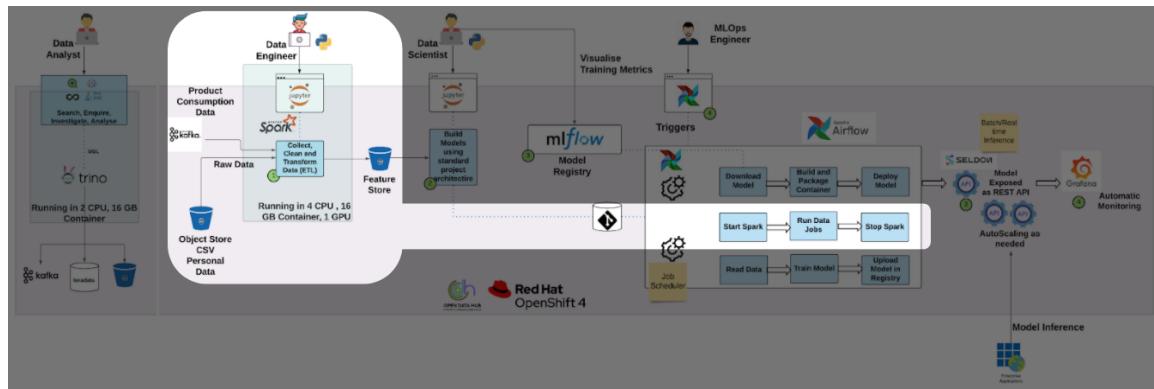


Lab 1 - Data Engineering

1.1 Introduction

The Open Data Hub exposes a data-focused tool - for Extract Transform Load (ETL) of data originating in multiple data sources, i.e. Apache Spark. Spark allows fine-grained ETL control, e.g. using Regex to match data patterns. Spark provides a further toolset to allow data professionals to prepare quality data for consumption by data scientists and AI models.

This diagram illustrates the workflow we're implementing - the beginning part of the overall AI/ML workflow:



You can see, we source raw data from Kafka and S3 object storage. We use Jupyter notebooks to do some simple data engineering - combining these datasets on customerId using Spark. We then push that prepared data (a CSV file) to another bucket in our S3 object store, called Minio.

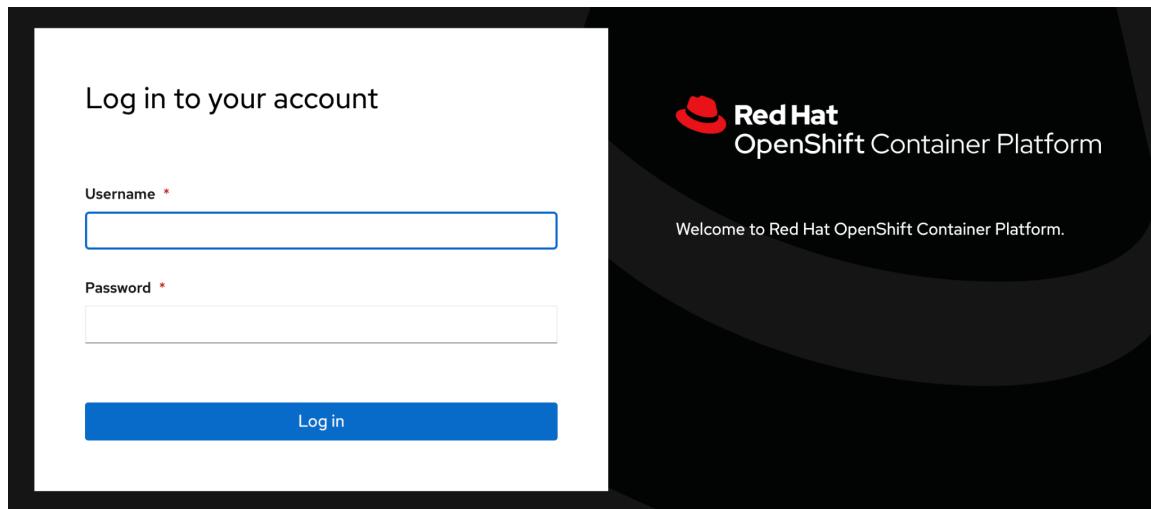
We also introduce a new level of autonomy and control - automation and scheduling of this task - using the workflow scheduling and management tool - Apache Airflow. You'll see this is very simple - democratising activities that used to be the realm of DevOPs specialists.



1.2 Access the OpenShift environment

Your instructor will provide you with a username, password, and link to access your environment. Using your browser and the URL provided by the instructor:

1. Open a new web page
2. Enter your username and password
3. Click **Log in**.



The main OpenShift console will be displayed.

A screenshot of the Red Hat OpenShift Container Platform main console. The top navigation bar includes the Red Hat logo, a search bar, and user information. The left sidebar has a "Developer" section with options like "+Add", "Topology" (which is selected and highlighted in grey), "Monitoring", "Search", "Builds", "Helm", "Project" (which is also highlighted), and "ConfigMaps". The main content area shows a "Topology" section with a message "Select a Project to view the topology or [create a Project](#)". Below this is a table listing projects. The table has columns for Name, Display name, Status, Requester, and Created. Two projects are listed: "ml-workshop" and "user1-project", both created by "opentlc-mgr" on "11 Oct 2021, 09:14".



Open the project you will use in the workshop. In the menu bar:

4. Click **All Projects**.
 5. Type **ml-workshop** in the project text box.
 6. Click **ml-workshop** from the drop-down list.

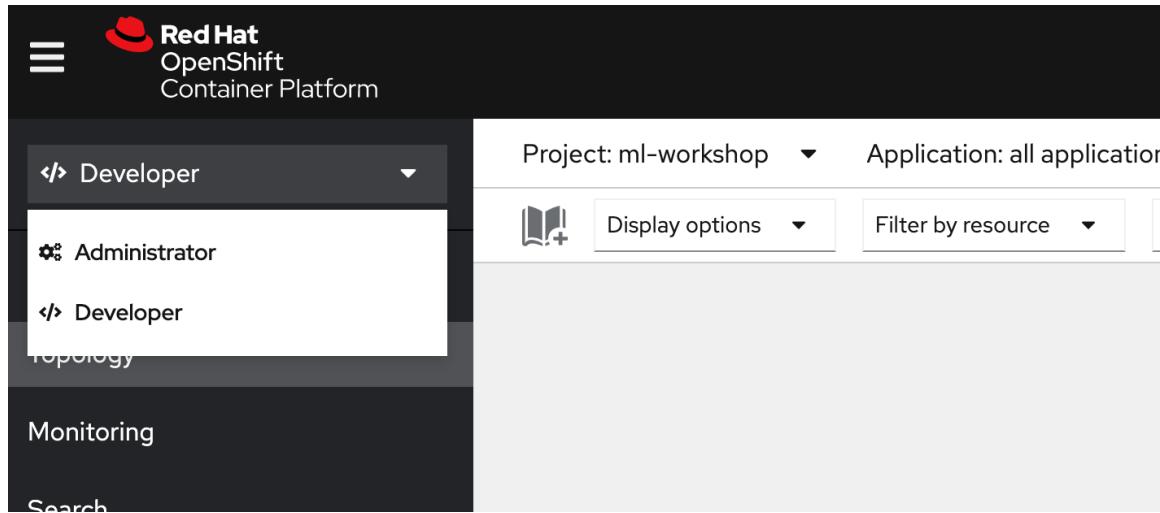
The screenshot shows the Red Hat OpenShift Container Platform interface. The top navigation bar includes the Red Hat logo and the text "Red Hat OpenShift Container Platform". On the left, a sidebar menu lists "Developer", "+Add", "Topology", "Monitoring", and "Search". The main content area is titled "Project: All Projects" and displays a search bar with the letter "m" typed in. Below the search bar are two items: "Create Project" and "ml-workshop". At the bottom of the main content area, there is a search bar with the placeholder "Search by name..." and a filter dropdown set to "Name".

OpenShift will open the project and display the project Topology.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The top navigation bar includes the Red Hat logo, 'Red Hat OpenShift Container Platform', and user information ('user1'). The left sidebar has a 'Developer' dropdown, a '+Add' button, and links for 'Topology', 'Monitoring', 'Search', 'Builds', 'Helm', 'Project', 'ConfigMaps', and 'Secrets'. The main content area displays a 'Topology' view for the 'ml-workshop' project. At the top of the main area are 'Display options' (with a book icon), 'Filter by resource' (with a magnifying glass icon), and a search bar ('Find by name...'). Below these are several clusters of nodes, each represented by a dashed box containing multiple circular icons. Some nodes have labels like 'app-1', 'app-2', 'app-3', etc., and some have small icons above them. A legend in the bottom right corner identifies the node types: a red circle with a white target symbol for 'Deployment', a blue circle with a white gear symbol for 'Service', a green circle with a white cloud symbol for 'ConfigMap', and a grey circle with a white wrench symbol for 'Secret'.



Note: Throughout these labs, you will be asked to select either the **Developer** or **Administrator** perspectives to perform the lab steps. To change the perspective, click the Perspective dropdown menu in the top left of the console, then click the perspective you need to use.



You have now completed accessing the lab environment. Proceed to the next section.



1.3 Instructions for the Spark workshop

Change to the **Administrator** perspective:

- Click the **Perspective** drop-down list in the top left of the console.
- Click **Administrator**.

A screenshot of the OpenShift Container Platform console. The top navigation bar shows the Red Hat logo and the text "OpenShift Container Platform". Below this, a dropdown menu labeled "Developer" is open, with "Administrator" selected. To the right, the main interface displays a "Project: ml-workshop" view. It includes a "Display options" dropdown and a "Filter by resource" search bar. The central area shows a grid of application icons, including "seldon_aniger", "seldon_v17.0", "jupyter_user29", "kafka", "app_af_low-web", "modeld_aresal", and "modeld_gresol". On the left side, there is a vertical sidebar with links for "Topology", "Monitoring", and "Search".

The OpenShift console displays the **Administrator** perspective.

A screenshot of the OpenShift Container Platform console. The top navigation bar shows the Red Hat logo and the text "OpenShift Container Platform". Below this, a dropdown menu labeled "Administrator" is selected. To the right, the main interface displays a "Projects" view. It includes a search bar with "Name" and "Search by name...". A table lists projects: "ml-workshop" (PR) with status "Act", and "user1-project" (PR) with status "Act". The left sidebar has links for "Home", "Projects", "Search", "Explore", and "Events", with "Projects" currently selected.



Open a second browser tab with the same URL for the OpenShift console. We will refer to these as **tab 1** and **tab 2** in the instructions below.

In Tab 1

- Click **Workloads > Pods**.

The screenshot shows the OpenShift Container Platform interface. On the left, there is a navigation sidebar with the following items:

- Administrator
- Home
- Projects
- Search
- Explore
- Events
- Operators
- Workloads
 - Pods
 - Deployments
 - DeploymentConfigs

The "Workloads" section is currently selected. The main content area is titled "Project: ml-workshop". It displays a table of "Pods" with the following columns: Name, Status, Ready, Restarts, and Owner. The table contains four rows of data:

Name	Status	Ready	Restarts	Owner
app-aflow-airflow-scheduler-74bff4d9c6-lrrtz	Running	2/2	3	RS app-aflow-airflow-scheduler-74bff4d9c6
app-aflow-airflow-web-56cbc96d58-rxd9s	Running	2/2	4	RS app-aflow-airflow-web-56cbc96d58
app-aflow-airflow-worker-0	Running	2/2	4	SS app-aflow-airflow-worker
app-aflow	Running	1/1	0	app-aflow

A blue "Create Pod" button is located in the top right corner of the main content area.

OpenShift displays a list of all pods running in the ml-workshop project.

- Enter your username in the Filter **Search by Name** text box

The screenshot shows the OpenShift Container Platform interface. The navigation sidebar is identical to the previous one, with the "Workloads" section selected. The main content area is titled "Project: ml-workshop". The "Name" filter input field is highlighted with a red box and contains the text "user30". Below the filter, there is a "Clear all filters" link. At the bottom of the page, it says "No Pods found".

OpenShift will remove all (or almost all) of the pods so you only see the pods related to you.



Red Hat

We will now open JupyterHub using the second tab. Open **Tab 2** in your browser

- Make sure you have the Administrator perspective selected.
- Click **Networking > Routes**.

OpenShift displays the external network routes to the applications in the ml-workshop projects. These routes are how you open the console for each application in Open Data Hub.

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar is titled "Administrator" and includes links for Home, Operators, Workloads, Networking (with Services, Routes, Ingresses, and NetworkPolicies), and a "Create Route" button. The main content area is titled "Routes" and shows a table for the "ml-workshop" project. The table has columns for Name, Status, Location, and Service. It lists three routes: "ap-airflow2" (Status: Accepted, Location: https://ap-airflow2-ml-workshop.apps.cluster-lwwqr.lwwqr.sandbox7), "customerchurnuser291020212330547 57270" (Status: Accepted, Location: https://customerchurn-054757270-ml-workshop.lwwqr.lwwqr.sandbox7), and "jenkins-ml-jenkins" (Status: Accepted, Location: https://jenkins-ml-jenkins-ml-workshop.apps.cluster-lwwnr.lwwnr.sandbox7).

Name	Status	Location	Service
ap-airflow2	Accepted	https://ap-airflow2-ml-workshop.apps.cluster-lwwqr.lwwqr.sandbox7	
customerchurnuser291020212330547 57270	Accepted	https://customerchurn-054757270-ml-workshop.lwwqr.lwwqr.sandbox7	
jenkins-ml-jenkins	Accepted	https://jenkins-ml-jenkins-ml-workshop.apps.cluster-lwwnr.lwwnr.sandbox7	

Filter the URL to find Jupyterhub.

- Type *Jupyterhub* in the **Search by name** text box.
- Click the route in the **Location** column

The screenshot shows the Red Hat OpenShift Container Platform interface with a search filter applied. The search bar contains "jupyterhub". The table lists one route named "jupyterhub" with the status "Accepted". The "Location" column shows the URL "https://jupyterhub-ml-workshop.apps.cluster-fcla.fcla.sandbox640.opentlc.com".

Name	Status	Location	Service
jupyterhub	Accepted	https://jupyterhub-ml-workshop.apps.cluster-fcla.fcla.sandbox640.opentlc.com	jupyterhub



OpenShift will launch a new browser tab and prompt you for a user name and password. Those are the same username and password you used to log in to OpenShift. We now want to authorise the Service account jupyterhub-hub to access your account. Click **Allow selected permissions**.

Authorize Access

Service account jupyterhub-hub in project ml-workshop is requesting permission to access your account (user123)

Requested permissions

- ### user:info

Read-only access to your user information (including username, identities, and group membership)

You will be redirected to https://iupvtrehub-ml-workshop.apps.cluster-dhls2.dhls2.sandbox1573.opentlc.com/hub/oauth_callback

Allow selected permissions

OpenShift will prompt you to select the type and size of notebook server to start.

- Click **Elyra Notebook Image with Spark**
 - Expand the **Container size** dropdown listbox and click **Large**.

 jupyterhub Home Token Services▼

Start a notebook server

Select options for your notebook server.

Notebook image

SciKit v1.10 - Elyra Notebook Image

Elyra Notebook Image with Spark

Deployment size

Container size

Environment variables

 Add more variables

Start server

- Click **Start Server**.

Warning: Please select the correct notebook image, otherwise the lab will not work.



OpenShift starts a new Jupyterhub notebook server for your lab environment.

A screenshot of a web browser showing the JupyterHub interface. The title bar says "jupyterhub". The main content area displays a message: "Your server is starting up. You will be redirected automatically when it's ready for you." Below this is a progress bar. At the bottom, there is a log entry: "2021-10-11T06:38:25Z [Normal] Pulling image "quay.io/odh-jupyterhub/s2i-spark-minimal-notebook@sha256:3d68e863b2575442e99239bcd963e53729e39c2ac10d6fef6ece8174a8de5513"" followed by a link "Event log".

After a few minutes, JupyterHub displays the notebook.

A screenshot of the JupyterHub file manager. The title bar says "jupyterhub". The top navigation bar includes "Logout" and "Control Panel". Below the title bar, there are tabs for "Files", "Running", and "Clusters", with "Running" being the active tab. A sub-header "Select items to perform actions on them." is present. On the left, there is a file list with a single item: "lost+found" (with a folder icon). On the right, there are filters for "Name" (sorted by name), "Last Modified" (sorted by last modified), and "File size".

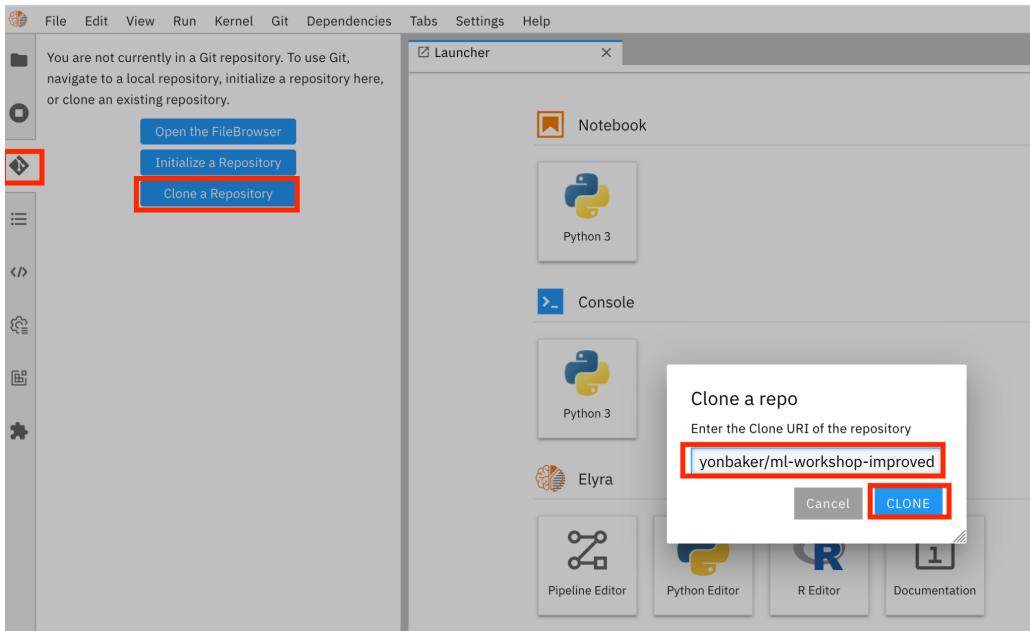
Copy notebooks

The first thing we want to do is pull down our notebooks from our repository
<https://github.com/bryonbaker/ml-workshop-improved>

- Do the following in this order - as shown below
 - a. Click the **Git icon** on the left and side of the screen
 - b. Click the **Clone a Repository** button
 - c. Enter <https://github.com/bryonbaker/ml-workshop-improved> in the textbox
 - d. Click **CLONE**

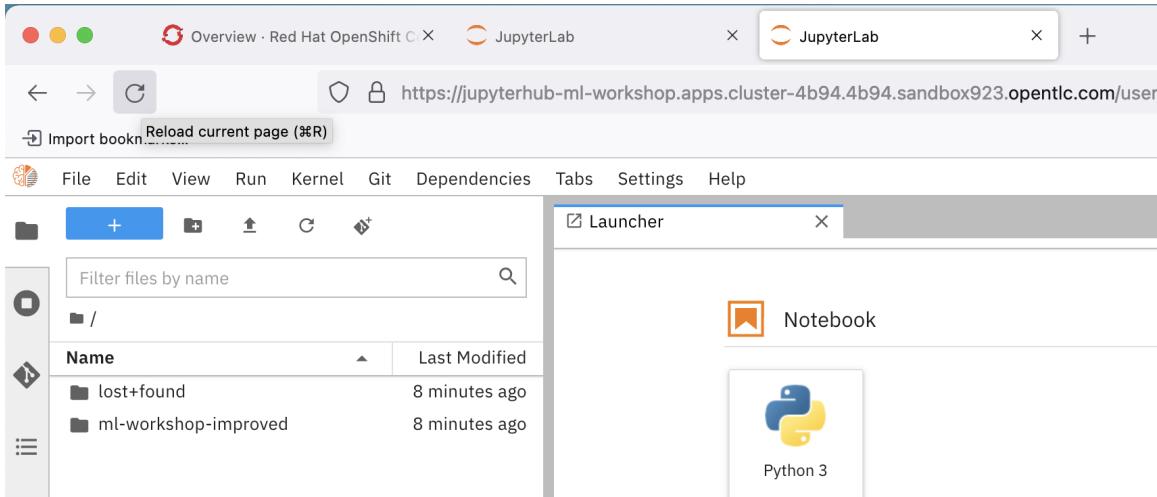


Red Hat



Jupyterhub clones the git repository that has all the code you will be using for this lab.

- Click the **Refresh or Reload** tab in your browser to display a Jupyterhub notebook.



Next, we must open the folder containing the notebooks for this workshop.

- Double click **ml-workshop-improved/notebook/**

OpenShift displays all of the notebooks you will be using in this lab.



Red Hat

The screenshot shows the Jupyter Notebook interface. On the left is a file browser with a search bar. The path is /ml-workshop-improved/notebook/. The list of files includes:

Name	Last Modified
hyper_parameters.py	11 minutes ago
Merge_Data.ipynb	11 minutes ago
Model_Experiments.ipynb	11 minutes ago
spark_util.py	11 minutes ago
Train_Model.ipynb	11 minutes ago
Visulaise_Model.ipynb	11 minutes ago

On the right is a 'Launcher' panel with sections for 'Notebook' and 'Python 3'. The 'Notebook' section has a 'New Notebook' button.

Before we get going, you need to make some small changes to the code.

- Open **hyper_parameters.py**
- Replace **user29** with your username that you are using for this lab (the same one you used to log on to OpenShift).

```
user_id = "<your username>"
```

Save your work

- Click the **Save** icon as shown to save your work.

The screenshot shows the Jupyter Notebook interface. The 'hyper_parameters.py' file is selected in the file browser on the left. On the right, the code editor shows the contents of the file:

```
def get_hyper_paras():
    # #####
    #
    # PLACE YOUR STUDENT CONFIGURATION INFORMATION IN
    #
    user_id = "user29"
    miniofilename = part-00000-aa888b5b-fd89-431e-831
    #
    #
    # DO NOT CHANGE ANY CODE BELOW THIS LINE
    #
    # #####
```

The 'Save' icon in the toolbar is highlighted with a red box.



- Click **Merge_Data.ipynb**.

Jupyterhub opens the **Merge_Data** notebook. This notebook is used by the Data Engineer to prepare the data.

Run the notebook

You are now good to go!

- Scroll to the top of the notebook.
- Click inside the first code cell.

A screenshot of a JupyterHub Notebook interface. The title bar says "jupyterhub Merge_Data Last Checkpoint: a minute ago (autosaved)". The status bar shows "Not Trusted" and "Python 3". The main area is titled "JupyterHub Notebook" and contains a note about being hosted on OpenShift. Below that, it says "First, install and import required libraries, watermark our file, initialise our Spark Session Builder and initialise our environment with required configuration". A code cell labeled "In []:" contains the command "%pip install watermark %pip install Minio", which is highlighted with a red box.

We will now step through each cell one by one so you can see what is going on.

- To execute a cell, type the **SHIFT + RETURN** keys together. Walk through the entire file this way, executing as you go. (if there are any blank cells, skip through them using SHIFT + RETURN).

Note: If you want to run the entire notebook click: **Kernel** from the top menu > **Restart & Run All**.



At a high level, this is what is happening in the most important cells:

```
1 %pip install watermark
%pip install Minio
%pip install matplotlib

2 import os
import json
from pyspark import SparkConf
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql.functions import from_json, col, to_json, struct
import watermark
from minio import Minio

%matplotlib inline
%load_ext watermark

3 %watermark -n -v -m -g -iv
```

Load Hyper parameters

```
4 from hyper_parameters import get_hyper_paras
user_id,PROJECT_NAME,EXPERIMENT_NAME,experiment_name, s3BucketFullPath = get_hyper_paras()

5 sparkSessionBuilder = SparkSession\
    .builder\
    .appName("Customer Churn ingest Pipeline")

6 submit_args = "--conf spark.jars.ivy=/tmp \
--conf spark.hadoop.fs.s3a.endpoint=http://minio-ml-workshop:9000 \
--conf spark.hadoop.fs.s3a.access.key=minio \
--conf spark.hadoop.fs.s3a.secret.key=minio123 \
--conf spark.hadoop.fs.s3a.path.style.access=true \
--conf spark.hadoop.fs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem \
--packages org.apache.hadoop:aws:3.2.0"
```

Connect to Spark Cluster provided by OpenShift Platform

```
7 import spark_util

spark = spark_util.getOrCreateSparkSession("ML Ops Demo", submit_args)
spark.sparkContext.setLogLevel("INFO")
print('Spark context started.')
```

1. *pip install xxxx*, installs various libraries that aren't contained in our case container image
2. import the python libraries we need



3. *watermark* outputs the versions of various components, libraries, operating system attributes etc.
4. Here we load the user specific parameters - based on the username you entered in `hyper_parameters.py`
5. Here we create a Spark session, a precursor to firing up our own Spark server.
6. Here we set up various environment variables, including connection access to our S3 object store, in our case implemented using the open-source component Minio.
7. Here we actually start our Spark server. This cell can take several minutes to start.

Declare our input data sources, import and combine them

```
8  dataFrame_Customer = spark.read\  
     .options(delimiter=',', inferSchema='True', header='True') \  
     .csv("s3a://rawdata/customers/Customer-Churn_P1.csv")  
dataFrame_Customer.printSchema()  
  
9  dataFrame_Products = spark.read\  
     .options(delimiter=',', inferSchema='True', header='True') \  
     .csv("s3a://rawdata/products/Customer-Churn_P2.csv")  
dataFrame_Products.printSchema()  
  
10 # from pyspark.sql.types import *  
# from pyspark.sql.functions import *  
  
# srcKafkaBrokers = "odh-message-bus-kafka-bootstrap:9092"
```

8. Here we pull in our data from S3 - our CSV based demographic data for each of our approximately 7000 customers.
9. Note - we are upgrading our demo and workshop. Kafka will be available shortly for the second half(side) of the data set. For now, both halves of our dataset comes from CSV files in S3 object storage
10. Commented out Kafka connection code - ignore for now



```
11 dataFrom_All = dataFrame_Customer.join(dataFrame_Products, "customerID", how="full")
```

Push prepared data to object storage and stop Spark cluster to save resources

```
12 file_location = "s3a://data/full_data_csv-" + user_id  
dataFrom_All.repartition(1).write.mode("overwrite")\  
    .option("header", "true")\  
    .format("csv").save(file_location)
```

```
13 spark.stop()
```

```
14 print('Notebook complete!')
```

11. We join these 2 datasets, on the common column to each: *customerID*.
12. We push our data to our object store - filename contains our username.
13. We are all done now - we stop our Spark server.
14. Signal that we're done - and the notebook has run successfully

This is our data engineering notebook finished. It's a simple exercise, though the same toolset could be used for much more complex data engineering tasks. Later on, when we automate this Merge-Data task in a pipeline, you'll see the output file in Minio S3 object storage.

Allow the Data Engineer to automate and schedule their work

Next, we introduce a new level of autonomy and control to the data engineer - automated running and optionally scheduling of the Merge Data notebook. As a data engineer, you've built a data pipeline (merge-data) that merges customer data from 2 data sources. But:

- How do I run it with production workloads?
- How do I schedule it to run - overnight, every day or every week etc.?
- How do I ensure this pipeline works right across the ecosystem?

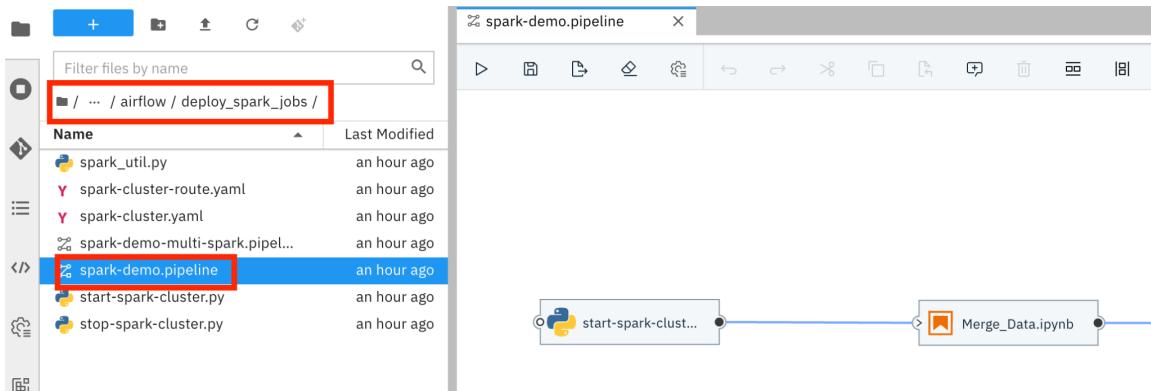
That's where a tool called Airflow comes in. With Airflow,

- you can automate the running of your notebook - and run it anywhere, or scheduled any time
- the same IDE (Jupyter Hub in OpenShift) can be used to develop and create the automated data pipelines



Let's get started

- Navigate to **ml-workshop-improved/airflow/deploy_spark_jobs/**
- Open **spark-demo-multi-spark.pipeline**



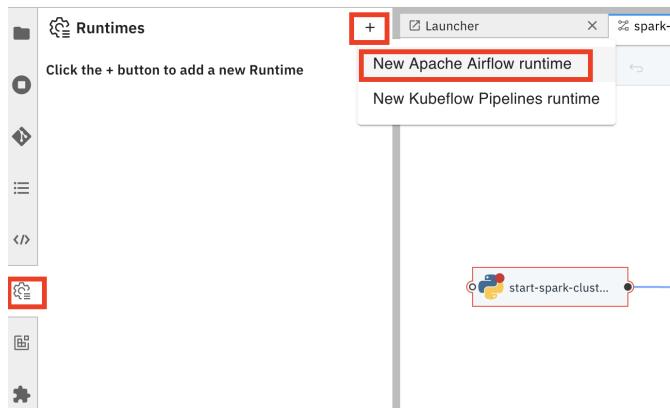
Notice it's a canvas-type GUI component onto which Python files can be dragged. Each component in the pipeline can be configured from this GUI - which we'll do later.

- The next thing we need to do is set up an Apache Airflow runtime. Before we do that we need to get the value of your **Airflow Route URL**.
- On OpenShift, navigate to **Networking > Routes** and filter by the name **ap-airflow2** as shown. Copy the URL under Location. This is your **Airflow Route URL**.

The screenshot shows the Red Hat OpenShift Container Platform UI. On the left, there is a sidebar with a navigation menu. The **Networking** section is expanded, and the **Routes** option is selected, indicated by a red box around the **Routes** link. In the main content area, the title is **Routes**. There is a filter bar with a **Name** input field containing `ap-airflow2`, which is also highlighted with a red box. Below the filter, a table lists a single route entry: `ap-airflow2` with status `Accepted` and location `https://ap-airflow2-ml-workshop.apps.cluster-4b94.4b94.sandbox923.opentlc.com`, which is also highlighted with a red box.

Back in Jupyter Hub, do the following:

- Click on the **Runtimes** icon on the left-hand side
- Click on the **+** icon on the top
- Select **New Apache Airflow runtime**



- Fill it in as follows - and click **Save and Close**

Name:	<code>userXX</code> Airflow runtime [footnote¹]
Apache Airflow UI endpoint:	your < Airflow Route URL > retrieved above
Apache Airflow User Namespace:	ml-workshop
Github API endpoint:	https://api.github.com [footnote²]
Github DAG Repo:	airflow-dags/dags [footnote³]
Branch:	main
Github Personal Access Token:	<code>ghp_OHsOZykc6nv0Ll1S4KiEdLJcsebTIn0mB9oH</code>
Cloud Object Storage Endpoint	http://minio-ml-workshop:9000 [footnote⁴]
The Cloud Object Storage Username	minio
The Cloud Object Storage Password	minio123
Cloud Object Storage bucket name	airflow

¹ Replacing `userXX` with your username

² Note: when you create the pipeline, it doesn't get saved to Airflow, rather it is saved to Git - to give you versioning etc.
Then Airflow downloads from Git. Means you need to use a Git repo.

³ For convenience, use our temporary repo setup for workshops:
<https://github.com/airflow-dags/dags>

⁴ Airflow needs S3 to push logs to

As shown here:

Edit "user30 Airflow runtime"

All fields marked with an asterisk are required. [Learn more ...]

Name *	user30 Airflow runtime	Description	
Tags	Add Tag +		
Apache Airflow			
Apache Airflow UI Endpoint *	https://ap-airflow2-ml-workshop.apps.cluster-4b94.4b94.sandbox923.opentlc.com/		
Github API Endpoint *	https://api.github.com		
Github DAG Repository Branch *	main	Github DAG Repository *	airflow-dags/dags
Github Personal Access Token *		
Cloud Object Storage			
Cloud Object Storage Endpoint *	http://minio-ml-workshop:9000		
Cloud Object Storage Username *	minio		
Cloud Object Storage Bucket Name *	airflow		
Cloud Object Storage Credentials Secret			
Cloud Object Storage Password *		
SAVE & CLOSE			

Next, we need to add a Python Runner.

- Choose **Runtime Images > +** and fill it in as follows - and click **Save and Close**

Name: Airflow Python Runner
 Image Name: quay.io/ml-aml-workshop/airflow-python-runner:0.0.8
 Image Pull Policy: IfNotPresent

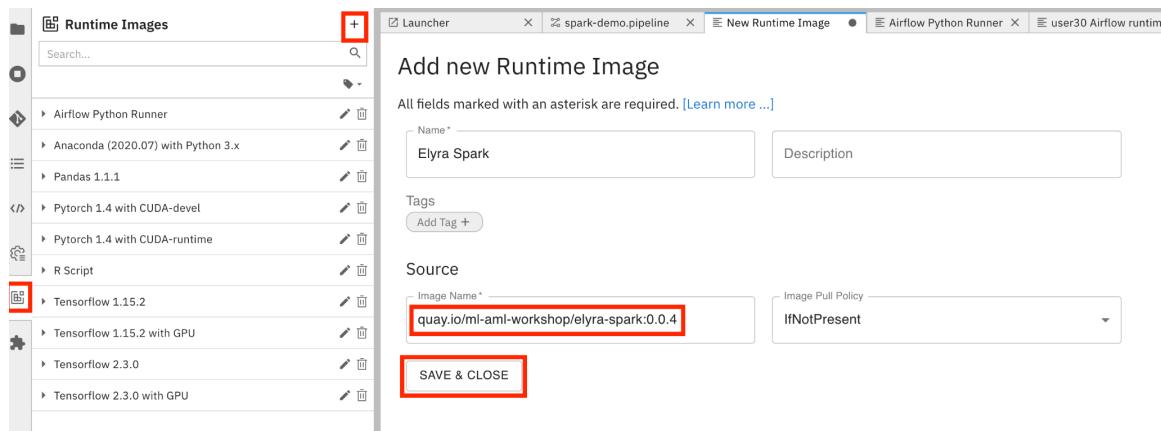
as shown here:

Runtime Images	+																		
<table border="1"> <tr> <td>Search...</td> </tr> <tr> <td>Anaconda (2020.07) with Python 3.x</td> </tr> <tr> <td>Pandas 1.1.1</td> </tr> <tr> <td>Pytorch 1.4 with CUDA-devel</td> </tr> <tr> <td>Pytorch 1.4 with CUDA-runtime</td> </tr> <tr> <td>R Script</td> </tr> <tr> <td>Tensorflow 1.15.2</td> </tr> <tr> <td>Tensorflow 1.15.2 with GPU</td> </tr> <tr> <td>Tensorflow 2.3.0</td> </tr> <tr> <td>Tensorflow 2.3.0 with GPU</td> </tr> </table>		Search...	Anaconda (2020.07) with Python 3.x	Pandas 1.1.1	Pytorch 1.4 with CUDA-devel	Pytorch 1.4 with CUDA-runtime	R Script	Tensorflow 1.15.2	Tensorflow 1.15.2 with GPU	Tensorflow 2.3.0	Tensorflow 2.3.0 with GPU								
Search...																			
Anaconda (2020.07) with Python 3.x																			
Pandas 1.1.1																			
Pytorch 1.4 with CUDA-devel																			
Pytorch 1.4 with CUDA-runtime																			
R Script																			
Tensorflow 1.15.2																			
Tensorflow 1.15.2 with GPU																			
Tensorflow 2.3.0																			
Tensorflow 2.3.0 with GPU																			
<table border="1"> <tr> <td colspan="2">Add new Runtime Image</td> </tr> <tr> <td colspan="2">All fields marked with an asterisk are required. [Learn more ...]</td> </tr> <tr> <td>Name *</td> <td>Airflow Python Runner</td> </tr> <tr> <td colspan="2">Description</td> </tr> <tr> <td>Tags</td> <td>Add Tag +</td> </tr> <tr> <td colspan="2">Source</td> </tr> <tr> <td>Image Name *</td> <td>quay.io/ml-aml-workshop/airflow-python-runner:0.0.8</td> </tr> <tr> <td>Image Pull Policy</td> <td>IfNotPresent</td> </tr> <tr> <td colspan="2">SAVE & CLOSE</td> </tr> </table>		Add new Runtime Image		All fields marked with an asterisk are required. [Learn more ...]		Name *	Airflow Python Runner	Description		Tags	Add Tag +	Source		Image Name *	quay.io/ml-aml-workshop/airflow-python-runner:0.0.8	Image Pull Policy	IfNotPresent	SAVE & CLOSE	
Add new Runtime Image																			
All fields marked with an asterisk are required. [Learn more ...]																			
Name *	Airflow Python Runner																		
Description																			
Tags	Add Tag +																		
Source																			
Image Name *	quay.io/ml-aml-workshop/airflow-python-runner:0.0.8																		
Image Pull Policy	IfNotPresent																		
SAVE & CLOSE																			

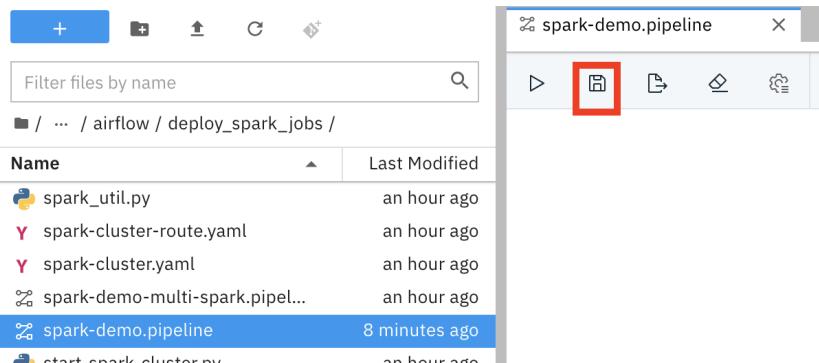
- Again, choose **Runtime Images > +** and fill it in as follows - and click **Save and Close**

Name: Elyra Spark
 Image Name: quay.io/ml-aml-workshop/elyra-spark:0.0.4
 Image Pull Policy: IfNotPresent

as shown here:

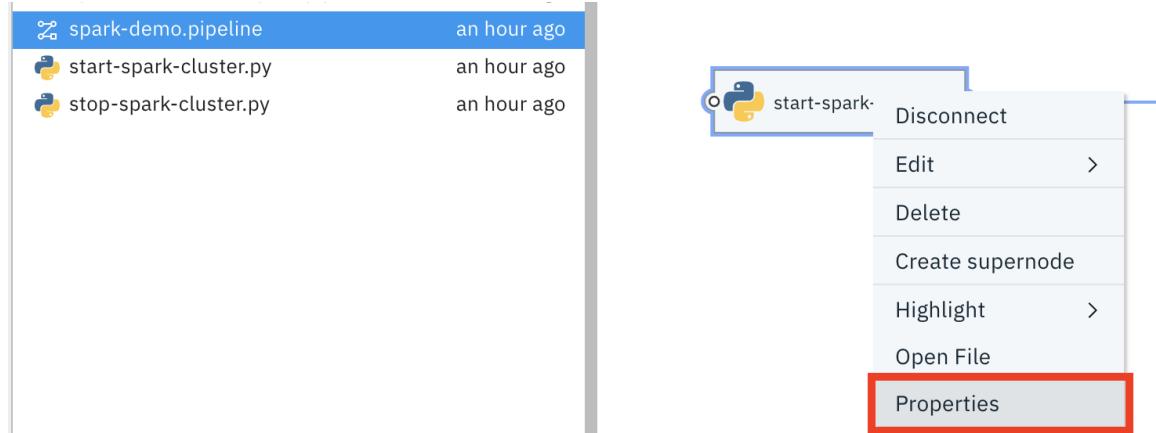


- Save your work by clicking on the **Save** button



Each component in the pipeline can be configured from the pipeline GUI - which we'll do now beginning with the start-spark-cluster element of the pipeline.

- Right-click on **start-spark-cluster** and choose **Properties**



- Select these values as shown in the screenshot below
 - Runtime Image:** Airflow Python Runner
 - File Dependencies**
 - spark_util.py
 - spark-cluster-route.yaml
 - spark-cluster.yaml
 - Environment Variables**
 - S3_ENDPOINT_URL=<http://minio-ml-workshop:9000>
 - SPARK_CLUSTER=**userXX**-cluster
replacing **userXX** with your username and ensuring your cluster has 2 worker nodes
(if a second SPARK_CLUSTER is there, **delete it**).
 - WORKER_NODES=2
 - Output Files**
 - spark-info.txt

As shown here:



start-spark-cluster.py

Filename (required)
start-spark-cluster.py

Runtime Image (required)

CPU GPU RAM(GB)

File Dependencies

spark_util.py
spark-cluster-route.yaml
spark-cluster.yaml

Include Subdirectories in Dependencies

Environment Variables

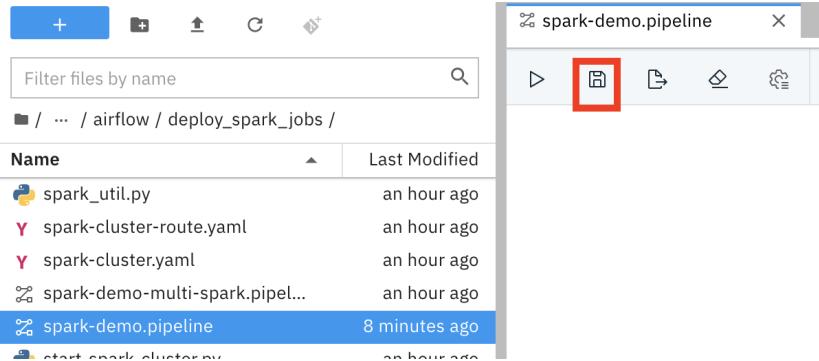
SPARK_CLUSTER=user30-cluster
WORKER_NODES=2
S3_ENDPOINT_URL=http://minio-ml-

Output Files

spark-info.txt



- Save your work by clicking on the **Save** button



- Do the same with Merge_Data, right-clicking and filling as follows:
- Select these values as shown in the screenshot below
 - **Runtime Image: Elyra Spark**
 - File Dependencies
 - spark_util.py
 - spark-cluster-route.yaml
 - hyper_parameters.py
 - Environment Variables
 - S3_ENDPOINT_URL=<http://minio-ml-workshop:9000>
 - SPARK_CLUSTER=userXX-cluster
 - WORKER_NODES=2replacing **userXX** with your username and ensuring your cluster has 2 worker nodes

As shown here:

- **hyper_parameters.py** as a *File Dependency*

Merge_Data.ipynb 

filename (required)

Runtime Image (required) 

Elyra Spark 

CPU  GPU  RAM(GB) 

File Dependencies 

spark_util.py  
spark-cluster-route.yaml  
hyper_parameters.py  

Include Subdirectories in Dependencies 

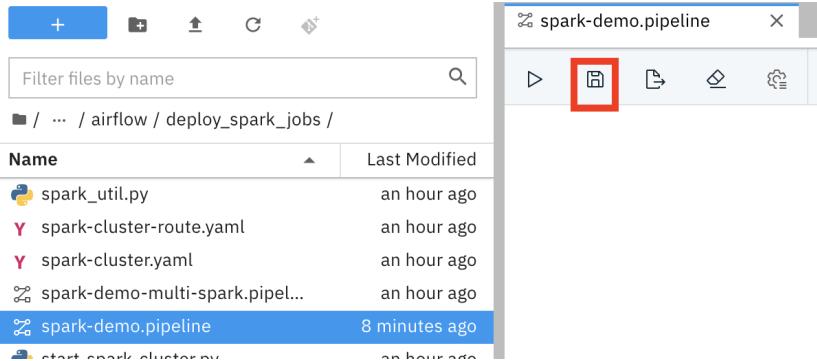
Environment Variables 

SPARK_CLUSTER=user30-cluster 
WORKER_NODES=2 
URL=http://minio-ml-workshop:9000 

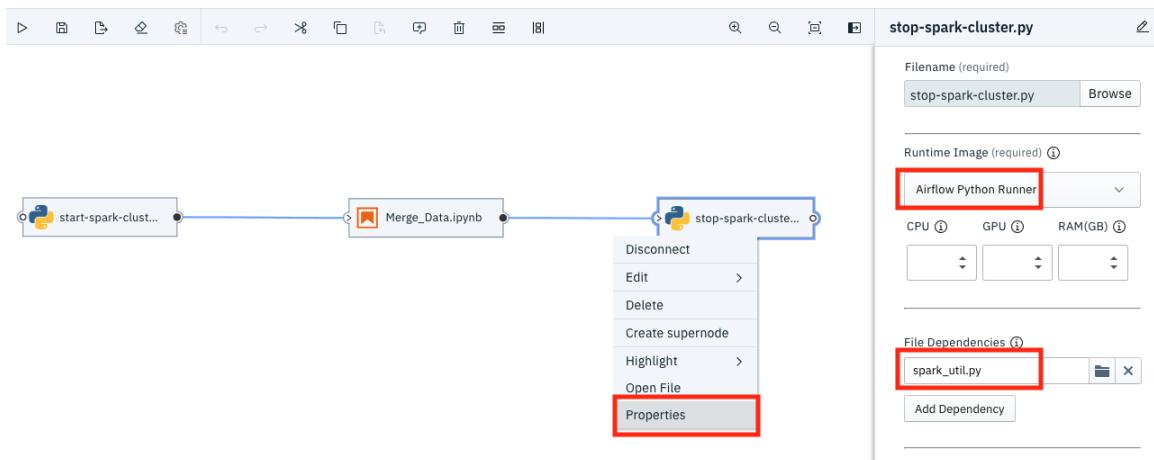
Output Files 



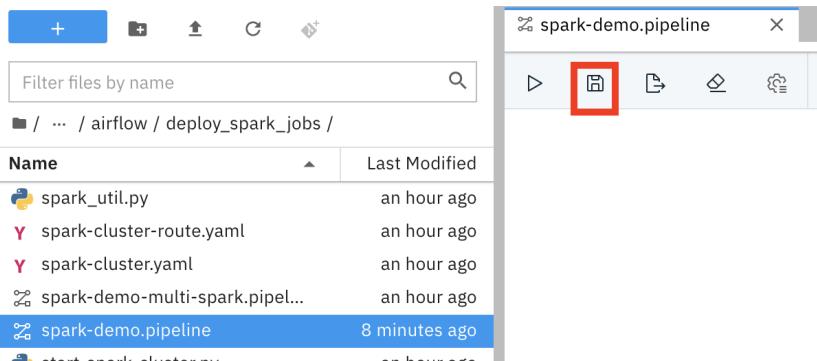
- Save your work by clicking on the **Save** button



- Finally, do the same with stop-spark-cluster - which has fewer still values



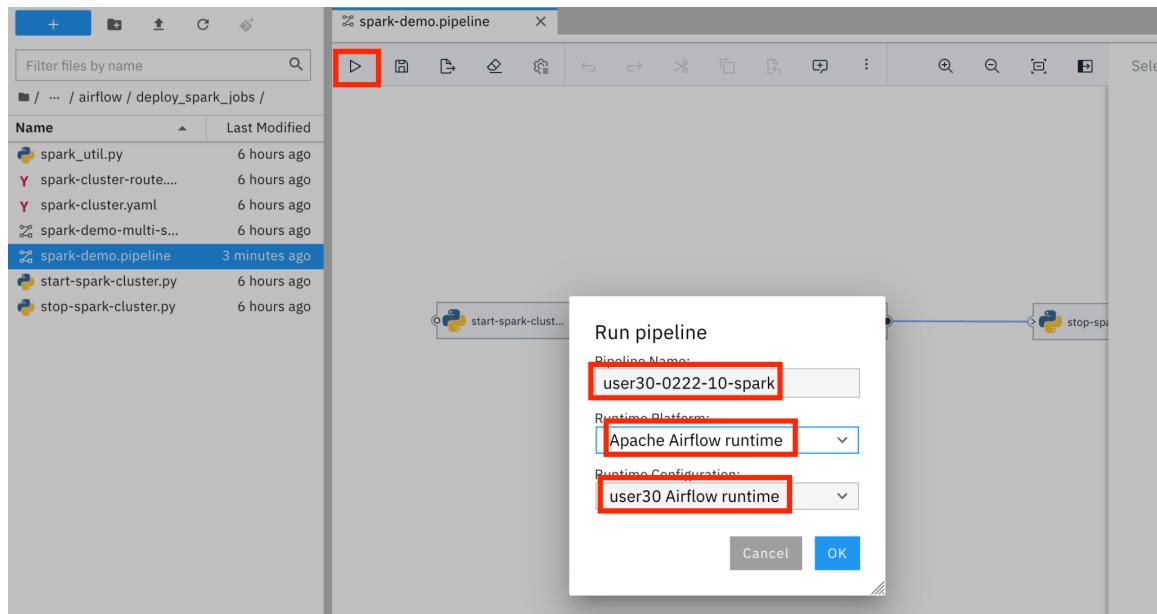
- Save your work by clicking on the **Save** button



Run Pipeline

You can now run your pipeline. Click the Play button as shown. A useful naming convention is to enter a name in a format beginning with your username - followed by the month and day and the count of runs that day. This sends each run to the bottom of the list on the Airflow GUI.

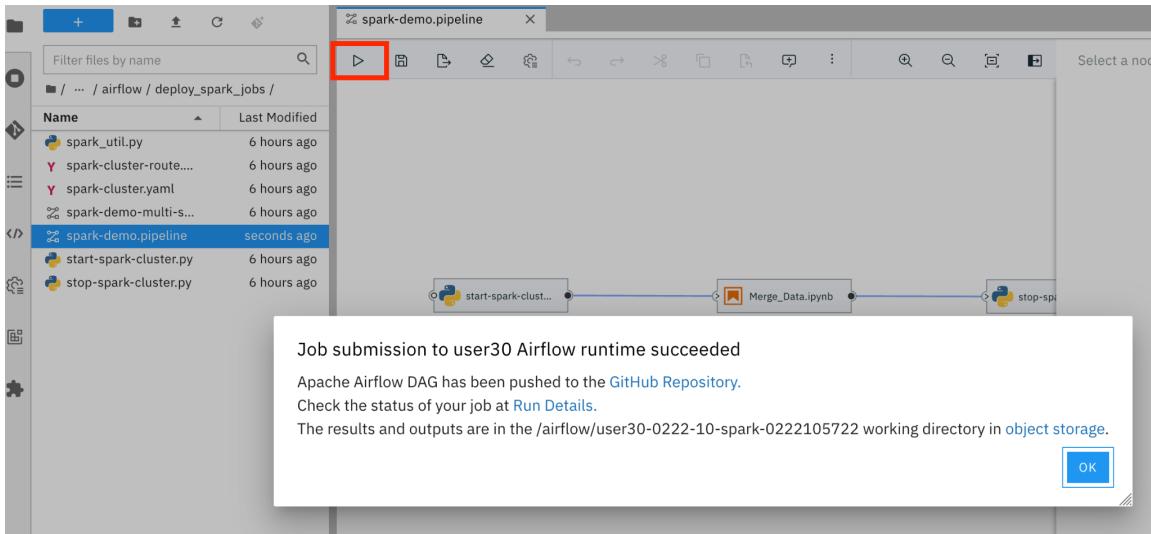
userXX-MMDD-01-spark, in my case **user30-0223-01-spark**. Also choose your Airflow Runtime and Runner as shown:



Two pop-ups will show - including this one. OK them



Red Hat



Now let's take a look at the Spark job we pushed to our workflow engine Airflow.

In a browser open your **Airflow Route URL** as described above in step 30. After logging in with your OpenShift credentials. click DAGs and you'll be presented with the a page like this:

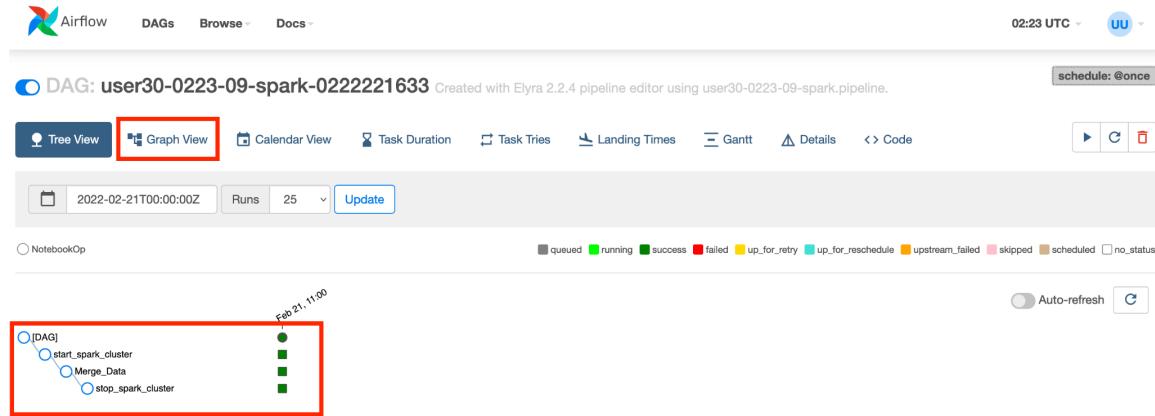
The screenshot shows the 'DAGs' page in the Airflow web interface. At the top, there are tabs for 'DAGs' (which is selected and highlighted with a red box), 'Browse', and 'Docs'. On the right, there is a timestamp '02:18 UTC' and a user icon. Below the tabs, there is a search bar with the placeholder 'Filter DAGs by tag' and a dropdown menu with the value 'user30' (also highlighted with a red box). The main table lists DAGs with columns: 'DAG', 'Owner', 'Runs', 'Schedule', 'Last Run', 'Recent Tasks', 'Actions', and 'Links'. One row is selected, showing the DAG name 'user30-0222-10-spark-0222105722', owner 'airflow', schedule '@once', last run '2022-02-21, 00:00:00', and recent tasks (represented by green circles). Action buttons for the selected DAG include a play icon, a refresh icon, a trash icon, and a more options icon.

You will see many DAGs, representing previous pipeline runs. Filter on your username in the right hand side box to see only yours.

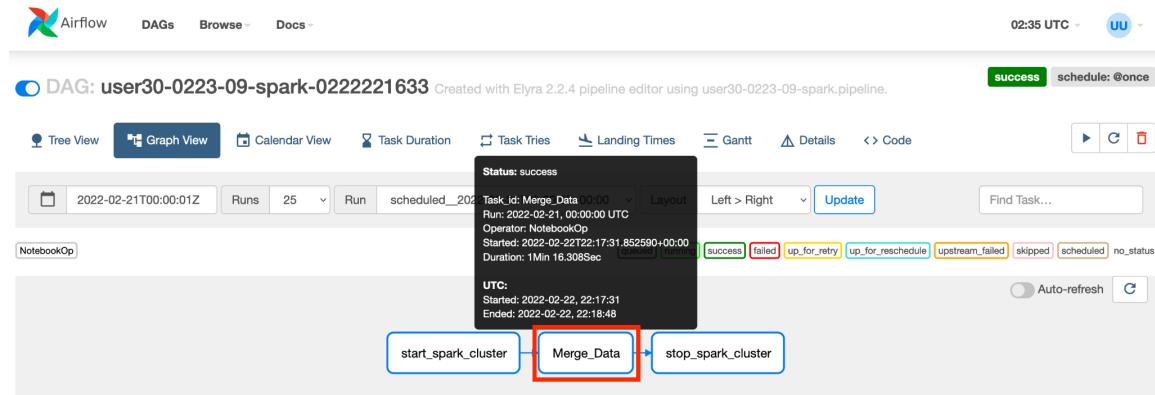


Red Hat

Drill into your Spark job DAG by clicking on it - the one you just ran. You can see the stages and the success or otherwise of the various stages in your pipeline. Click on Graph View for a more detailed view



Click on the Merge Data stage





You have many options to control and re-run stages for example. You can also view the logs for the stage from here as well

Task Instance: Merge_Data
at: 2022-02-21T00:00:00+00:00

[Instance Details](#) [Rendered](#) [Log](#) [All Instances](#) [Filter Upstream](#)

Task Actions

[Ignore All Deps](#) [Ignore Task State](#) [Ignore Task Deps](#) [Run](#)

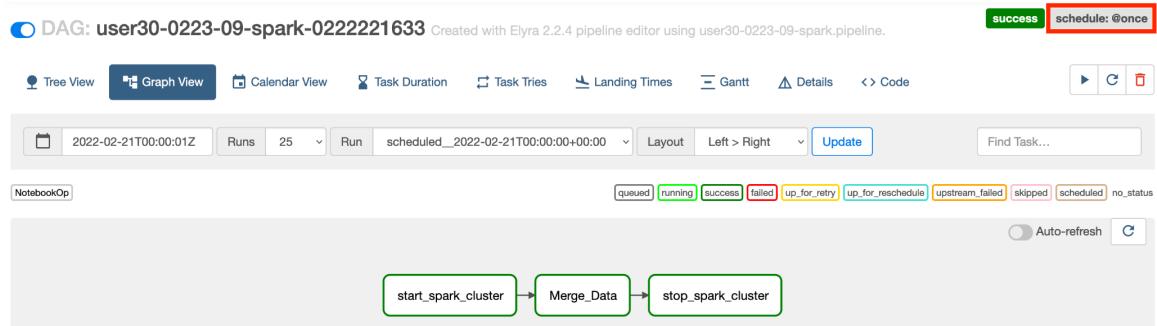
[Past](#) [Future](#) [Upstream](#) [Downstream](#) [Recursive](#) [Failed](#) [Clear](#)

[Past](#) [Future](#) [Upstream](#) [Downstream](#) [Mark Failed](#)

[Past](#) [Future](#) [Upstream](#) [Downstream](#) [Mark Success](#)

[Close](#)

Finally, by clicking **schedule:@once** you can schedule this job to run on a periodic basis - fully automated through cron.





Click the blue **+** icon to do this. We'll leave this as an exercise for you, the user.

List Dag Run

Search

Actions **+**

Add a new record

	State	Dag Id	Execution Date	Run Id	Run Type	Queued At	Start Date	End Date	External
	user30-0223-09- spark-022221633	scheduled	2022-02-21, 00:00:00	scheduled_2022-02-21T00:00:00+00:00	scheduled	2022-02-22 22:16:41.621392+00:00	2022-02-22, 22:16:41	2022-02-22, 22:19:10	False

Record Count: 1

That concludes our Airflow pipeline and workflow section. Hopefully you agree this is a very powerful tool - bringing agility and autonomy to the data engineer!

Recap and Cleanup

Before we move on, let's have a look at how OpenShift provisioned a Spark cluster for you to do your work.

- Open **Tab 1**.

Pods - Red Hat OpenShift Container Platform

Administrator

Home

Projects

Search

Explore

Events

Operators

Workloads

Pods

Deployments

DeploymentConfigs

StatefulSets

Secrets

Project: ml-workshop

Pods

Filter Name user1

Name user1 Clear all filters

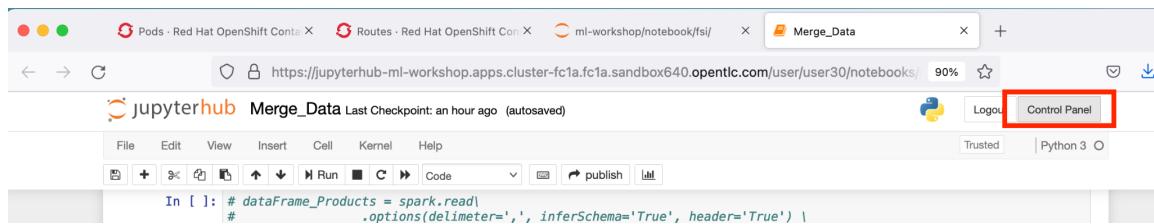
Name	Status	Ready	Restarts	Ow
customerchurnpredictor-1-build	Completed	0/1	0	B
jupyterhub-nb-user1	Running	1/1	0	No
odh-message-bus-zookeeper-1	Running	1/1	0	SS
spark-cluster-user1-m-7p2ps	Running	1/1	0	RC
spark-cluster-user1-w-c5bvx	Running	1/1	0	RC
spark-cluster-user1-w-x2tph	Running	1/1	0	RC



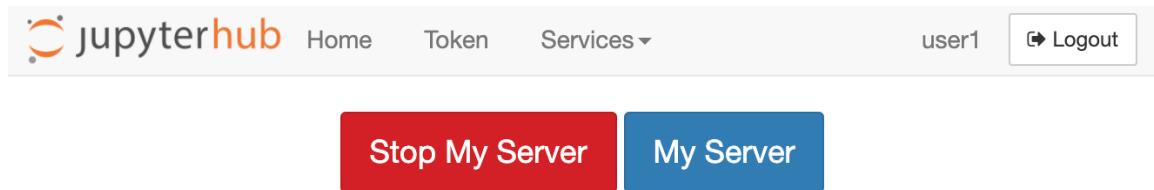
OpenShift displays all of the pods for your username. Here you can observe that a Spark cluster was started for the Data Engineering work. This is an example of the self-service capabilities OpenShift brings to the table and unlocks the productivity of your team.

We don't need this notebook any more so we will shut it down - along with the Spark cluster that was created with it. This is another example of how OpenShift provides efficient use of resources by returning them to the pool when they are no longer needed.

- Open the Jupyterhub notebook tab.
- Click **Control Panel**



Jupyterhub displays the server-control page.



- Click **Stop My Server**.

Jupyterhub commences releasing all of the OpenShift resources.

To watch this in action:

- Open **Tab 1**

Observe your Spark pods being destroyed.



Name	Status	Ready	Restarts	Owner	Memory	CPU
P spark-cluster-user30-m-qg6qk	Terminating	1/1	0	RC spark-cluster-user30-m	174.9 MiB	0.004 cores
P spark-cluster-user30-w-65i98	Terminating	1/1	0	RC spark-cluster-user30-w	164.6 MiB	0.004 cores
P spark-cluster-user30-w-l5vtd	Terminating	1/1	0	RC spark-cluster-user30-w	158.3 MiB	0.004 cores

You can see each of the pods terminating. This is a powerful demonstration of OpenShift's self service capabilities. No waiting for IT to provision you a server, no waiting around for access to a scarce Spark server. All self service, on demand, and those resources returned back to the central pool when finished.

Congratulations - you've completed the first lab. Now move to the next one: **AI/ML on OpenShift Workshop - Lab 2 - Data Science.**