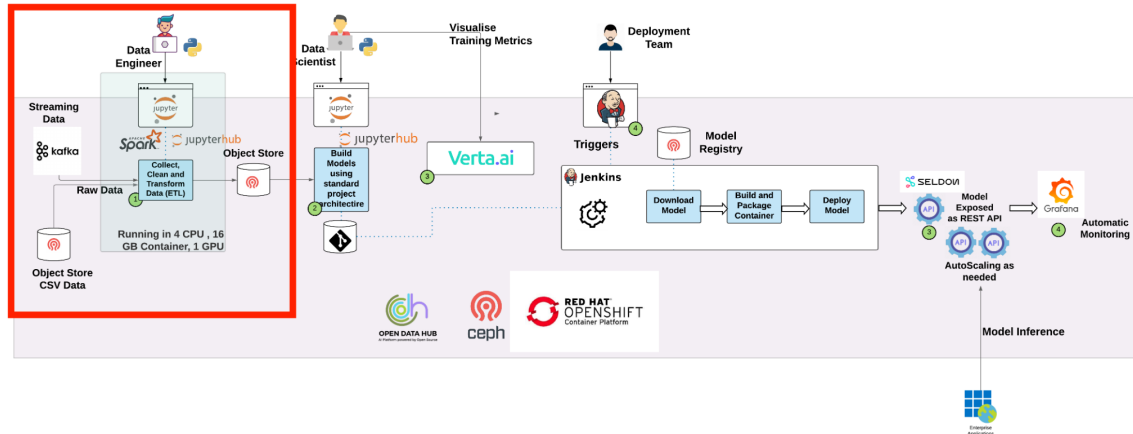


Lab 1 - Data Engineering

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.



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 login page features a white box on a dark background. Inside the box, the text 'Log in to your account' is at the top. Below it are two input fields: 'Username' and 'Password', both marked with a red asterisk. A blue 'Log in' button is at the bottom of the box. To the right of the box, the Red Hat logo and 'OpenShift Container Platform' text are displayed, along with a welcome message: 'Welcome to Red Hat OpenShift Container Platform.'

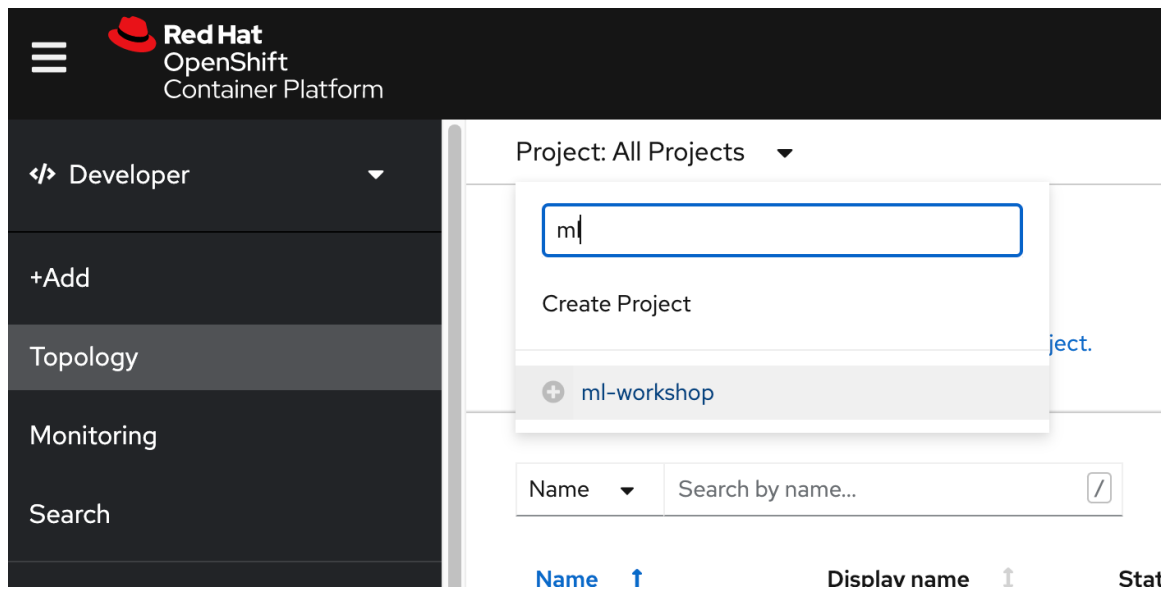
The main OpenShift console will be displayed

The screenshot shows the OpenShift console interface. On the left is a dark sidebar with navigation options: Developer, +Add, Topology, Monitoring, Search, Builds, Helm, Project, and ConfigMaps. The main area has a header with the Red Hat logo and 'OpenShift Container Platform'. Below the header, there's a 'Project: All Projects' dropdown. The 'Topology' section is active, showing a message: 'Select a Project to view the topology or [create a Project](#).' Below this is a search bar and a table of projects.

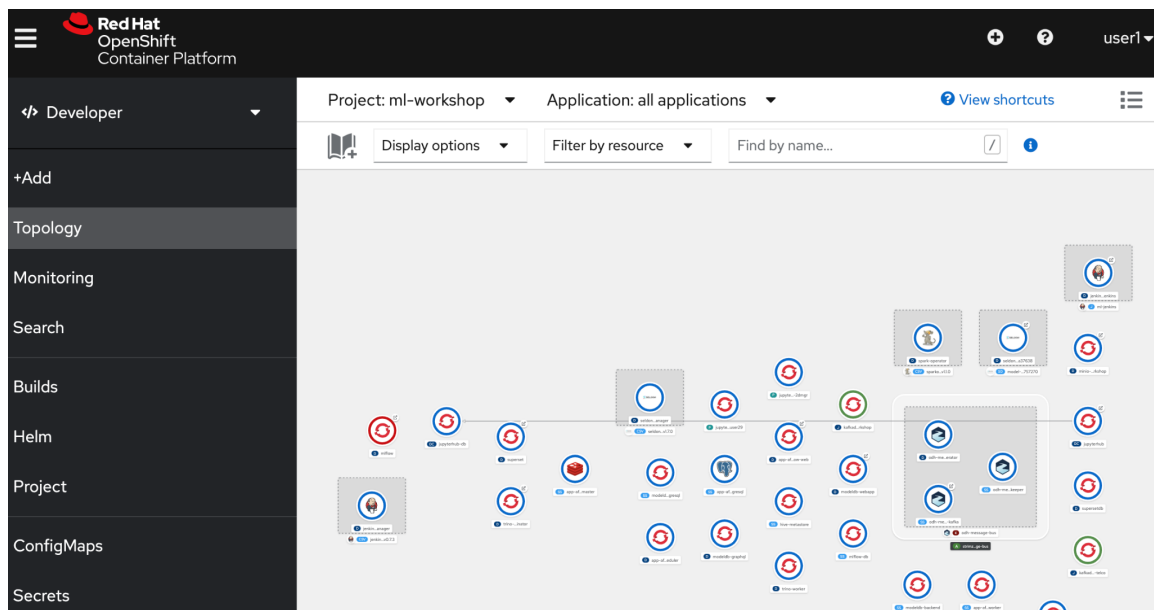
Name	Display name	Status	Requester	Created
ml-workshop	No display name	Active	opentlc-mgr	11 Oct 2021, 09:14
user1-project	No display name	Active	opentlc-mgr	11 Oct 2021, 09:44

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.

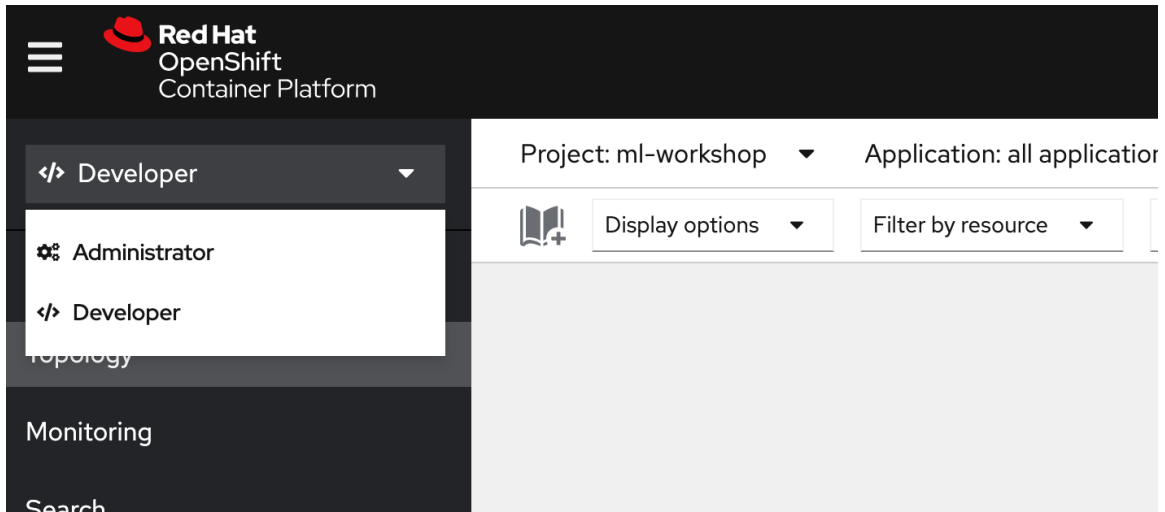


OpenShift will open the project and display the project Topology.





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.

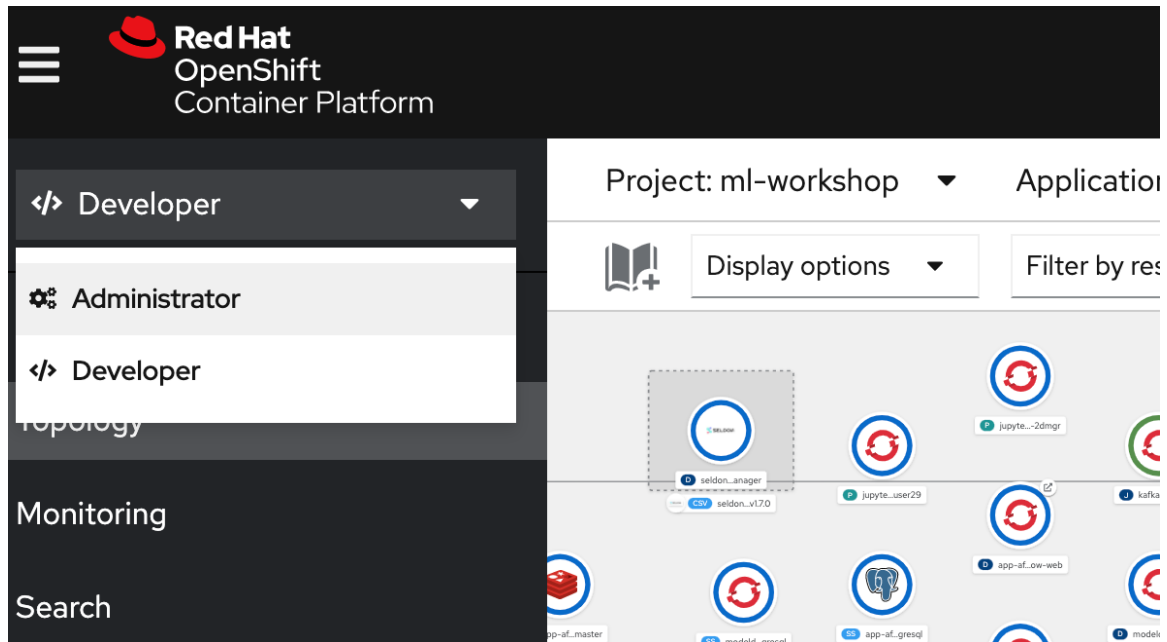


You have now completed accessing the lab environment. Proceed to the next section.

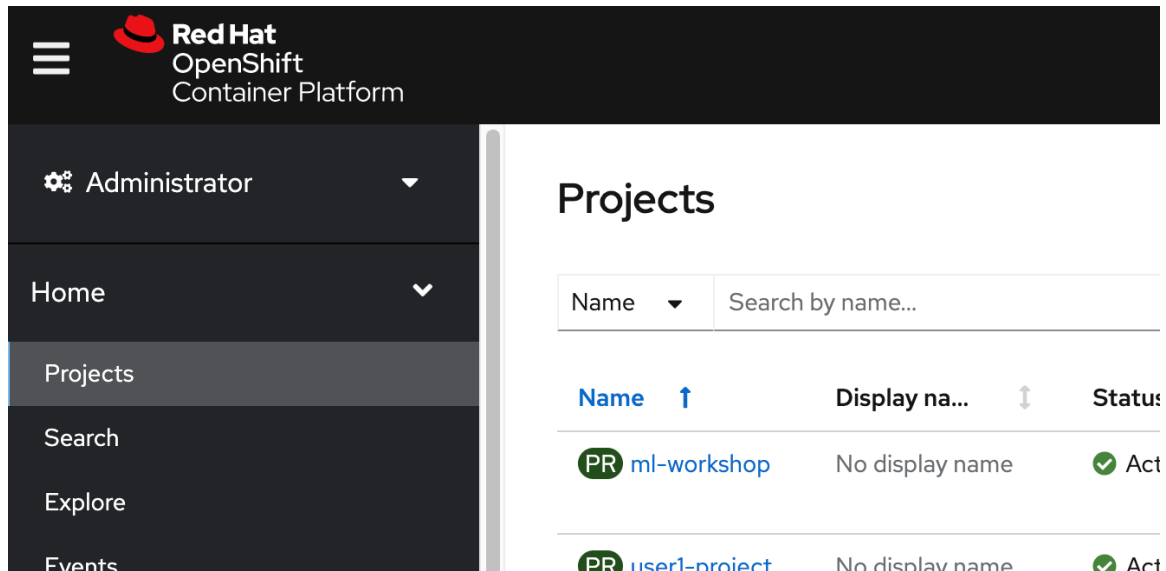
Instructions for the Spark workshop

Change to the **Administrator** perspective:

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



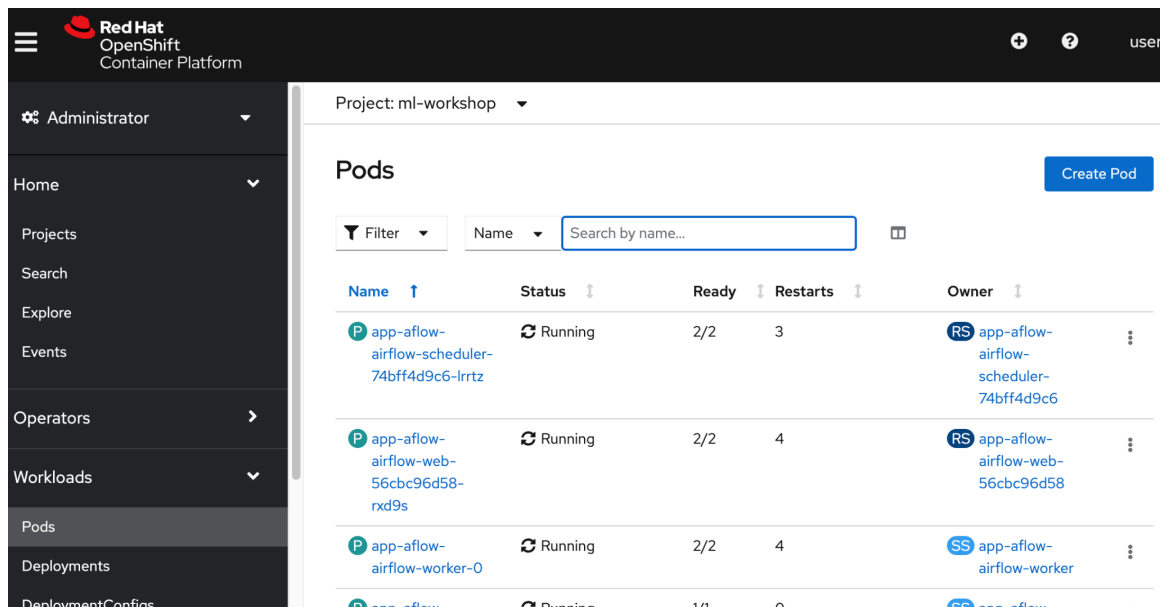
The OpenShift console displays the **Administrator** perspective.



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.

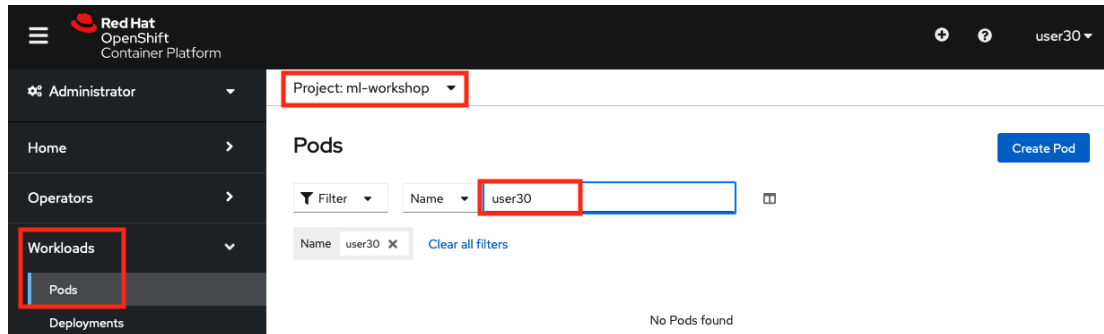
Tab 1

9. **Click Workloads > Pods.**



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

10. Enter your username in the Filter **Search by name** text box

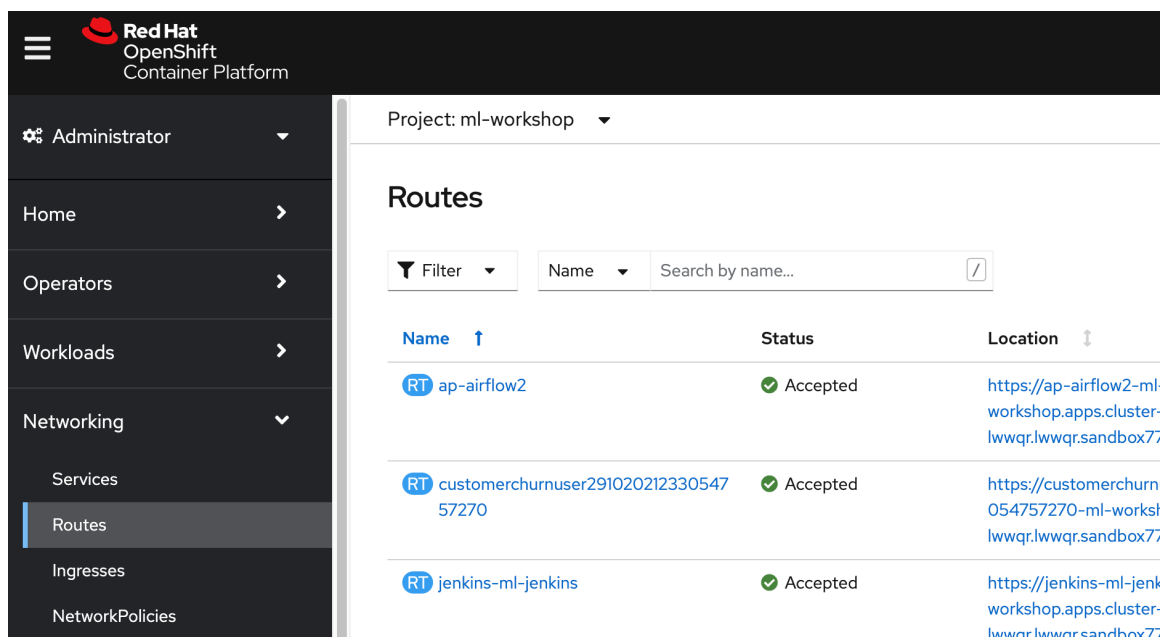


OpenShift will remove all (or almost all) of the pods so you only see the pods related to you.

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

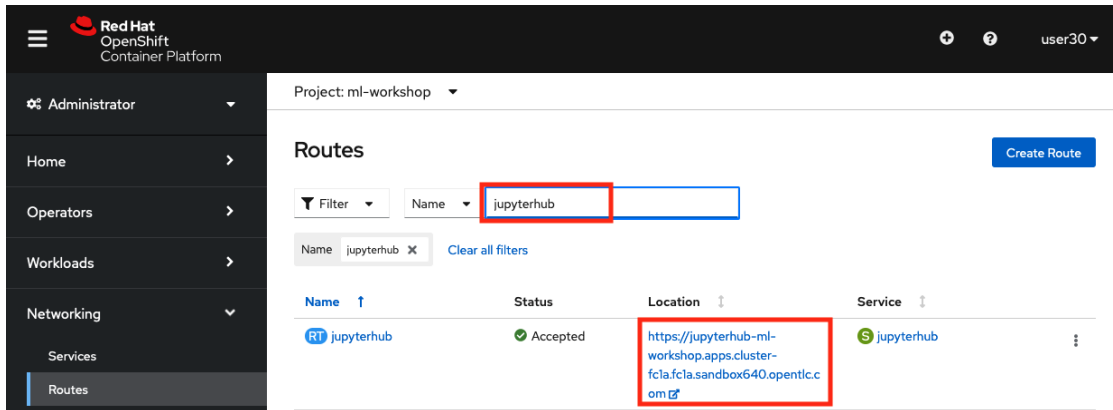
11. Make sure you have the Administrator perspective selected.
12. Click **Networking > Routes**.

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.



Filter the url to find Jupyterhub..

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



Project: ml-workshop

Routes

Filter Name

Name jupyterhub X Clear all filters

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

OpenShift will launch a new browser tab and prompt you to authorise access jupyterhub-hub project 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 (user12

Requested permissions

☒ **user:info**

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

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

Allow selected permissions

Deny




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

15. Click **Minimal Python with Apache Spark**.
16. Expand the **Container size** dropdown listbox and click **Large**.

Start a notebook server

Select options for your notebook server.

Notebook image

- ☐ MLWorkShop Notebook Image
- ☐ MLWorkShop Notebook Image Drift and Outlier
- ☐ Elyra Notebook Image
- ☐ Minimal Python 
Python v3.6.8
- ☒ Minimal Python with Apache Spark
- ☐ Tensorflow Notebook Image
- ☐ MLWorkShop Notebook Image with newer extensions
- ☐ Standard Data Science 
Python v3.8.3
- ☐ Minimal Python 
Python v3.8.3
- ☐ SciPy Notebook Image
- ☐ Minimal Python with Apache Spark and SciPy

Deployment size

Container size

Large

Environment variables

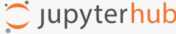
[+ Add more variables](#)

Start server

17. Click **Start Server**.



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

 Home Token Services user1 Logout


Your server is starting up.

You will be redirected automatically when it's ready for you.

2021-10-11T06:38:25Z [Normal] Pulling image "quay.io/odh-jupyterhub/s2i-spark-minimal-notebook@sha256:3d68e863b2575442e99239bcd963e53729e39c2ac10d6fef6ece8174a8de5513"

Event log

After a few minutes JupyterHub displays the notebook.

 Logout Control Panel

Files Running Clusters

Select items to perform actions on them. Upload New ↕


☐ 0 ▾

/

Name ▾

Last Modified

File size

☐  lost+found

a minute ago

Copy notebooks

The first thing we want to do is pull down our notebooks from our repository

<https://github.com/masoodfaisal/ml-workshop>

18. Click **New > Terminal**.



Jupyterhub displays a terminal window.



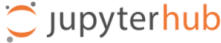
19. Type (or copy and paste) the following command in the terminal window:

`git clone https://github.com/masoodfaisal/ml-workshop`

20. Press **Enter**.



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



LogoutControl Panel

```
sh-4.2$ git clone https://github.com/masoodfaisal/ml-workshop
Cloning into 'ml-workshop'...
remote: Enumerating objects: 7625, done.
remote: Counting objects: 100% (1614/1614), done.
remote: Compressing objects: 100% (844/844), done.
remote: Total 7625 (delta 972), reused 1353 (delta 733), pack-reused 6011
Receiving objects: 100% (7625/7625), 38.25 MiB | 11.83 MiB/s, done.
Resolving deltas: 100% (3429/3429), done.
sh-4.2$
```

Once the repository has finished cloning you can close the browser tab.

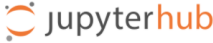
21. Click the **Home** tab in your browser to display a Jupyterhub notebook.

Pods · Red Hat OpenShRoutes · Red Hat OpenHome

https://jupyterhub-ml-v

jupyterhub-ml-workshop.apps.cluster-lwwqr.lwwqr.sandbox779.opentlc.com/user/u...

0 - LabRed HatDeakinAIMLAI/ML DemoMusicAnsibleOther Bookmarks



LogoutControl Panel

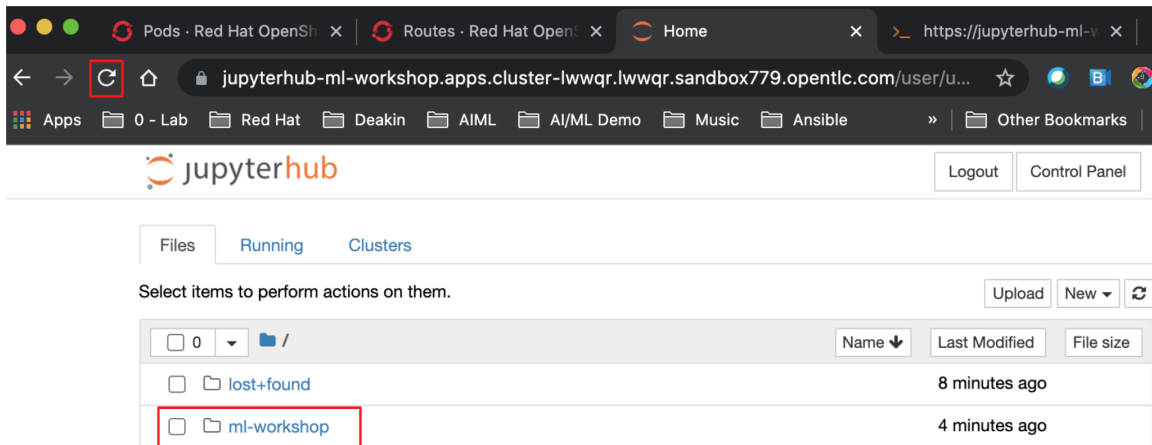
FilesRunningClusters

Select items to perform actions on them.

UploadNew↺

<input type="checkbox"/>	0	/	Name	Last Modified	File size
<input type="checkbox"/>		lost-found		7 minutes ago	
<input type="checkbox"/>		ml-workshop		2 minutes ago	

22. If you do not already see the ml-workshop folder, click **Refresh** in the browser to refresh the page.



Next we must open the folder containing the notebooks for this workshop. This workshop has two different themes (Telco and Financial Services) so you will find these in different locations .

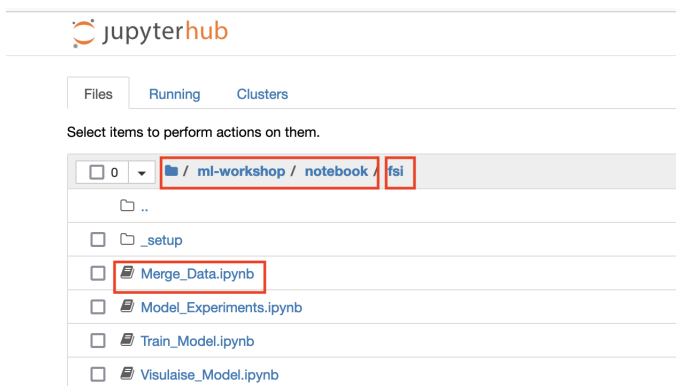
For Financial Service (FSI):

23. Click **ml-workshop > notebook > fsi**.

For Telecommunications:

24. Click **ml-workshop > notebook > telco**.

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



25. Click **Merge_Data.ipynb**.

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

Before we get going, you need to make a small change to the code.

26. Scroll down to the last cell in the notebook.
27. Replace **user29** with your username that you are using for this lab (the same one you used to log on to OpenShift)..

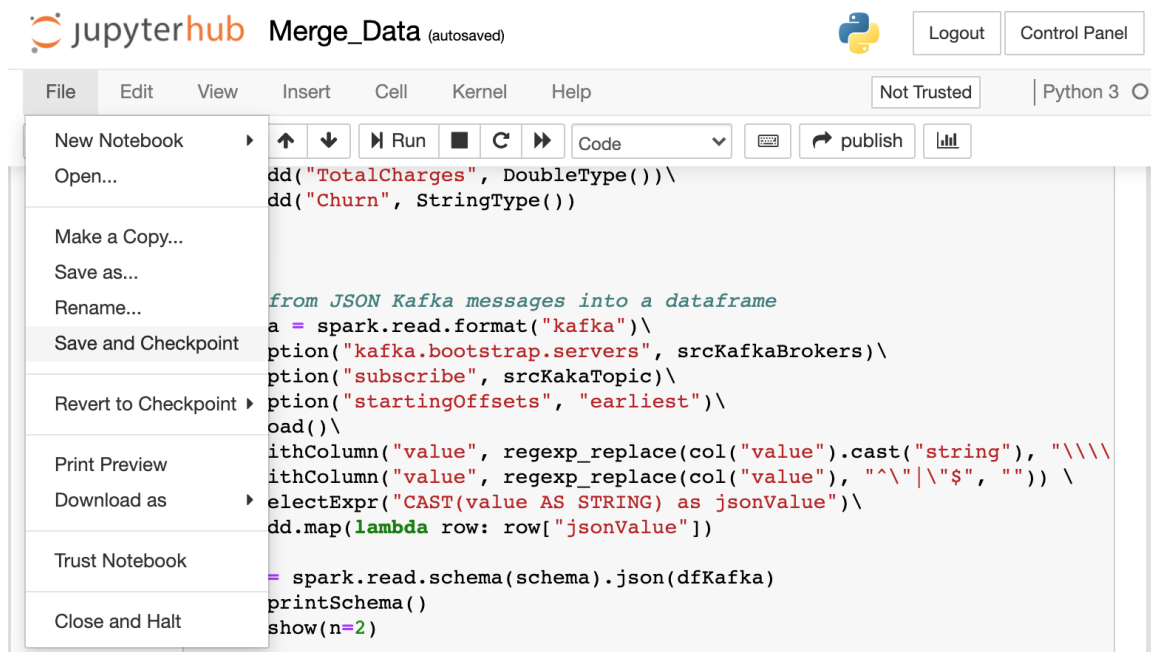
```
user_id = "<your username>"
```

Note - be sure to change this user_id on the next line to your username (something in the range user1 ... user30)

```
In [ ]: user_id = "user29"
file_location = "s3a://data/full_data_csv" + user_id
dataFrom_All.repartition(1).write.mode("overwrite")\
.option("header", "true")\
.format("csv").save(file_location)
```

Save your work

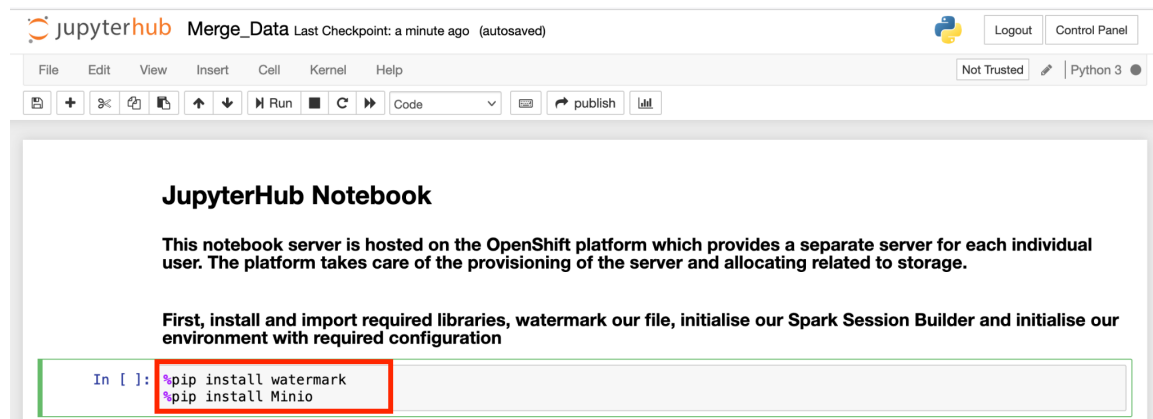
28. Click **File > Save and Checkpoint** to save your work.



Run the notebook

You are now good to go!

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



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

31. 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 > Restart & Run All**.

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

```
1 In [ ]: %pip install watermark
          %pip install Minio
```

```
2 In [ ]: 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 In [ ]: %watermark -n -v -m -g -iv
```

```
In [ ]:
```

```
4 In [ ]: sparkSessionBuilder = SparkSession\
          .builder\
          .appName("Customer Churn ingest Pipeline")
```

```
5 In [ ]: os.environ['PYSPARK_SUBMIT_ARGS'] = \
          '--packages \
          org.postgresql:postgresql:42.2.10,\
          org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5,\
          org.apache.kafka:kafka-clients:2.4.0,\
          org.apache.spark:spark-streaming_2.11:2.4.5,\
          org.apache.hadoop:hadoop-aws:2.7.3 \
          --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 \
          --master spark://' + os.environ['SPARK_CLUSTER'] + ':7077 pyspark-shell '
```

Connect to Spark Cluster provided by OpenShift Platform

```
6 In [ ]: spark = sparkSessionBuilder.getOrCreate()
          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 create a Spark session, a precursor to firing up our own Spark server.
5. 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.
6. Here we actually start our Spark server. This cell can take several minutes to start.


```
7 In [ ]: dataframe_Customer = spark.read\
          .options(delimiter=',', inferSchema='True', header='True') \
          .csv("s3a://rawdata/Customer-Churn_P1.csv")
          dataframe_Customer.printSchema()
```

```
8 In [ ]: # dataframe_Products = spark.read\
          # .options(delimiter=',', inferSchema='True', header='True') \
          # .csv("s3a://rawdata/Customer-Churn_P2.csv")
          # dataframe_Products.printSchema()
```

```
9 In [ ]: from pyspark.sql.types import *
          from pyspark.sql.functions import *

          srcKafkaBrokers = "odh-message-bus-kafka-bootstrap:9092"
          srcKakaTopic = "data"

          ..
          ..
```

```
10 In [ ]: dataFrom_All = dataframe_Customer.join(df0bj, "customerID", how="full")
```

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

Note - be sure to change this user_id on the next line to your username (something in the range user1 ... user30)

```
11 In [ ]: user_id = "user29"
          file_location = "s3a://data/full_data_csv" + user_id
          dataFrom_All.repartition(1).write.mode("overwrite")\
          .option("header", "true")\
          .format("csv").save(file_location)
```

```
12 In [ ]: spark.stop()
```

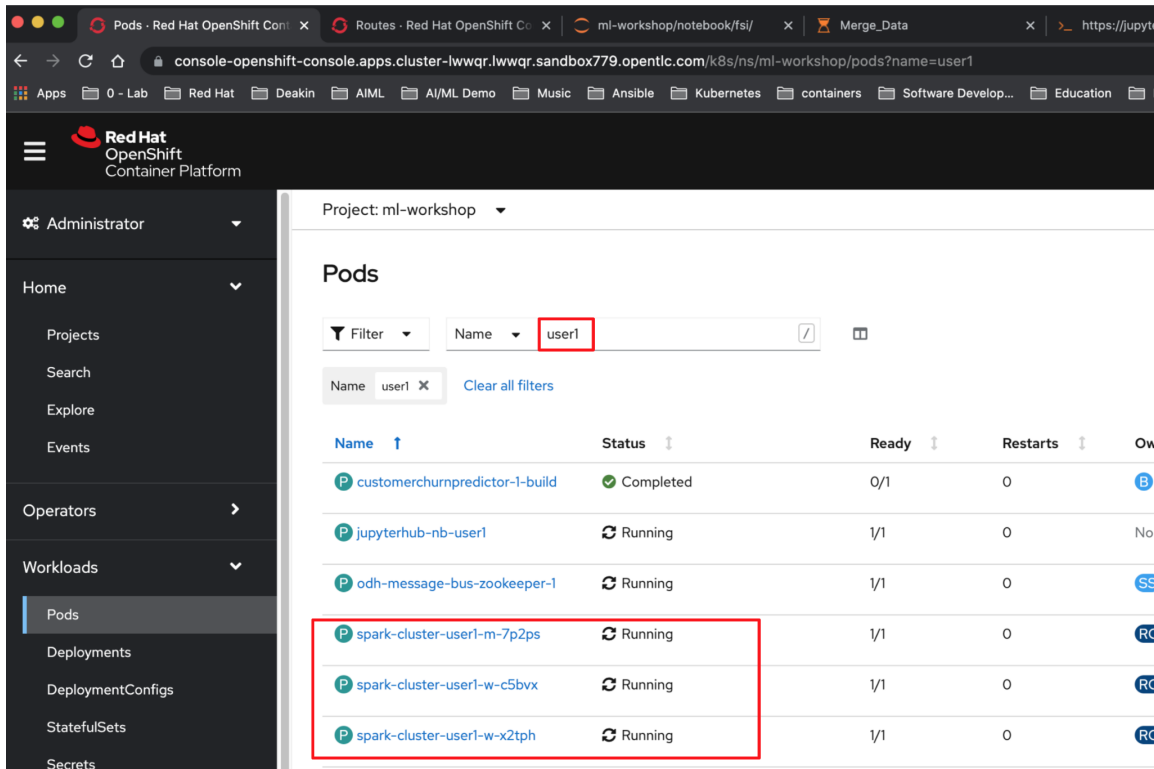
7. Here we pull in our data from S3 - our CSV based demographic data for each of our approximately 7000 customers.
8. Commented out - ignore
9. Here we pull in our data from Kafka - our product consumption data for each of our approximately 7000 customers.
10. We join these 2 datasets, on the common column to each: *customerID*.
11. We push our data to our object store - filename contains our username.
12. We are all done now - we stop our Spark server.

This is our data engineering workshop finished. It's a simple exercise, though the same toolset could be used for much more complex data engineering tasks.

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.

32. Open **Tab 1**.



The screenshot shows the Red Hat OpenShift console interface. The left sidebar contains navigation menus for Administrator, Home, Operators, and Workloads. The 'Pods' page is selected under the 'Workloads' menu. The main content area displays a table of pods for the 'ml-workshop' project, filtered by the name 'user1'. The table has columns for Name, Status, Ready, Restarts, and Owner. A red box highlights three pods related to the Spark cluster:

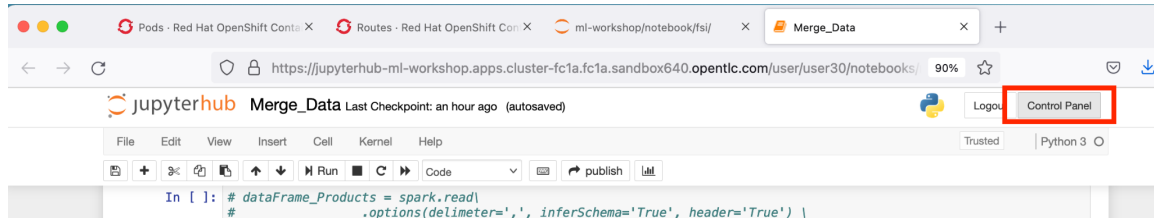
Name	Status	Ready	Restarts	Owner
customerchurnpredictor-1-build	Completed	0/1	0	B
jupyterhub-nb-user1	Running	1/1	0	No
odh-message-bus-zookeeper-1	Running	1/1	0	SS
spark-cluster-user1-m-7p2ps	Running	1/1	0	RC
spark-cluster-user1-w-c5bvx	Running	1/1	0	RC
spark-cluster-user1-w-x2tph	Running	1/1	0	RC

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.

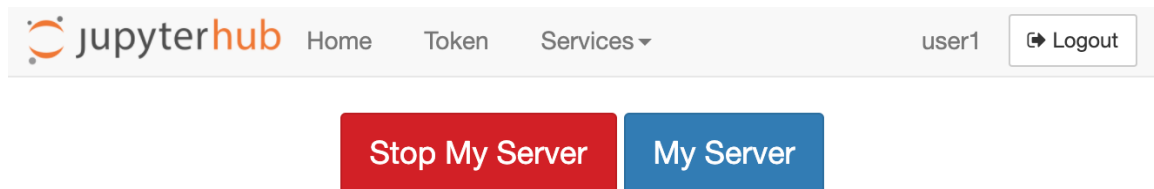
We don't need this notebook any more so we will shut it down - along with the Spark cluster that was created with it. This is another example of how OpenShift provides efficient use of resources by returning them to the pool when they are no longer needed.

33. Open the Jupyterhub notebook tab.

34. Click **Control Panel**



Jupyterhub displays the server-control page.












35. Click **Stop My Server**.

Jupyterhub commences releasing all of the OpenShift resources.

To watch this in action:

36. Open **Tab 1**

Observe your Spark pods being destroyed.

Name ↑	Status ↑	Ready ↑	Restarts ↑	Owner ↑	Memory ↑	CPU ↑
 spark-cluster-user30-m-qg6qk	 Terminating	1/1	0	 spark-cluster-user30-m	174.9 MiB	0.004 cores
 spark-cluster-user30-w-65l98	 Terminating	1/1	0	 spark-cluster-user30-w	164.6 MiB	0.004 cores
 spark-cluster-user30-w-l5vtd	 Terminating	1/1	0	 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.