{
  "nodeType": "ml-pipeline",
  "possibleSources": [
    "ml-estimator",
    "ml-transformer"
  ],
  "minNumOfInputs": 1,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": "1",
  "sourceRestrictions": [],
  "backgroundColor": "#1FFF62",
  "nodeIcon": "fa-qrcode",
  "nodeShape": "rectangle"
},
{
  "nodeType": "ml-crossvalidator",
  "possibleSources": [
    "ml-evaluator"
  ],
  "minNumOfInputs": 1,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#F9FC81",
  "nodeIcon": "fa-qrcode",
  "nodeShape": "rectangle"
},
{
  "nodeType": "ml-trainvalidationsplit",
  "possibleSources": [
    "ml-evaluator"
  ],
  "minNumOfInputs": 1,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#B681FC",
  "nodeIcon": "fa-qrcode",
  "nodeShape": "rectangle"
},
{
  "nodeType": "ml-modelsave",
```


(continues on next page)

(continued from previous page)


```
  "possibleSources": [
    "ml-estimator",
    "ml-pipeline",
    "ml-crossvalidator",
    "ml-trainvalidationsplit"
  ],
  "minNumOfInputs": 1,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": "1",
  "sourceRestrictions": [],
  "backgroundColor": "#FCB881",
  "nodeIcon": "fa-qrcode",
  "nodeShape": "rectangle"
},
{
  "nodeType": "ml-modelload",
  "possibleSources": [
    "dataset",
    "transform",
    "join",
    "ml-estimator",
    "ml-transformer",
    "ml-pipeline",
    "ml-crossvalidator",
    "ml-modelsave"
  ],
  "minNumOfInputs": 0,
  "maxNumOfInputs": 1,
  "maxNumOfOutputs": "1",
  "sourceRestrictions": [],
  "backgroundColor": "#FCB881",
  "nodeIcon": "fa-qrcode",
  "nodeShape": "rectangle"
},
{
  "nodeType": "doc",
  "possibleSources": [
    "doc"
  ],
  "minNumOfInputs": 0,
  "maxNumOfInputs": 0,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#FFFF88",
  "nodeIcon": "fa-file-text",
  "nodeShape": "rectangle"
},
{
  "nodeType": "sticky",
  "possibleSources": [],
  "minNumOfInputs": 0,
  "maxNumOfInputs": 0,
  "maxNumOfOutputs": null,
  "sourceRestrictions": [],
  "backgroundColor": "#FFFF88",
  "nodeIcon": "fa-file-text",
  "nodeShape": "rectangle"
```


(continues on next page)

(continued from previous page)

```
},
```

### 29.1.2 Datasets REST API

#### Overview

The Dataset REST APIs, allow you to manage the Datasets.

Below are the various Dataset APIs available in Fire Insights, They should be executed after you have logged into Fire Insights.

#### GET List of Datasets by Application

Returns the list of Datasets for the logged in user for a given application id:

```
curl -X GET --header 'Accept: application/json' --header 'api_key: cookies' 'http://
↪localhost:8080/api/v1/datasets?sortPara=dsc&projectId=1'
```

#### Create / Update Dataset

If id value is not passed, new dataset will be created:

#### JSON

```
{
  "id": 13,
  "version": 0,
  "name": "spam",
  "header": true,
  "path": "data\/spam.csv",
  "delimiter": ",",
  "schemaModel": {
    "schemaColList": [
      {
        "colName": "label",
        "colType": "DOUBLE",
        "colFormat": "",
        "colMLType": "NUMERIC"
      },
      {
        "colName": "message",
        "colType": "STRING",
        "colFormat": "",
        "colMLType": "TEXT"
      },
      {
        "colName": "id",
        "colType": "DOUBLE",
        "colFormat": "",
        "colMLType": "NUMERIC"
```

(continues on next page)

(continued from previous page)

```
        }
      ]
    }
}
```

### Curl

```
curl-X POST --header 'Content-Type: application/json' --header 'Accept: /' -d     '{
↪"id":13,"version":0,"name":"spam","header":true,"path":"data/spam.csv","delimiter":
↪",","schemaModel":{"schemaColList":[{"colName":"label","colType":"DOUBLE","colFormat
↪":"","colMLType":"NUMERIC"},{"colName":"message","colType":"STRING","colFormat":"",
↪"colMLType":"TEXT"},{"colName":"id","colType":"DOUBLE","colFormat":"","colMLType":
↪"NUMERIC"}]}}' localhost:8080/dataset/save -b /tmp/cookies.txt
```

### Delete Dataset

- “datasetId”: “98”
- “projectId”: “33”

An example request for Deleting dataset:

```
curl -X DELETE --header 'Accept: text/plain' 'http://localhost:8080/api/v1/datasets/
↪98?projectId=33'
```

An example response:

```
Dataset with id 98 deleted successfully
```

### Get Dataset by Id

- “datasetId”: “65”
- “projectId”: “33”

An example request for Getting dataset by Id:

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/
↪datasets/65?projectId=33'
```

An example response:

```
{
   "id": 65,
   "userId": 33,
   "uuid": "1e13ec2a-4094-405e-a6e7-ffed3bd027f7",
   "version": 0,
   "name": "Test-dataset",
   "category": null,
   "description": "Test",
   "header": true,
   "readOptions": null,
   "path": "/user/sparkflows/Clickthru.csv",
```

(continues on next page)

(continued from previous page)

```
   "delimiter": ",",
   "datasetType": "CSV",
   "filterLinesContaining": null,
   "datasetSchema": "{colNames:[\"Timestamp\",\"UserId\",\"IP Address\",\"Product Id\
↪"],colTypes:[\"STRING\",\"INTEGER\",\"STRING\",\"INTEGER\"],colFormats:[\"\",\"\",\
↪"\",\"\"],colMLTypes:[\"TEXT\",\"NUMERIC\",\"TEXT\",\"NUMERIC\"]}",
   "dateCreated": 1566880637842,
   "dateLastUpdated": 1566880637846,
   "permission": null,
   "readOptionsModel": null,
   "schemaModel": {
    "schemaColList": [
    {
      "colName": "Timestamp",
      "colType": "STRING",
      "colFormat": "",
      "colMLType": "TEXT"
    },
    {
      "colName": "UserId",
      "colType": "INTEGER",
      "colFormat": "",
      "colMLType": "NUMERIC"
    },
    {
      "colName": "IP Address",
      "colType": "STRING",
      "colFormat": "",
      "colMLType": "TEXT"
    },
    {
      "colName": "Product Id",
      "colType": "INTEGER",
      "colFormat": "",
      "colMLType": "NUMERIC"
    }
  ]
  },
    "sampleData": {
    "headers": [
    "Timestamp",
    "UserId",
    "IP Address",
    " Product Id"
  ],
  "cells": [
    [
      "9:03 AM",
      "275",
      "207.51.113.192",
      "1"
    ],
    [
      "12:57 AM",
      "586",
      "62.34.98.94",
      "2"
```

(continues on next page)

(continued from previous page)


```
    ],
    [
      "2:45 AM",
      "508",
      "20.237.172.182",
      "3"
    ],
    [
      "2:13 PM",
      "378",
      "69.215.255.150",
      "4"
    ],
    [
      "9:27 AM",
      "965",
      "56.101.183.251",
      "5"
    ],
    [
      "8:18 AM",
      "263",
      "9.151.97.180",
      "6"
    ],
    [
      "9:40 AM",
      "670",
      "101.195.1.186",
      "7"
    ],
    [
      "7:14 AM",
      "447",
      "232.29.216.95",
      "8"
    ],
    [
      "12:57 AM",
      "33",
      "85.119.50.62",
      "9"
    ],
    [
      "12:56 AM",
      "589",
      "185.132.243.178",
      "10"
    ],
    [
      "11:04 PM",
      "22",
      "120.212.232.218",
      "11"
    ],
    [
      "8:29 PM",
```


(continues on next page)

(continued from previous page)

```
      "504",
      "226.70.25.117",
      "12"
    ],
    [
      "5:18 PM",
      "228",
      "213.53.100.18",
      "13"
    ],
    [
      "2:56 PM",
      "536",
      "60.65.25.167",
      "14"
    ],
    [
      "3:57 AM",
      "46",
      "149.156.17.120",
      "15"
    ],
    [
      "8:05 AM",
      "812",
      "23.213.182.107",
      "16"
    ],
    [
      "12:02 PM",
      "980",
      "93.20.165.16",
      "17"
    ],
    [
      "12:53 PM",
      "915",
      "24.180.112.147",
      "18"
    ],
    [
      "11:32 AM",
      "814",
      "110.81.139.11",
      "19"
    ],
    [
      "11:01 PM",
      "429",
      "115.123.246.193",
      "20"
    ]
  ]
  },
"json": "{\"id\":65,\"userId\":33,\"uuid\":\"1e13ec2a-4094-405e-a6e7-ffed3bd027f7\",\
↪"version\":0,\"name\":\"Test-dataset\",\"description\":\"Test\",\"header\":true,\
↪"path\":\"/user/sparkflows/Clickthru.csv\",\"delimiter\":\",\",\"datasetType\":\
↪"CSV\",\"datasetSchema\":\"{colNames:[\\\"Timestamp\\\",\\\"UserId\\\",\\\"IP
↪Address\\\",\\\"Product Id\\\"],colTypes:[\\\"STRING\\\",\\\"INTEGER\\\",\\\
↪"STRING\\\",\\\"INTEGER\\\"],colFormats:[\\\"\\\",\\\"\\\",\\\"\\\",\\\"\\\"],
↪colMLTypes:[\\\"TEXT\\\",\\\"NUMERIC\\\",\\\"TEXT\\\",\\\"NUMERIC\\\"]}\",\
↪"dateCreated\":\"Aug 27, 2019 4:37:17 AM\",\"dateLastUpdated\":\"Aug 27, 2019
↪4:37:17 AM\",\"schemaModel\":{\"schemaColList\":[{\"colName\":\"Timestamp\",\
↪"colType\":\"STRING\",\"colFormat\":\"\",\"colMLType\":\"TEXT\"},{\"colName\":\
```

(continues on next page)

(continued from previous page)

```
"projectId": 33
 },
```

### Get Dataset Count

Returns the count of datasets available:

```
curl -X GET --header 'Accept: application/json' --header 'api_key: cookies' 'http://
↪localhost:8080/api/v1/datasets/count'
```

### Get sample data

Delimiter and header are optional values

- path: data/spam.csv
- schema: {“colNames”:[“0.0”,”this is not a spam”,”3.0”],”colTypes”:[“DOUBLE”,”STRING”,”DOUBLE”],”colFormats”:[“”,””,””

CURL:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
↪json' --header 'api_key: cookies' -d
'{"colNames":["0.0","this is not a spam","3.0"],"colTypes":["DOUBLE","STRING","DOUBLE
↪"],"colFormats":["","",""],"colMLTypes":["NUMERIC","TEXT","NUMERIC"]}' http://
↪localhost:8080/api/v1/datasets/sample-data
```

### Returns schema of the files in the given path using the given delimiter

- delimiter and header are optional values
- path:data/spam.csv
- schema: {“colNames”:[“0.0”,”this is not a spam”,”3.0”],”colTypes”:[“DOUBLE”,”STRING”,”DOUBLE”],”colFormats”:[“”,””,””

CURL:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
↪json' --header 'api_key: cookies' -d
'{"colNames":["0.0","this is not a spam","3.0"],"colTypes":["DOUBLE","STRING","DOUBLE
↪"],"colFormats":["","",""],"colMLTypes":["NUMERIC","TEXT","NUMERIC"]}' http://
↪localhost:8080/api/v1/datasets/schema
```

### Get Latest Five Datasets

Returns the latest updated datasets:

```
curl -X GET --header 'Accept: application/json' --header 'api_key: cookies' 'http://
↪localhost:8080/api/v1/datasets/latest'
```

#### Get the list of files/directories in the given path

- path:data/transaction.csv

CURL:

```
curl   -X GET --header 'Content-Type: application/json' --header 'Accept: application/
↪json' -d 'data/transaction.csv' 'http://localhost:8080/filesInPathJSON -b /tmp/
↪cookies.txt'
```

### 29.1.3 Workflow REST API

The Workflow REST API's, allow you to interact with the Workflows.

Below are the various Workflow API's available in Fire Insights. They should be executed after you have logged into Fire Insights.

#### Create Workflow

Create a new Workflow.

An example request for creating workflow:

```
  curl -X POST --header 'Content-Type: application/json' --header 'Accept:␣
↪application/json' -d '{
"analysisflowId": 1,
"comment": "string",
"projectId": 33,
"workflow": {
  "category": "string",
  "dataSetDetails": [
    {
      "datasetSchema": "string",
      "datasetType": "CSV",
      "delimiter": "string",
      "description": "string",
      "filterLinesContaining": "string",
      "header": true,
      "id": 0,
      "name": "string",
      "path": "string",
      "readOptions": "string",
      "uuid": "string",
      "version": 0
    }
  ],
  "description": "string",
  "edges": [
    {
      "id": 0,
      "source": "string",
      "target": "string"
    }
  ],
  "engine": "string",
  "h2OWorkflow": true,
```

(continues on next page)

(continued from previous page)

```
  "name": "string",
  "nodes": [
    {
      "description": "string",
      "details": "string",
      "engine": "string",
      "examples": "string",
      "fields": [
        {
          "datatypes": [
            "string"
          ],
          "description": "string",
          "disableRefresh": true,
          "display": true,
          "editable": true,
          "name": "string",
          "optionsArray": [
            "string"
          ],
          "optionsMap": {},
          "required": true,
          "title": "string",
          "value": "string",
          "widget": "string"
        }
      ],
      "iconImage": "string",
      "id": "string",
      "name": "string",
      "nodeClass": "string",
      "path": "string",
      "type": "string",
      "x": "string",
      "y": "string"
    }
  ],
  "parameters": "string",
  "uuid": "string"
}
}' 'http://hostname:port/api/v1/workflows' -b /tmp/cookies.txt
```

An example response:

```
193
```

### Execute Workflow

Execute specified Workflow.

An example request for Executing specified workflow:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
↪json' -d '{
"emailOnFailure": "string",
```

(continues on next page)

(continued from previous page)

```
"emailOnSuccess": "string",
"libJars": "string",
"programParameters": "string",
"sparkConfig": "string",
"workflowId": 1
}' 'http://hostname:port/api/v1/workflow/execute' -b /tmp/cookies.txt
```

An example response:

```
162
```

### Update Workflow

Update specified Workflow.

An example request for updating workflow:

```
 curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/
↪json' -d '{
"analysisflowId": 1,
"comment": "string",
"projectId": 33,
"workflow": {
  "category": "string",
  "dataSetDetails": [
    {
      "datasetSchema": "string",
      "datasetType": "CSV",
      "delimiter": "string",
      "description": "string",
      "filterLinesContaining": "string",
      "header": true,
      "id": 0,
      "name": "string",
      "path": "string",
      "readOptions": "string",
      "uuid": "string",
      "version": 0
    }
  ],
  "description": "string",
  "edges": [
    {
      "id": 0,
      "source": "string",
      "target": "string"
    }
  ],
  "engine": "string",
  "h2OWorkflow": true,
  "name": "string",
  "nodes": [
    {
      "description": "string",
      "details": "string",
```

(continues on next page)

(continued from previous page)

```
      "engine": "string",
      "examples": "string",
      "fields": [
        {
          "datatypes": [
            "string"
          ],
          "description": "string",
          "disableRefresh": true,
          "display": true,
          "editable": true,
          "name": "string",
          "optionsArray": [
            "string"
          ],
          "optionsMap": {},
          "required": true,
          "title": "string",
          "value": "string",
          "widget": "string"
        }
      ],
      "iconImage": "string",
      "id": "string",
      "name": "string",
      "nodeClass": "string",
      "path": "string",
      "type": "string",
      "x": "string",
      "y": "string"
    }
  ],
  "parameters": "string",
  "uuid": "string"
}
}' 'http://hostname:port/api/v1/workflows' -b /tmp/cookies.txt
```

An example response:

```
131
```

### Get workflow by Id

Gets the workflow with the specified id.

- id: 1

An example request for getting workflow by id:

```
curl -X GET --header 'Accept: application/json' 'http://hostname:port/api/v1/
↪workflows/id/1' -b /tmp/cookies.txt
```

An example response:

```
  {
"id": 1,
```

(continues on next page)

(continued from previous page)


```
"userId": 1,
"uuid": "3a3dfa34-bbd7-4c05-8745-55628d90cbf6",
"name": "Analyze Flights Delay",
"category": "Analytics",
"content": "{\"name\":\"Analyze Flights Delay\",\"uuid\":\"3a3dfa34-bbd7-4c05-8745-
↪55628d90cbf6\",\"category\":\"Analytics\",\"description\":\"Find Flights which are
↪delayed by more than 40 minutes.\",\"nodes\":[{\"id\":\"1\",\"name\":\
↪"DatasetStructured\",\"description\":\"This Node creates a DataFrame by reading
↪data from HDFS, HIVE etc. The dataset has been defined earlier in Fire by using the
↪Dataset Feature. As a user, you just have to select the Dataset of your interest.\",
↪\"details\":\"This Node creates a DataFrame by reading data from HDFS, HIVE etc.
↪\\u003cbr\\u003e\\n\\u003cbr\\u003e\\nThe data has been defined earlier in Fire by
↪using the Dataset Feature. As a user, you just have to select the Dataset of your
↪interest.\\u003cbr\\u003e\",\"examples\":\"\",\"type\":\"dataset\",\"nodeClass\":\
↪"fire.nodes.dataset.NodeDatasetStructured\",\"x\":\"38.9492px\",\"y\":\"275.613px\",
↪\"fields\":[{\"name\":\"storageLevel\",\"value\":\"DEFAULT\",\"widget\":\"array\",\
↪"title\":\"Output Storage Level\",\"description\":\"Storage Level of the Output
↪Datasets of this Node\",\"optionsArray\":[\"DEFAULT\",\"NONE\",\"DISK_ONLY\",\"DISK_
↪ONLY_2\",\"MEMORY_ONLY\",\"MEMORY_ONLY_2\",\"MEMORY_ONLY_SER\",\"MEMORY_ONLY_SER_2\
↪",\"MEMORY_AND_DISK\",\"MEMORY_AND_DISK_2\",\"MEMORY_AND_DISK_SER\",\"MEMORY_AND_
↪DISK_SER_2\",\"OFF_HEAP\"],\"required\":false,\"display\":true,\"editable\":true,\
↪"disableRefresh\":false},{\"name\":\"dataset\",\"value\":\"2ff32692-9b3c-49de-91a7-
↪401daf2590c1\",\"widget\":\"dataset\",\"title\":\"Dataset\",\"description\":\
↪"Selected Dataset\",\"required\":true,\"display\":true,\"editable\":true,\
↪"disableRefresh\":false}],\"engine\":\"scala\"},{\"id\":\"2\",\"name\":\"PrintNRows\
↪",\"description\":\"Prints the specified number of records in the DataFrame. It is
↪useful for seeing intermediate output\",\"details\":\"\",\"examples\":\"\",\"type\
↪":\"transform\",\"nodeClass\":\"fire.nodes.util.NodePrintFirstNRows\",\"x\":\"38.
↪4336px\",\"y\":\"59.1094px\",\"fields\":[{\"name\":\"storageLevel\",\"value\":\
↪"DEFAULT\",\"widget\":\"array\",\"title\":\"Output Storage Level\",\"description\":\
↪"Storage Level of the Output Datasets of this Node\",\"optionsArray\":[\"DEFAULT\",\
↪"NONE\",\"DISK_ONLY\",\"DISK_ONLY_2\",\"MEMORY_ONLY\",\"MEMORY_ONLY_2\",\"MEMORY_
↪ONLY_SER\",\"MEMORY_ONLY_SER_2\",\"MEMORY_AND_DISK\",\"MEMORY_AND_DISK_2\",\"MEMORY_
↪AND_DISK_SER\",\"MEMORY_AND_DISK_SER_2\",\"OFF_HEAP\"],\"required\":false,\"display\
↪":true,\"editable\":true,\"disableRefresh\":false},{\"name\":\"title\",\"value\":\
↪"Row Values\",\"widget\":\"textfield\",\"title\":\"Title\",\"required\":false,\
↪"display\":true,\"editable\":true,\"disableRefresh\":false},{\"name\":\"n\",\"value\
↪":\"10\",\"widget\":\"textfield\",\"title\":\"Num Rows to Print\",\"description\":\
↪"number of rows to be printed\",\"required\":false,\"display\":true,\"editable\
↪":true,\"disableRefresh\":false}],\"engine\":\"scala\"},{\"id\":\"3\",\"name\":\
↪"CastColumnType\",\"description\":\"This node creates a new DataFrame by casting
↪input columns with a new data type\",\"details\":\"\",\"examples\":\"\",\"type\":\
↪"transform\",\"nodeClass\":\"fire.nodes.etl.NodeCastColumnType\",\"x\":\"313.223px\
↪",\"y\":\"61.8633px\",\"fields\":[{\"name\":\"storageLevel\",\"value\":\"DEFAULT\",\
↪"widget\":\"array\",\"title\":\"Output Storage Level\",\"description\":\"Storage
↪Level of the Output Datasets of this Node\",\"optionsArray\":[\"DEFAULT\",\"NONE\",\
↪"DISK_ONLY\",\"DISK_ONLY_2\",\"MEMORY_ONLY\",\"MEMORY_ONLY_2\",\"MEMORY_ONLY_SER\",\
↪"MEMORY_ONLY_SER_2\",\"MEMORY_AND_DISK\",\"MEMORY_AND_DISK_2\",\"MEMORY_AND_DISK_
↪SER\",\"MEMORY_AND_DISK_SER_2\",\"OFF_HEAP\"],\"required\":false,\"display\":true,\
↪"editable\":true,\"disableRefresh\":false},{\"name\":\"inputCols\",\"value\":\"[\\\
↪"CRS_DEP_TIME\\\",\\\"CRS_ARR_TIME\\\",\\\"CRS_ELAPSED_TIME\\\"]\",\"widget\":\
↪"variables\",\"title\":\"Columns\",\"description\":\"Columns to be cast to new data
↪type\",\"required\":false,\"display\":true,\"editable\":true,\"disableRefresh\
↪":false},{\"name\":\"outputColType\",\"value\":\"DOUBLE\",\"widget\":\"array\",\
↪"title\":\"New Data Type\",\"description\":\"New data type(INTEGER, DOUBLE, STRING,
↪LONG, SHORT)\",\"optionsArray\":[\"BOOLEAN\",\"BYTE\",\"DATE\",\"DECIMAL\",\"DOUBLE\
↪",\"FLOAT\",\"INTEGER\",\"LONG\",\"SHORT\",\"STRING\",\"TIMESTAMP\"],\"required\
↪":false,\"display\":true,\"editable\":true,\"disableRefresh\":false},{\"name\":\
↪"replaceExistingCols\",\"value\":\"true\",\"widget\":\"array\",\"title\":\"Replace
↪Existing Cols\",\"description\":\"Whether to replace existing columns or create new
↪ones\",\"optionsArray\":[\"true\",\"false\"],\"required\":false,\"display\":true,\
↪"editable\":true,\"disableRefresh\":false}],\"engine\":\"scala\"},{\"id\":\"4\",\
↪"name\":\"CastColumnType\",\"description\":\"This node creates a new DataFrame by
↪casting input columns with a new data type\",\"details\":\"\",\"examples\":\"\",\
```


(continues on next page)

(continued from previous page)

```
"description": "Find Flights which are delayed by more than 40 minutes.",
"version": 1,
"dateCreated": 1566551544583,
"dateLastUpdated": 1566551544583,
"lockedByUserId": null,
"permission": null,
"workflow": {
  "name": "Analyze Flights Delay",
  "uuid": "3a3dfa34-bbd7-4c05-8745-55628d90cbf6",
  "category": "Analytics",
  "description": "Find Flights which are delayed by more than 40 minutes.",
  "parameters": null,
  "nodes": [
    {
      "id": "1",
      "path": null,
      "name": "DatasetStructured",
      "iconImage": null,
      "description": "This Node creates a DataFrame by reading data from HDFS, HIVE␣
↪etc. The dataset has been defined earlier in Fire by using the Dataset Feature. As␣
↪a user, you just have to select the Dataset of your interest.",
      "details": "This Node creates a DataFrame by reading data from HDFS, HIVE etc.
↪<br>\n<br>\nThe data has been defined earlier in Fire by using the Dataset Feature.␣
↪As a user, you just have to select the Dataset of your interest.<br>",
      "examples": "",
      "type": "dataset",
      "nodeClass": "fire.nodes.dataset.NodeDatasetStructured",
      "x": "38.9492px",
      "y": "275.613px",
      "fields": [
        {
          "name": "storageLevel",
          "value": "DEFAULT",
          "widget": "array",
          "title": "Output Storage Level",
          "description": "Storage Level of the Output Datasets of this Node",
          "optionsMap": null,
          "datatypes": null,
          "optionsArray": [
            "DEFAULT",
            "NONE",
            "DISK_ONLY",
            "DISK_ONLY_2",
            "MEMORY_ONLY",
            "MEMORY_ONLY_2",
            "MEMORY_ONLY_SER",
            "MEMORY_ONLY_SER_2",
            "MEMORY_AND_DISK",
            "MEMORY_AND_DISK_2",
            "MEMORY_AND_DISK_SER",
            "MEMORY_AND_DISK_SER_2",
            "OFF_HEAP"
          ],
          "required": false,
          "display": true,
          "editable": true,
          "disableRefresh": false
```

(continues on next page)

(continued from previous page)

```
        },
        {
          "name": "dataset",
          "value": "2ff32692-9b3c-49de-91a7-401daf2590c1",
          "widget": "dataset",
          "title": "Dataset",
          "description": "Selected Dataset",
          "optionsMap": null,
          "datatypes": null,
          "optionsArray": null,
          "required": true,
          "display": true,
          "editable": true,
          "disableRefresh": false
        }
      ],
      "engine": "scala"
    },
```

### Delete Workflow

Deletes a workflow with the given workflowId.

- workflowId: 1955

An example request for deleting workflow:

```
curl -X DELETE --header 'Accept: application/json' 'http://localhost:8080/api/v1/
↪workflows/id/1955' -b /tmp/cookies.txt
```

An example response:

```
Workflow deleted successfully.
```

### Get Latest WorkFlows

Gets the latest workFlows available in the given application:

An example request for getting Latest WorkFlows availble in application:

```
curl -X GET --header 'Accept: application/json' 'http://hostname:port/api/v1/
↪workflows/latest' -b /tmp/cookies.txt
```

An example response:

```
{
"id": 1954,
"userId": 3,
"uuid": "0e119cf1-2833-4c62-8466-21853fc1fb21",
"name": "aaaaawqw",
"category": "-",
"content": "{\"name\":\"aaaaawqw\",\"uuid\":\"0e119cf1-2833-4c62-8466-21853fc1fb21\",\
↪"category\":\"-\",\"description\":\"1111\",\"parameters\":\"2222@1111\",\"nodes\":[
↪{\"id\":\"1\",\"name\":\"ReadCSV\",\"description\":\"It reads in CSV files and␣
↪creates a DataFrame from it\",\"details\":\"\",\"examples\":\"\",\"type\":\"dataset\
↪",\"nodeClass\":\"fire.nodes.dataset.NodeDatasetCSV\",\"x\":\"243.5px\",\"y\":\
↪"206px\",\"fields\":[{\"name\":\"storageLevel\",\"value\":\"DEFAULT\",\"widget\":\
↪"array\",\"title\":\"Output Storage Level\",\"description\":\"Storage Level of the␣
↪Output Datasets of this Node\",\"optionsArray\":[\"DEFAULT\",\"NONE\",\"DISK_ONLY\",
↪\"DISK_ONLY_2\",\"MEMORY_ONLY\",\"MEMORY_ONLY_2\",\"MEMORY_ONLY_SER\",\"MEMORY_ONLY_
↪SER_2\",\"MEMORY_AND_DISK\",\"MEMORY_AND_DISK_2\",\"MEMORY_AND_DISK_SER\",\"MEMORY_
↪AND_DISK_SER_2\",\"OFF_HEAP\"],\"required\":false,\"display\":true,\"editable\
↪":true,\"disableRefresh\":false},{\"name\":\"path\",\"value\":\"/user/sparkflows/
```

(continues on next page)

(continued from previous page)


```
"description": "1111",
"version": 4,
"dateCreated": 1566395460079,
"dateLastUpdated": 1566395644690,
"lockedByUserId": null,
"permission": null,
"workflow": {
  "name": "aaaaawqw",
  "uuid": "0e119cf1-2833-4c62-8466-21853fc1fb21",
  "category": "-",
  "description": "1111",
  "parameters": "2222@1111",
  "nodes": [
    {
      "id": "1",
      "path": null,
      "name": "ReadCSV",
      "iconImage": null,
      "description": "It reads in CSV files and creates a DataFrame from it",
      "details": "",
      "examples": "",
      "type": "dataset",
      "nodeClass": "fire.nodes.dataset.NodeDatasetCSV",
      "x": "243.5px",
      "y": "206px",
      "fields": [
        {
          "name": "storageLevel",
          "value": "DEFAULT",
          "widget": "array",
          "title": "Output Storage Level",
          "description": "Storage Level of the Output Datasets of this Node",
          "optionsMap": null,
          "datatypes": null,
          "optionsArray": [
            "DEFAULT",
            "NONE",
            "DISK_ONLY",
            "DISK_ONLY_2",
            "MEMORY_ONLY",
            "MEMORY_ONLY_2",
            "MEMORY_ONLY_SER",
            "MEMORY_ONLY_SER_2",
            "MEMORY_AND_DISK",
            "MEMORY_AND_DISK_2",
            "MEMORY_AND_DISK_SER",
            "MEMORY_AND_DISK_SER_2",
            "OFF_HEAP"
          ],
          "required": false,
          "display": true,
          "editable": true,
          "disableRefresh": false
        },
        {
          "name": "path",
          "value": "/user/sparkflows/Clickthru.csv",
```


(continues on next page)

(continued from previous page)


```
        "widget": "textfield",
        "title": "Path",
        "description": "Path of the Text file/directory",
        "optionsMap": null,
        "datatypes": null,
        "optionsArray": null,
        "required": true,
        "display": true,
        "editable": true,
        "disableRefresh": false
      },
      {
        "name": "separator",
        "value": ",",
        "widget": "textfield",
        "title": "Separator",
        "description": "CSV Separator",
        "optionsMap": null,
        "datatypes": null,
        "optionsArray": null,
        "required": false,
        "display": true,
        "editable": true,
        "disableRefresh": false
      },
      {
        "name": "header",
        "value": "true",
        "widget": "array",
        "title": "Header",
        "description": "Does the file have a header row",
        "optionsMap": null,
        "datatypes": null,
        "optionsArray": [
          "true",
          "false"
        ],
        "required": false,
        "display": true,
        "editable": true,
        "disableRefresh": false
      },
      {
        "name": "dropMalformed",
        "value": "false",
        "widget": "array",
        "title": "Drop Malformed",
        "description": "Whether to drop Malformed records or error",
        "optionsMap": null,
        "datatypes": null,
        "optionsArray": [
          "true",
          "false"
        ],
        "required": false,
        "display": true,
        "editable": true,
```


(continues on next page)

(continued from previous page)


```
        "disableRefresh": false
      },
      {
        "name": "outputColNames",
        "value": "[\"Timestamp\",\"UserId\",\"IP Address\",\" Product Id\"]",
        "widget": "schema_col_names",
        "title": "Column Names for the CSV",
        "description": "New Output Columns of the SQL",
        "optionsMap": null,
        "datatypes": null,
        "optionsArray": null,
        "required": false,
        "display": true,
        "editable": true,
        "disableRefresh": false
      },
      {
        "name": "outputColTypes",
        "value": "[\"STRING\",\"INTEGER\",\"STRING\",\"INTEGER\"]",
        "widget": "schema_col_types",
        "title": "Column Types for the CSV",
        "description": "Data Type of the Output Columns",
        "optionsMap": null,
        "datatypes": null,
        "optionsArray": null,
        "required": false,
        "display": true,
        "editable": true,
        "disableRefresh": false
      },
      {
        "name": "outputColFormats",
        "value": "[\"\",\"\",\"\",\"\"]",
        "widget": "schema_col_formats",
        "title": "Column Formats for the CSV",
        "description": "Format of the Output Columns",
        "optionsMap": null,
        "datatypes": null,
        "optionsArray": null,
        "required": false,
        "display": true,
        "editable": true,
        "disableRefresh": false
      }
    ],
    "engine": "all"
  },
```


### Get Workflow Count

Gets the count of the workflows in the given application.

An example request for getting count of the Workflow:

```
curl -X GET --header 'Accept: application/json' 'http://hostname:port/api/v1/
↪workflows/count' -b /tmp/cookies.txt
```

An example response:

```
92
```

### Get Workflow Versions

Gets the versions of workflow.

- workflowId: 1

An example request for getting workflow by id:

```
curl -X GET --header 'Accept: application/json' 'http://hostname:port/api/v1/
↪workflows/versions?workflowId=1' -b /tmp/cookies.txt
```

An example response:


```
[
{
 "id": 1,
 "analysisflowId": 1,
 "content": "{\"name\":\"Analyze Flights Delay\",\"uuid\":\"3a3dfa34-bbd7-4c05-8745-
↪55628d90cbf6\",\"category\":\"Analytics\",\"description\":\"Find Flights which are␣
↪delayed by more than 40 minutes.\",\"nodes\":[{\"id\":\"1\",\"name\":\
↪"DatasetStructured\",\"type\":\"dataset\",\"nodeClass\":\"fire.nodes.dataset.
↪NodeDatasetStructured\",\"x\":\"38.9492px\",\"y\":\"275.613px\",\"fields\":[{\"name\
↪":\"dataset\",\"value\":\"2ff32692-9b3c-49de-91a7-401daf2590c1\",\"widget\":\
↪"dataset\",\"title\":\"Dataset\",\"description\":\"Selected Dataset\",\"required\
↪":false,\"display\":true,\"editable\":true,\"disableRefresh\":false}]},{\"id\":\"2\
↪",\"name\":\"PrintNRows\",\"description\":\"Prints the specified number of records␣
↪in the DataFrame\",\"type\":\"transform\",\"nodeClass\":\"fire.nodes.util.
↪NodePrintFirstNRows\",\"x\":\"38.4336px\",\"y\":\"59.1094px\",\"fields\":[{\"name\
↪":\"n\",\"value\":\"10\",\"widget\":\"textfield\",\"title\":\"Num Rows to Print\",\
↪"description\":\"number of rows to be printed\",\"required\":false,\"display\
↪":false,\"editable\":true,\"disableRefresh\":false}]},{\"id\":\"3\",\"name\":\
↪"CastColumnType\",\"description\":\"This node creates a new DataFrame by casting␣
↪input columns with a new data type\",\"type\":\"transform\",\"nodeClass\":\"fire.
↪nodes.etl.NodeCastColumnType\",\"x\":\"313.223px\",\"y\":\"61.8633px\",\"fields\":[
↪{\"name\":\"inputCols\",\"value\":\"[\\\"CRS_DEP_TIME\\\",\\\"CRS_ARR_TIME\\\",\\\
↪"CRS_ELAPSED_TIME\\\"]\",\"widget\":\"variables\",\"title\":\"Columns\",\
↪"description\":\"Columns to be cast to new data type\",\"required\":false,\"display\
↪":false,\"editable\":true,\"disableRefresh\":false},{\"name\":\"outputColType\",\
↪"value\":\"DOUBLE\",\"widget\":\"array\",\"title\":\"New Data Type\",\"description\
↪":\"New data type(INTEGER, DOUBLE, STRING, LONG, SHORT)\",\"optionsArray\":[\
↪"BOOLEAN\",\"BYTE\",\"DATE\",\"DOUBLE\",\"FLOAT\",\"INTEGER\",\"LONG\",\"SHORT\",\
↪"STRING\",\"TIMESTAMP\"],\"required\":false,\"display\":false,\"editable\":true,\
↪"disableRefresh\":false}]},{\"id\":\"4\",\"name\":\"CastColumnType\",\"description\
↪":\"This node creates a new DataFrame by casting input columns with a new data type\
↪",\"type\":\"transform\",\"nodeClass\":\"fire.nodes.etl.NodeCastColumnType\",\"x\":\
↪"322.949px\",\"y\":\"275.633px\",\"fields\":[{\"name\":\"inputCols\",\"value\":\
↪"[\\\"DAY_OF_MONTH\\\",\\\"DAY_OF_WEEK\\\"]\",\"widget\":\"variables\",\"title\":\
↪"Columns\",\"description\":\"Columns to be cast to new data type\",\"required\
↪":false,\"display\":false,\"editable\":true,\"disableRefresh\":false},{\"name\":\
↪"outputColType\",\"value\":\"STRING\",\"widget\":\"array\",\"title\":\"New Data␣
↪Type\",\"description\":\"New data type(INTEGER, DOUBLE, STRING, LONG, SHORT)\",\
↪"optionsArray\":[\"BOOLEAN\",\"BYTE\",\"DATE\",\"DOUBLE\",\"FLOAT\",\"INTEGER\",\
↪"LONG\",\"SHORT\",\"STRING\",\"TIMESTAMP\"],\"required\":false,\"display\":false,\
↪"editable\":true,\"disableRefresh\":false}]},{\"id\":\"5\",\"name\":\"StringIndexer\
↪",\"description\":\"StringIndexer encodes a string column of labels to a column of␣
↪label indices\",\"type\":\"ml-transformer\",\"nodeClass\":\"fire.nodes.ml.
↪NodeStringIndexer\",\"x\":\"630.238px\",\"y\":\"272.879px\",\"fields\":[{\"name\":\
↪"handleInvalid\",\"value\":\"skip\",\"widget\":\"array\",\"title\":\"Handle Invalid\
↪",\"description\":\"Invalid entries to be skipped or thrown error\",\"optionsArray\
↪":[\"skip\",\"error\"],\"required\":false,\"display\":false,\"editable\":true,\
↪"disableRefresh\":false},{\"name\":\"inputCols\",\"value\":\"[\\\"DAY_OF_MONTH\\\",
```


(continues on next page)

(continued from previous page)


```
  "dateLastUpdated": 1566551544603,
  "userName": null,
  "userId": null,
  "userComment": null
}
],
```


### 29.1.4 Workflow Execution REST API

#### Overview

The Workflow Execution REST API's, allow you to execute Workflows, get results etc.

Below are the various Workflow Execution API's available in Fire Insights, They should be executed after you have logged into Fire Insights.

#### List all the Executions

List all the workflow executions.

An example request for List all the executions:

```
curl -X GET --header 'Accept: application/json' 'http://hostname:port/api/v1/workflow-
↪executions?page=0&size=1000' -b /tmp/cookies.txt
```

An example response:


```
 [
 {
 "id": 135,
 "analysisFlowId": 161,
 "userId": 33,
 "projectId": 33,
 "analysisFlowScheduleId": null,
 "status": 2,
 "name": "Test_csv",
 "category": "-",
 "description": "Fired Manually",
 "logs": "/tmp/fire/workflowlogs/workflow-5342148677548385044.log",
 "fireJobId": "02aedbe5-0713-4172-9f7c-c63272f7cbd9",
 "applicationId": "application_1560754639341_5932",
 "uiWebUrl": "http://hostname:4042",
 "metrics": null,
 "startTime": 1566821007783,
 "endTime": 1566821024075,
 "emailOnSuccess": null,
 "emailOnFailure": null
},
```


#### List Executions of a Workflow

Return the list of Executions for given workflowId.

workflowId = 131:

An example request for List executions of a Workflow:

```
curl -X GET --header 'Accept: application/json' 'http://hostname:port/api/v1/workflow-
↪executions/workflows/131' -b /tmp/cookies.txt
```

An example response:

```
 [
 {
 "id": 99,
 "analysisFlowId": 131,
 "userId": 33,
 "projectId": 33,
 "analysisFlowScheduleId": null,
 "status": 2,
 "name": "Test_workflow",
 "category": "-",
 "description": "Fired Manually",
 "logs": "/tmp/fire/workflowlogs/workflow-4439919411814145818.log",
 "fireJobId": "7b7b7dd5-b27b-419e-b853-794b5f53a5b8",
 "applicationId": "application_1560754639341_5929",
 "uiWebUrl": "http://hostname:4041",
 "metrics": null,
 "startTime": 1566795625424,
 "endTime": 1566795650970,
 "emailOnSuccess": null,
 "emailOnFailure": null
}
],
```

### GET Status of Workflow Execution

Return status of workflow execution for given workflowId.

workflowId = 193:

An example request for status of workflow execution

```
curl -X GET --header 'Accept: text/plain' 'http://hostname:port/api/v1/workflow-
↪executions/193/status'
```

An example response:

```
KILLED
```

### Stop the Execution of workflow

Stops the execution of workflow with specified workflowExecutionId.

Workflow Execution Id = 100:

An example request for Stopping specified workflow:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain'
↪'http://hostname:port/api/v1/workflow-execution/100/stop'' -b /tmp/cookies.txt
```

An example response:

```
{"status":"ok","message":"Stopping Analysis Flow Execution"}
```

### Kill the Execution of workflow

Kill the execution of workflow with specified workflowExecutionId.

Workflow Execution Id = 100:

An example request for Killing specified workflow:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain'
↪'http://hostname:port/api/v1/workflow-execution/100/kill' -b /tmp/cookies.txt
```

An example response:

```
Killed YARN application : yarn application -kill application_1560754639341_5930,Exit␣
↪Value : 0
```

### Delete Workflow Executions by days

Delete Workflow Executions by days

"days": "7":

An example request for deleting workflow executions by days:

```
curl -X DELETE --header 'Accept: text/plain' 'http://hostname:port/api/v1/workflow-
↪executions/days/7' -b /tmp/cookies.txt
```

An Example response:

```
Workflow executions deleted successfully
```

### Get Executed Task Count

Get Executed Task Count:

An example request for Getting Executed Task Count:

```
curl -X GET --header 'Accept: application/json' 'http://hostname:port/api/v1/workflow-
↪executions/tasks/count' -b /tmp/cookies.txt
```

An example response:

```
92
```

### Get Latest Executions

Get Latest Executions:

An Example request for Getting Latest Executions:

```
curl -X GET --header 'Accept: application/json' 'http://hostname:port/api/v1/workflow-
↪executions/latest' -b /tmp/cookies.txt
```

An example response:

```
[
{
"id": 162,
"analysisFlowId": 131,
"userId": 33,
"projectId": 33,
"analysisFlowScheduleId": null,
"status": 2,
"name": "Test_workflow",
"category": "-",
"description": "Fired Manually",
"logs": "/tmp/fire/workflowlogs/workflow-3535160145732140945.log",
"fireJobId": "7b456feb-22fe-474e-a0c6-f31c40a1a9cd",
"applicationId": "application_1560754639341_5934",
"uiWebUrl": "http://hostname:4040",
"metrics": null,
"startTime": 1566834233892,
"endTime": 1566834262432,
"emailOnSuccess": null,
"emailOnFailure": null
},
```

### 29.1.5 Dashboard REST API

#### Overview

The Dashboards REST API's, allow you to interact with the Dashboards.

Below are the various Dashboard API's available in Sparkflows

They should be executed after you have logged into Sparkflows

#### Get List of Dashboards for the user

Returns the list of dashboards for the logged in user.

- Header: sortPara:asc/desc.

```
curl -i --header "Accept:application/json" -H "Content-Type:application/json" -H
↪"sortPara:desc" -X GET -b /tmp/cookies.txt localhost:8080/dashboardsJSON
```

#### Create New Dashboard / Save Dashboard

Set dashboardId value null to create new dashboard:

```
curl - X POST --header 'Content-Type: application/json' --header 'Accept: text/plain'␣
↪--header 'dashboardId: null' -d '{"category": "string", "description": "string",
↪"name": "string","sheets": [{"description": "string","idx": "string","items": [ {
↪"description": "string","id": 0,"name": "string","nodeId": "string","type": "string
↪", "workflowId": "string","workflowName": "string","x": "string","y": "string"}],
↪"name":"string","type": "string"}],"uuid": "string"}' 'http://localhost:8080/
↪saveDashboard' -b /tmp/cookies.txt
```

#### Get Dashboard by Id

- id:1(Url Parameter)

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/
↪dashboards?sortPara=dsc&projectId=1' -b /tmp/cookies.txt
```

#### Get dashboard results

- dashboardId:1
- sheetId:0

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/
↪dashboards/results?dashboardId=1&sheetId=0' -b /tmp/cookies.txt
```

#### update dashboard

- dashboardContent: abcd,
- dashboardId: 1,

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: */*' -d 'abcd
↪' 'http://localhost:8080/api/v1/dashboards/1'
```

#### Delete Dashboard

- dashboardId: 1,
- projectId: 1,

```
curl -X DELETE --header 'Accept: text/plain' 'http://localhost:8080/api/v1/dashboards/
↪1?projectId=1' -b /tmp/cookies.txt
```

### 29.1.6 HDFS REST API

#### Overview

The HDFS REST API's, allow you to interact with the HDFS of the Hadoop Cluster Sparkflows is connected to.

Below are the various HDFS API's available in Sparkflows

They should be executed after you have logged into Sparkflows

#### Get List of Files in Directory

Returns list of all the files on hdfs in the users home directory

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/hdfs'
```

### Create HDFS directory

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: text/plain'
↪'http://localhost:8080/api/v1/hdfs/dir/create'
```

### Get list of files in HDFS in the specified directory

Returns list of files in HDFS in the specified directory(/user/sparkflows/)

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/hdfs/
↪dir/open?path=%2Fuser%2Fsparkflows%2F'
```

### Get list of all the files on hdfs in the users home directory in sorted order

*sortPara: alphabetical

*path: /user/sparkflows/

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/hdfs/
↪files?sortPara=alphbetical&path=%2Fuser%2Fsparkflows%2F'
```

### Upload file

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
↪json' 'http://localhost:8080/api/v1/hdfs/files/upload' -b /tmp/cookies.txt
```

### Deletes a file from HDFS

*path: /user/sparkflows/Airline.csv

```
curl -X DELETE --header 'Accept: text/plain' 'http://localhost:8080/api/v1/hdfs/files/
↪delete?path=%2Fuser%2Fsparkflows%2FAirline.csv'
```

### Download HDFS file

*path: /user/sparkflows/Airline.csv

```
curl -X GET --header 'Accept: application/json' 'localhost:8080/api/v1/hdfs/files/
↪download?path=%2Fuser%2Fsparkflows%2FAirline.csv'
```

### Rename HDFS File

*sourceFilePath: /user/sparkflows/Airline.csv

*destinationFilePath: /user/sparkflows/airline.csv

```
curl -X GET --header 'Accept: text/plain' 'http://localhost:8080/api/v1/hdfs/files/
↪rename?sourceFilePath=%2Fuser%2Fsparkflows%2FAirline.csv&destinationFilePath=%2Fuser
↪%2Fsparkflows%2Fairline.csv'
```

#### Get first X bytes of content of a file

*path: /user/sparkflows/Airline.csv

```
curl -X GET --header 'Accept: text/plain' 'http://localhost:8080/api/v1/hdfs/files/
↪open?path=%2Fuser%2Fsparkflows%2FAirline.csv'
```

### 29.1.7 HIVE REST API

#### Overview

The HIVE REST API's, allow you to interact with the HIVE of the Hadoop Cluster Sparkflows is connected to.

Below are the various HIVE REST API's available in Sparkflows

They should be executed after you have logged into Sparkflows

#### Get all Hive Databases

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/hive/
↪databases' -b /tmp/cookies.txt
```

#### Get Table for a given Database

- "db": "default",
- "table": "sample_07"

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/hive/
↪tables?db=default&table=sample_07' -b /tmp/cookies.txt
```

#### Get all Hive Databases

```
* curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/hive/
↪databases' -b /tmp/cookies.txt
```

### 29.1.8 Scheduler REST API

#### Overview

The Scheduler REST API's, allow you to schedule some jobs once Sparkflows connected to Hadoop Cluster.

Below are the various Scheduler REST API's available in Sparkflows

They should be executed after you have logged into Sparkflows

### Get list of all Workflows Scheduled

- analysisflowId = 1

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/api/v1/
↪workflow-schedules/projects/1/workflows/1'  -b /tmp/cookies.txt
```

### Schedule new Workflow

curl:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: */*' -d '1'
↪'http://localhost:8080/api/v1/workflow-schedules'
```

JSON:

```
 "analysisFlowId": 0,
 "cronPattern": "string",
 "dateCreated": "2019-08-06T11:77:17.221Z",
 "dateLastUpdated": "2019-08-06T11:77:17.221Z",
 "day": "string",
 "dayOfTheMonth": 0,
 "description": "string",
 "emailonFailure":"string",
 "emailonSuccess": "string"
 "endTime": "2019-08-06T11:77:17.221Z",
 "fireEvery": "string",
 "firedTime": "2019-08-06T11:77:17.221Z",
 "hour": 0,
 "id": 0,
 "Libjars": "string",
 "minute": 0,
"name": "string",
"sparkSubmitOptions": "string",
"startTime": "22019-08-06T11:77:17.221Z",
"userId": "0",
"id": "string",
}'    'http://137.117.83.79:8080/api/v1/workflow-schedules'  -b /tmp/cookies.txt
```

### Delete Scheduled Workflow

It deletes a scheduled instance of a workflow:

```
curl  -X GET    --header 'Accept: application/json'    --header 'id: 1'    'http://
↪localhost:8080/api/v1/workflow-schedules/1' -b /tmp/cookies.txt
```

CHAPTER 30

# Third Party Acknowledgements

## 30.1 Third Party Acknowledgements

Sparkflows uses and distributes the following third party software. These are open source software licensed as mentioned.

### 30.1.1 Server Libraries

- Apache Spark
  - https://spark.apache.org/
  - Copyright © 2018 The Apache Software Foundation
  - License: Apache-2.0
- Apache Avro
  - https://avro.apache.org/
  - Copyright © 2012 The Apache Software Foundation
  - License: Apache-2.0
- Apache Commons
  - https://commons.apache.org/
  - Copyright © 2019 The Apache Software Foundation.
  - License: Apache-2.0
- Apache Hadoop
  - https://hadoop.apache.org/
  - Copyright © 2018 The Apache Software Foundation.
  - License: Apache-2.0

- Apache HBase
  - https://hbase.apache.org/
  - Copyright ©2007–2019 The Apache Software Foundation
  - License: Apache-2.0
- Apache Hive
  - https://hive.apache.org/
  - Copyright © 2011-2014 The Apache Software Foundation
  - License: Apache-2.0
- Apache HTTP
  - https://hc.apache.org/
  - Copyright © 1999–2019 The Apache Software Foundation.
  - License: Apache-2.0
- spark-streaming-kafka
  - http://spark.apache.org/
  - © 2017 Apache Software Foundation
  - License: Apache-2.0
- Apache pdfbox
  - https://pdfbox.apache.org
  - Copyright © 2009–2019 The Apache Software Foundation
  - License: Apache-2.0
- Apache OpenNLP
  - https://opennlp.apache.org/
  - Copyright © 2017 The Apache Software Foundation
  - License: Apache-2.0
- Apache Tika
  - https://tika.apache.org/
  - Copyright © 2019 The Apache Software Foundation
  - License: Apache-2.0
- Apache Tomcat
  - http://tomcat.apache.org/
  - Copyright © 1999-2019, The Apache Software Foundation
  - License: Apache-2.0
- AWS Java SDK
  - https://aws.amazon.com/
  - Copyright © 2019, Amazon Web Services, Inc. or its affiliates
  - License: Apache-2.0

- Eclipse jetty
  - https://www.eclipse.org/jetty/
  - Copyright © 2016 The Eclipse Foundation.
  - License: EPL- v 2.0
- Elasticsearch-spark-20_2.11
  - https://github.com/elastic/elasticsearch-hadoop
  - © 2019. Elasticsearch B.V.
  - License: Apache-2.0
- Guava
  - https://github.com/google/guava
  - https://github.com/google/guava/blob/master/COPYING
  - License: Apache-2.0
- H2O
  - https://www.h2o.ai/
  - © Copyright 2013, 0xdata, Inc.
  - License: Apache-2.0
- Json Java
  - http://www.json.org
  - Copyright (c) 2002 JSON.org
  - License: BSD-style with "no evil" clause
- Log4J
  - http://logging.apache.org/log4j/2.x/
  - Author: The Apache Software Foundation
  - License: Apache-2.0
- Sagemaker-spark_2.11
  - https://github.com/aws/sagemaker-spark
  - Author: The Apache Software Foundation
  - License: Apache-2.0
- Mongo_spark_connector_2.11
  - http://github.com/mongo-spark
  - Author: The Apache Software Foundation
  - License: Apache-2.0
- Python
  - https://www.python.org/
  - Copyright ©2001-2019. Python Software Foundation
  - License: PSFL2

- Quartz
  - http://www.quartz-scheduler.org/
  - Copyright© Terracotta, Inc., a wholly-owned subsidiary of Software AG USA, Inc. All rights reserved
  - License: Apache-2.0
- Spring Framework
  - https://spring.io/
  - Copyright © 2019 Pivotal Software, Inc. All Rights Reserved
  - License: Apache-2.0
- SLF4J
  - http://www.slf4j.org/
  - Copyright (c) 2004-2017 QOS.ch
  - License: MIT

### 30.1.2 Frontend Libraries

- angularjs
  - https://angularjs.org/
  - Copyright (c) 2010-2014 Google, Inc.
  - License: MIT
- bootstrap
  - http://getbootstrap.com/2.3.2/
  - Copyright 2011-2014 Twitter, Inc
  - License: MIT
- jquery
  - https://jquery.com/
  - Copyright 2019 The jQuery Foundation. jQuery License
  - License: MIT
- rxjs
  - https://rxjs-dev.firebaseapp.com/
  - Copyright 2015-2018 Google, Inc., Netflix, Microsoft Corp.
  - License: Apache License 2.0

### 30.1.3 Definitions

- Apache-2.0 : Apache License, Version 2.0 : http://www.apache.org/licenses/LICENSE-2.0.html
- MIT : MIT License : https://en.wikipedia.org/wiki/MIT_License#Relation_to_Patents
- BSD-style: BSD-style License : http://json.org/license.html
- EPL: EPL - v 2.0 License: https://www.eclipse.org/legal/epl-2.0/

- PSFL2 : Python Software Foundation License Version 2

CHAPTER 31

# Indices and tables

- genindex
- modindex
- search