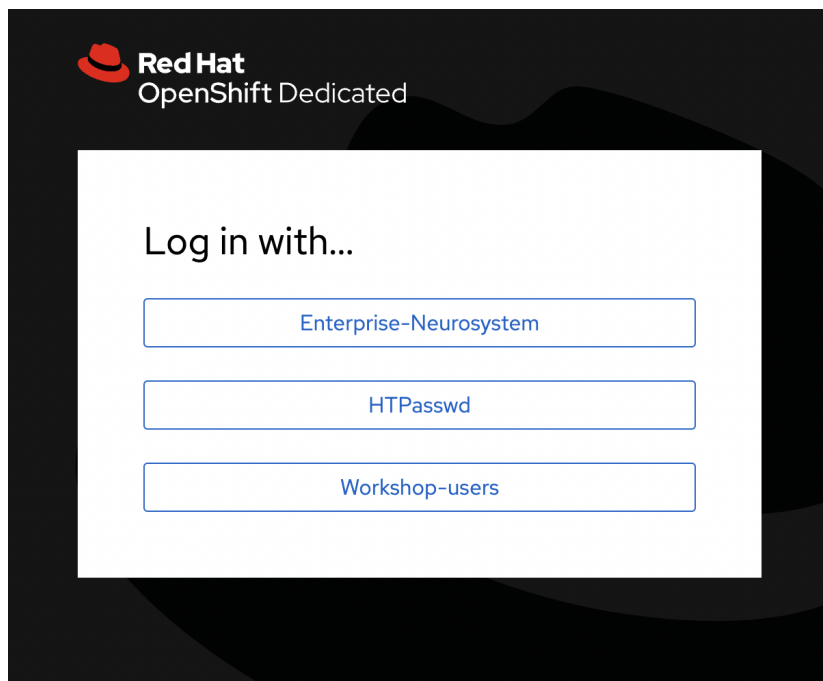


Table of Contents

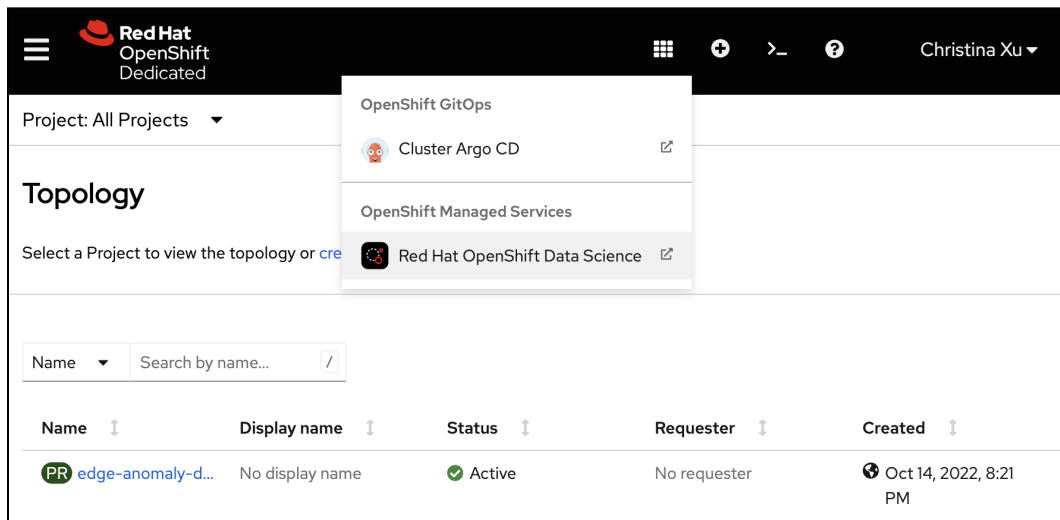
- I. Log into the Red Hat OpenShift Data Science Platform
- II. Start your Jupyter Notebook server
- III. Preparing our data for model usage
- IV. Deploy the Failure Detection web application as a container

I. Log into the Red Hat OpenShift Data Science Platform

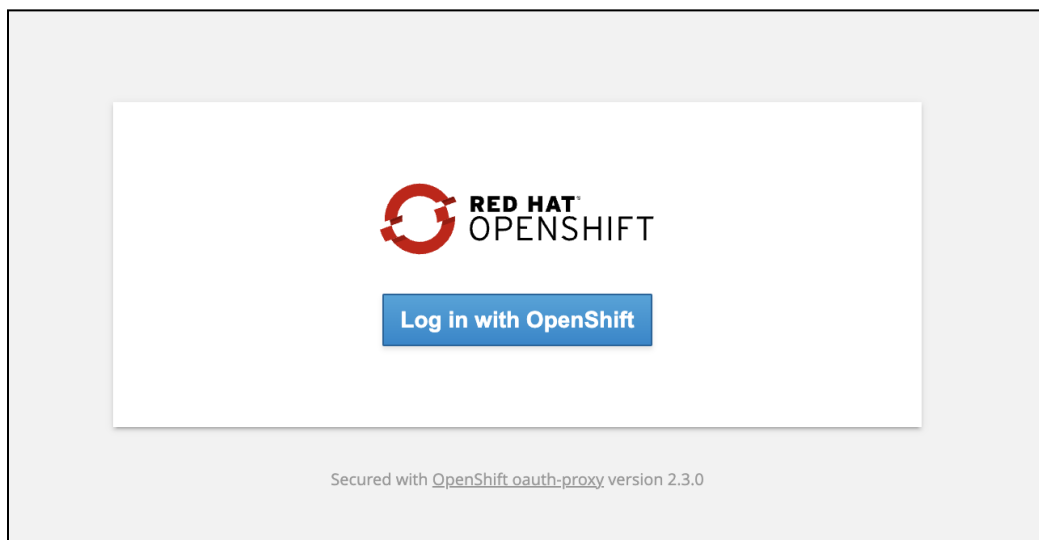
1. Use the following url to access the Red Hat OpenShift Data Science platform:
<https://console-openshift-console.apps.anomaly-cluster.rqdu.p1.openshiftapps.com/>
2. You should see the following login page appear. Click on the '**Workshop-users**' button.



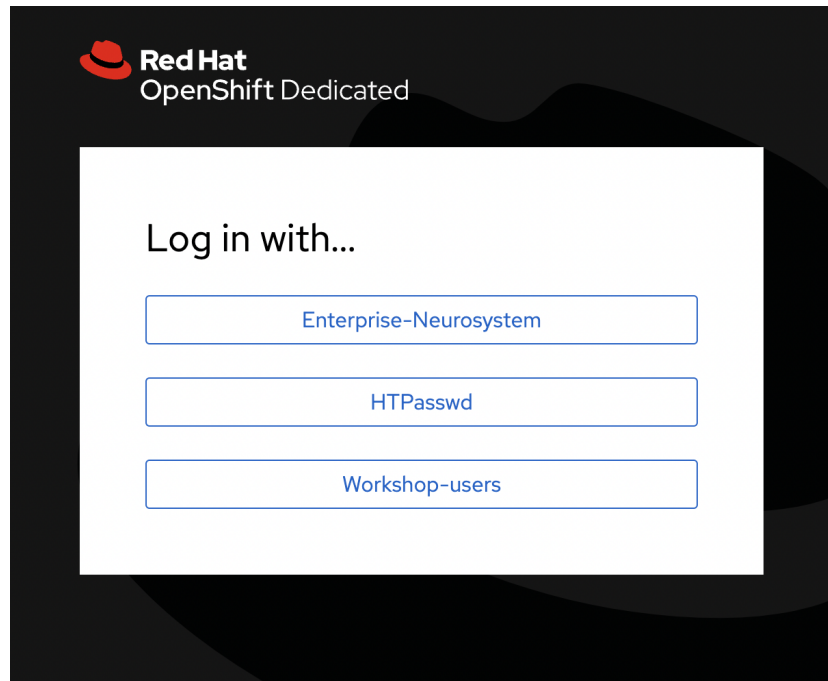
3. Upon successful login, you will see the Red Hat OpenShift Dedicated environment. Navigate to Red Hat OpenShift Data Science by selecting the launcher icon in the upper right corner.



4. Upon clicking the Launcher Icon you will see another login screen. Click the 'Log in with OpenShift' icon.



5. You will see another Login prompt Click the **‘Workshops-users’** button.



6. You may see an ‘Authorize Access’ screen. Click the ‘Allow selected permissions’ button.

Authorize Access

Service account `rhods-dashboard` in project `redhat-ods-applications` is requesting permission to access your account (`christinaexyou`)

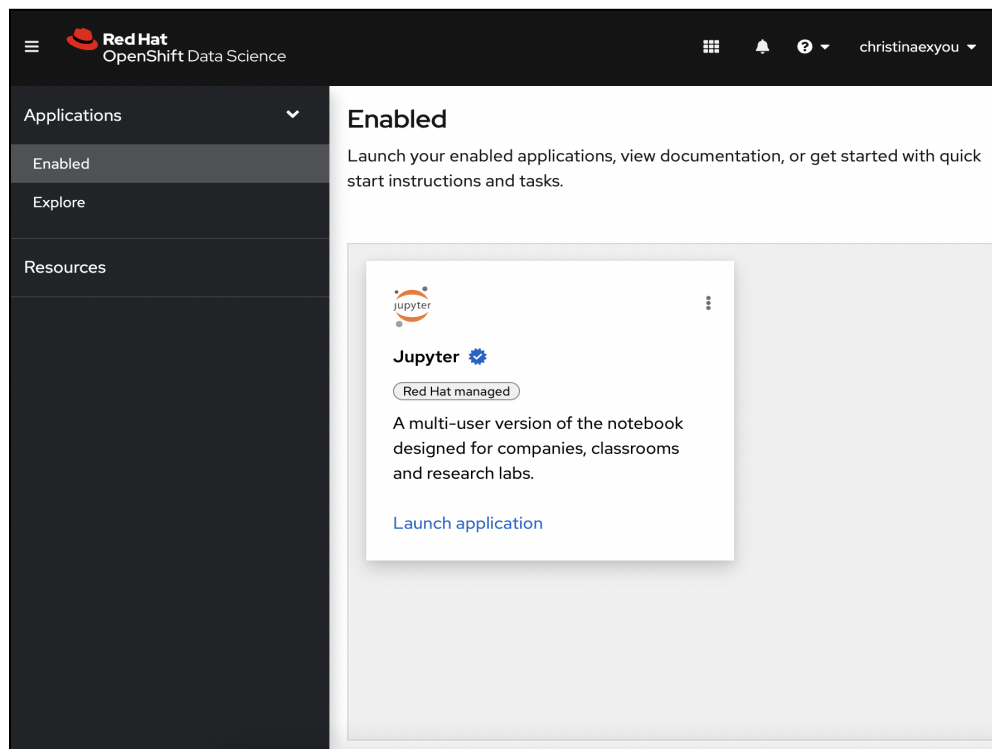
Requested permissions

- ☒ **user:info**
Read-only access to your user information (including username, identities, and group membership)
- ☒ **user:check-access**
Read-only access to view your privileges (for example, "can I create builds?")

You will be redirected to <https://rhods-dashboard-redhat-ods-applications.apps.anomaly-cluster.rqdu.p1.openshiftapps.com/oauth/callback>

Allow selected permissionsDeny

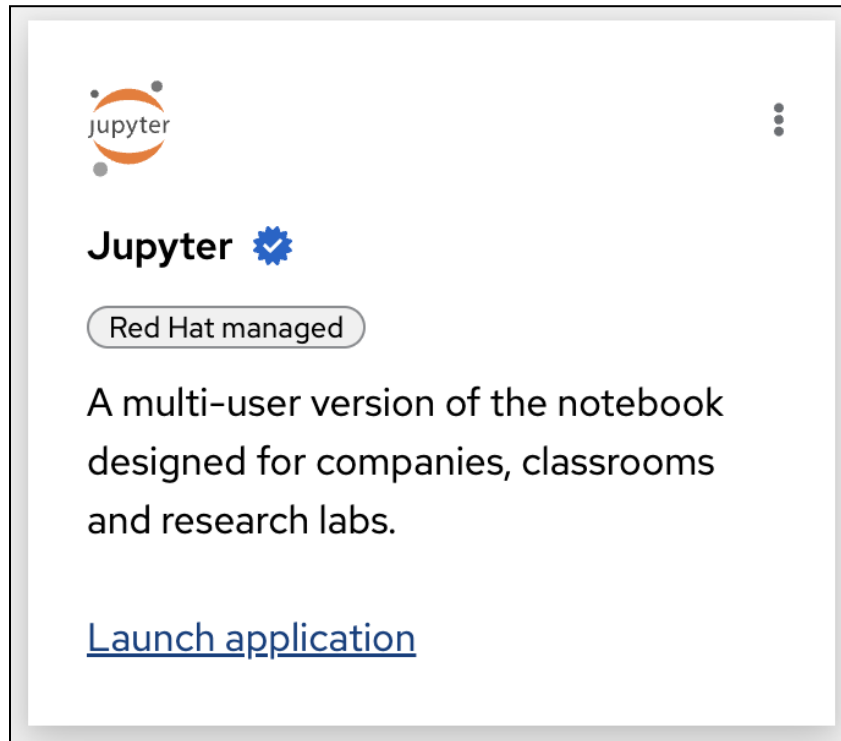
7. You will now be logged into the Red Hat OpenShift Data Science platform.



We are ready to start our JupyterHub notebook server!

II. Start your Jupyter Notebook server

8. Click the “Launch application” url.



9. Choose ‘Standard Data Science’ for your notebook image and a ‘Small’ container size and click on start the ‘Start Server’ button.

Start a notebook server

Select options for your notebook server.

Notebook image

☐ Minimal Python ⓘ
Python v3.8

☒ Standard Data Science ⓘ
Python v3.8

☐ CUDA ⓘ
Python v3.8, CUDA v11.4

☐ PyTorch ⓘ
Python v3.8, PyTorch v1.8, CUDA v11.4

☐ TensorFlow ⓘ
Python v3.8, TensorFlow v2.7, CUDA v11.4

Deployment size

Container Size

Small ▼

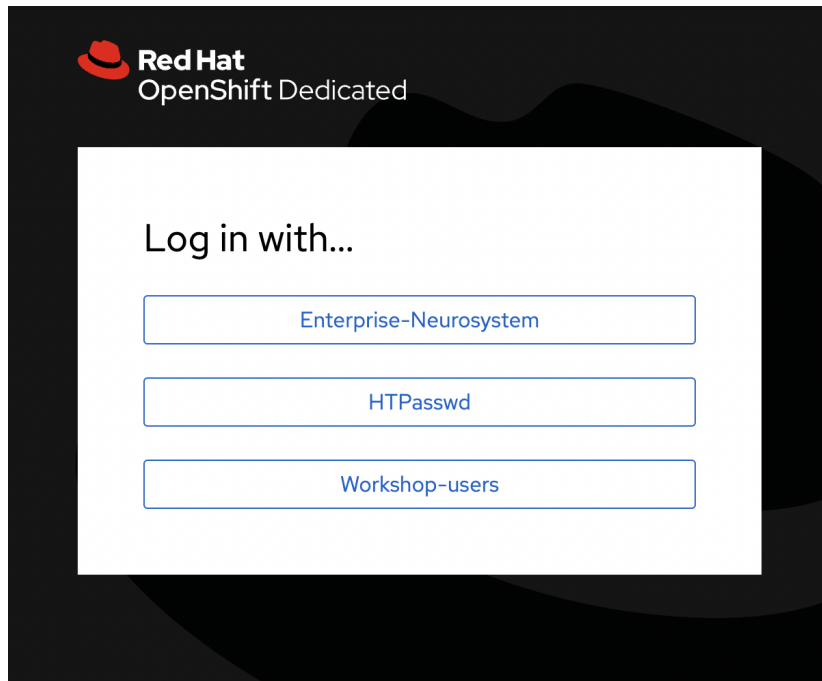
Environment variables

⚙ Add more variables

Start server

Cancel

10. You will be prompted to log in again. Click on 'Workshop-users'



11. You may see an 'Authorize Access' screen. Click the 'Allow selected permissions' button.

Authorize Access

Service account `rhods-dashboard` in project `redhat-ods-applications` is requesting permission to access your account (`christinaexyou`)

Requested permissions

- ☒ **user:info**
Read-only access to your user information (including username, identities, and group membership)
- ☒ **user:check-access**
Read-only access to view your privileges (for example, "can I create builds?")

You will be redirected to <https://rhods-dashboard-redhat-ods-applications.apps.anomaly-cluster.rqdu.p1.openshiftapps.com/oauth/callback>

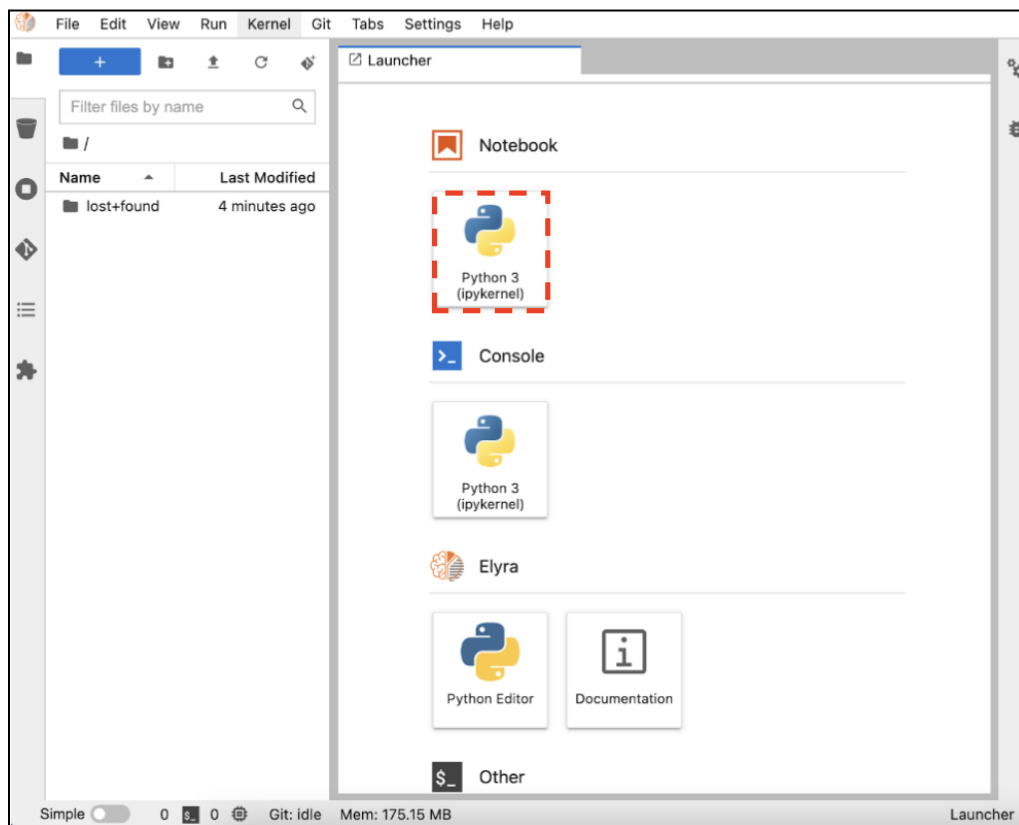
Allow selected permissionsDeny

You are now inside your Jupyter environment. As you can see, it's a web-based environment, but everything you'll do here is in fact happening on the Red Hat OpenShift Data Science cluster. This means that without having to install and maintain anything on your own computer, and without disposing of lots of local resources like CPU and RAM, you can still conduct your Data Science work in this powerful and stable managed environment.

Let's get started with understanding how we can launch a new jupyter notebook and get you familiar with using cells in a jupyter notebook.

Note: the language we will be using with our jupyter notebooks is python.

12. Launch a Jupyter Notebook by selecting the first option in the Launcher.



13. Let's learn how to use jupyter notebook cells. In an empty cell, copy and paste the following command `print('Hello World')`:

```
[ ]: print('Hello World')
```

14. To run the code cell, just select it (click in the cell, or on the left side of it), and click the Run/Play button from the toolbar (you can also press CTRL+Enter to run a cell, or Shift+Enter to run the cell and automatically select the following one).



15. If your output looks like the below, let's move on to preparing our data.

```
[1]: print('Hello World')  
Hello World
```

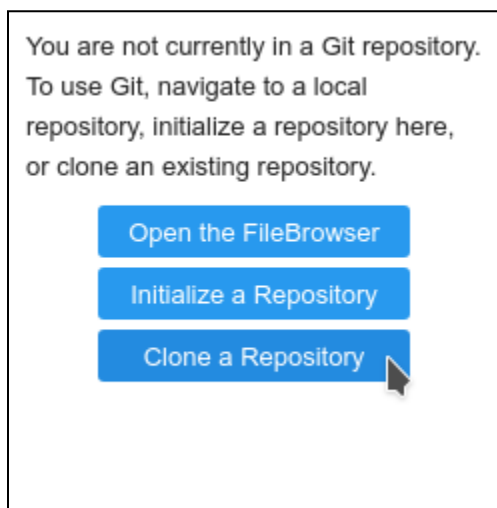

IV. Preparing our data for model use

In the "file-browser" like window you're in right now, you'll find the files and folders that are saved inside your own personal space inside Red Hat OpenShift Data Science. It's pretty empty right now though... So the first thing we will do is to bring the content of the workshop inside this environment.

- On the left toolbar, click on the Git icon:



- Then click on Clone a Repository:



- Enter git the URL, <https://github.com/Enterprise-Neurosystem/edge-prediction-failure>, then click CLONE:

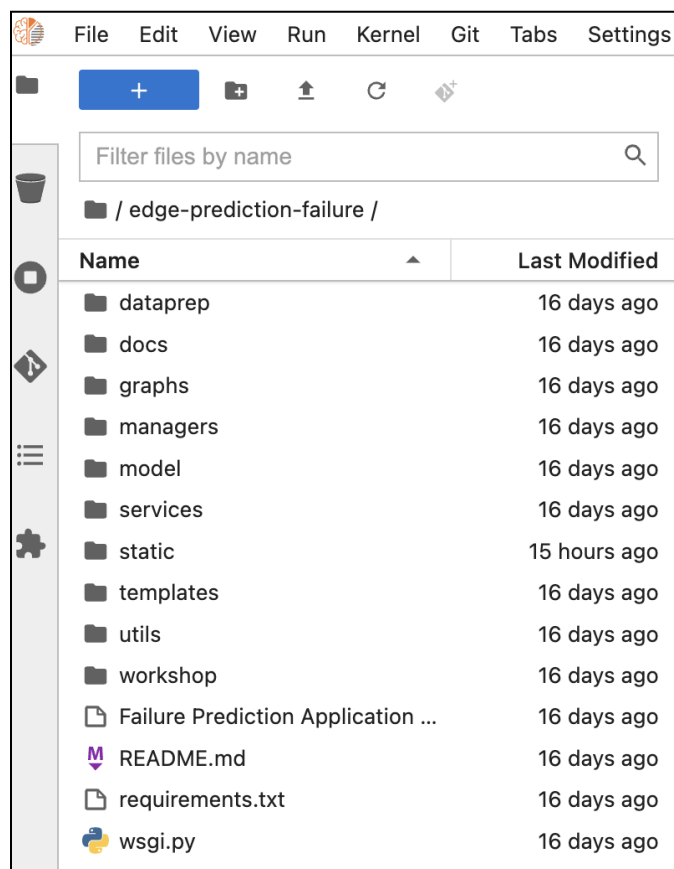
`https://github.com/Enterprise-Neurosystem/edge-prediction-failure`

Clone a repo

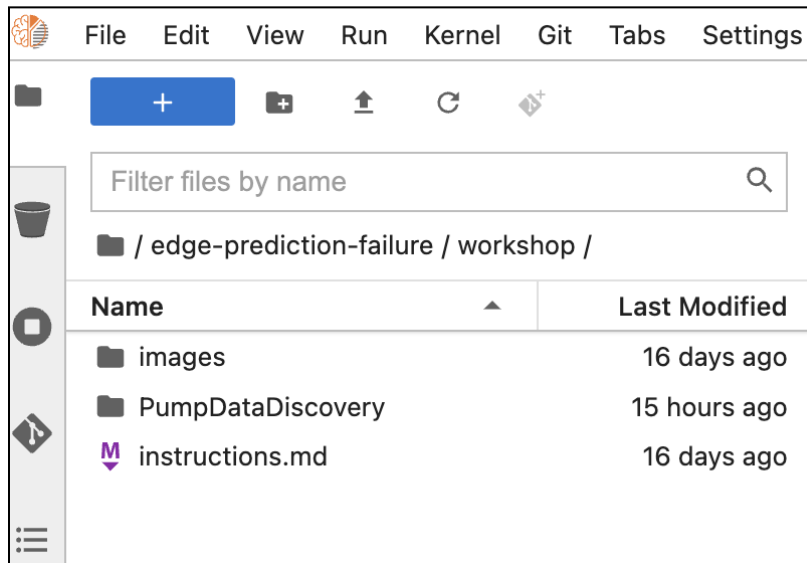
Enter the Clone URI of the repository

CancelCLONE

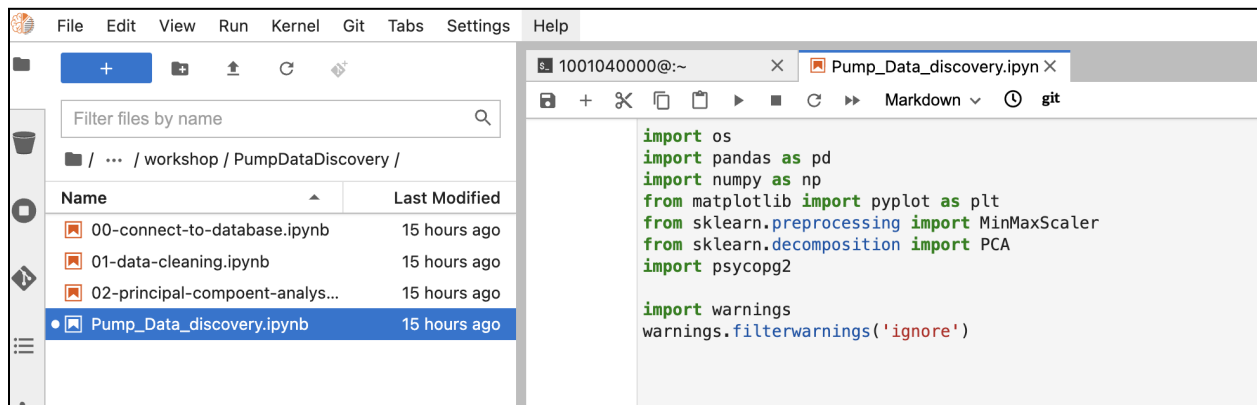
- It takes a few seconds, after which you can double-click and navigate to the newly-created folder, `edge-prediction-failure`:



Within the 'edge-failure-prediction' parent folder, find the 'workshop' folder and double click this folder to display its contents.

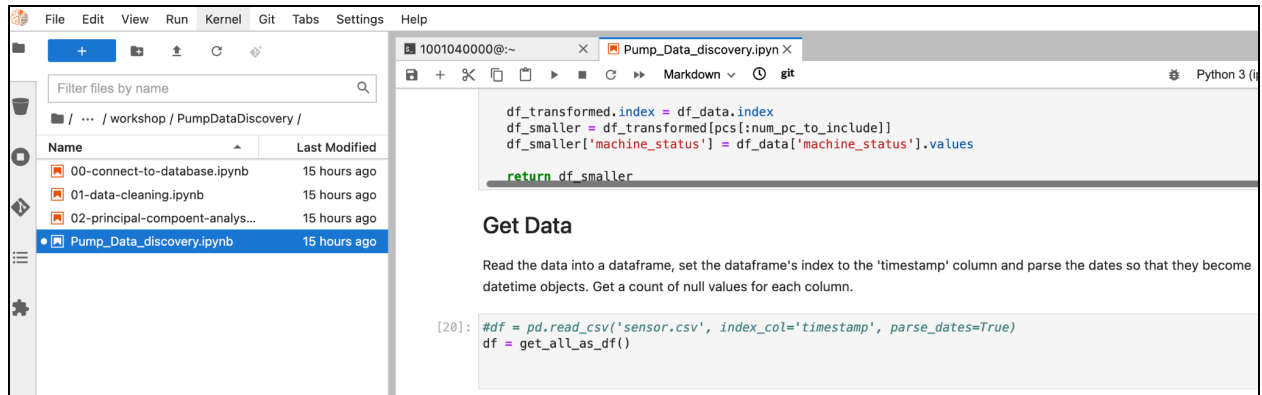


You will see an 'images' and 'PumpDataDiscovery' folder. Open the PumpDataDiscovery folder.



Double click the "Pump_Data_discovery.ipynb" jupyter notebook to open it. Within this notebook we will walk through the steps we used to prepare our pump data for usage with the failure prediction model. We have created a number of functions to help you prepare the data. Go ahead and execute (shift/enter) each function cell to bring the functions into memory.

We will then start looking at each step of the Data Preparing process. The first step will be Getting the Data :)

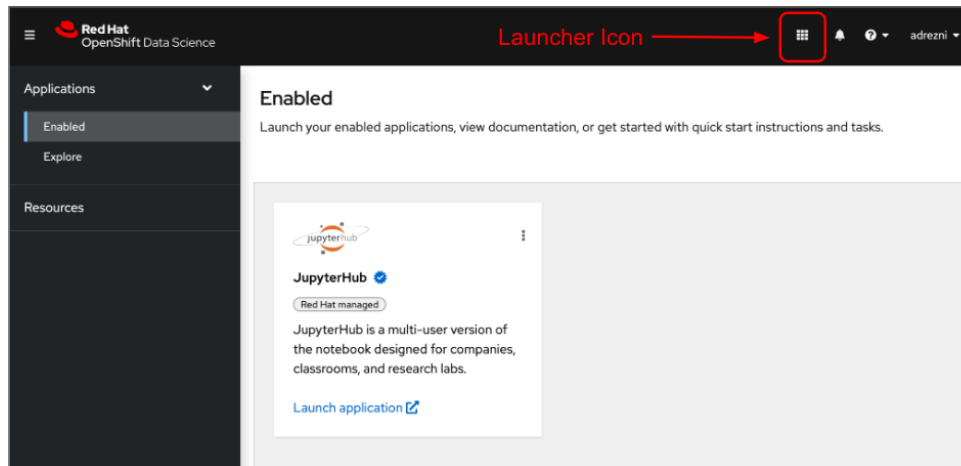


You will execute each cell of the data preparation process and note how the data is cleaned and prepared. This is a great deal of work!

When you are done with the data preparation process, you can go to the next step which is deploying and executing the Failure Prediction application.

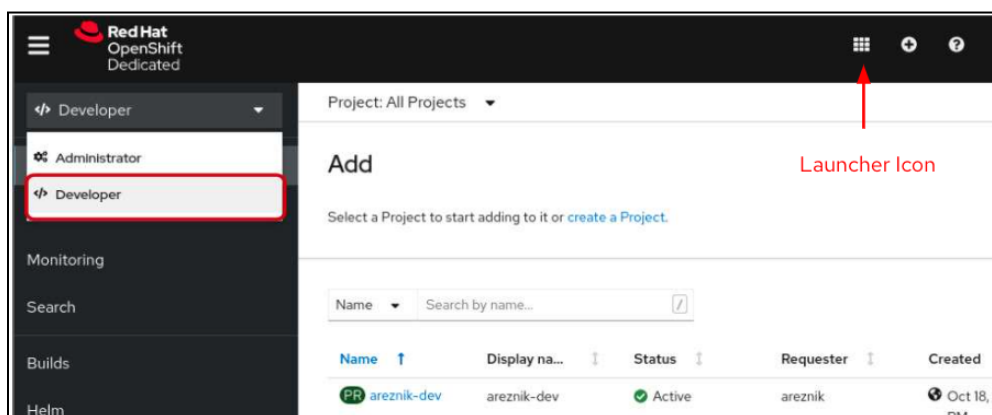
III. Deploy the Failure Detection web application as a container

16. From the Red Hat OpenShift Data Science platform choose the Launcher Tool and Select 'OpenShift Console'

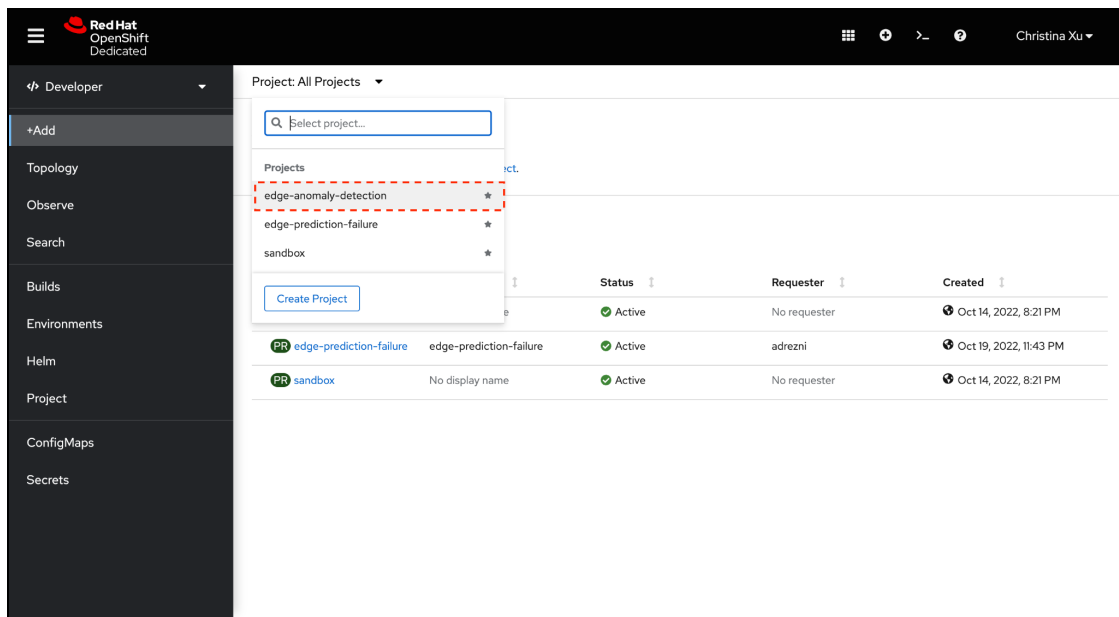


17. Within the OpenShift console, make certain you are in 'Developer' mode.

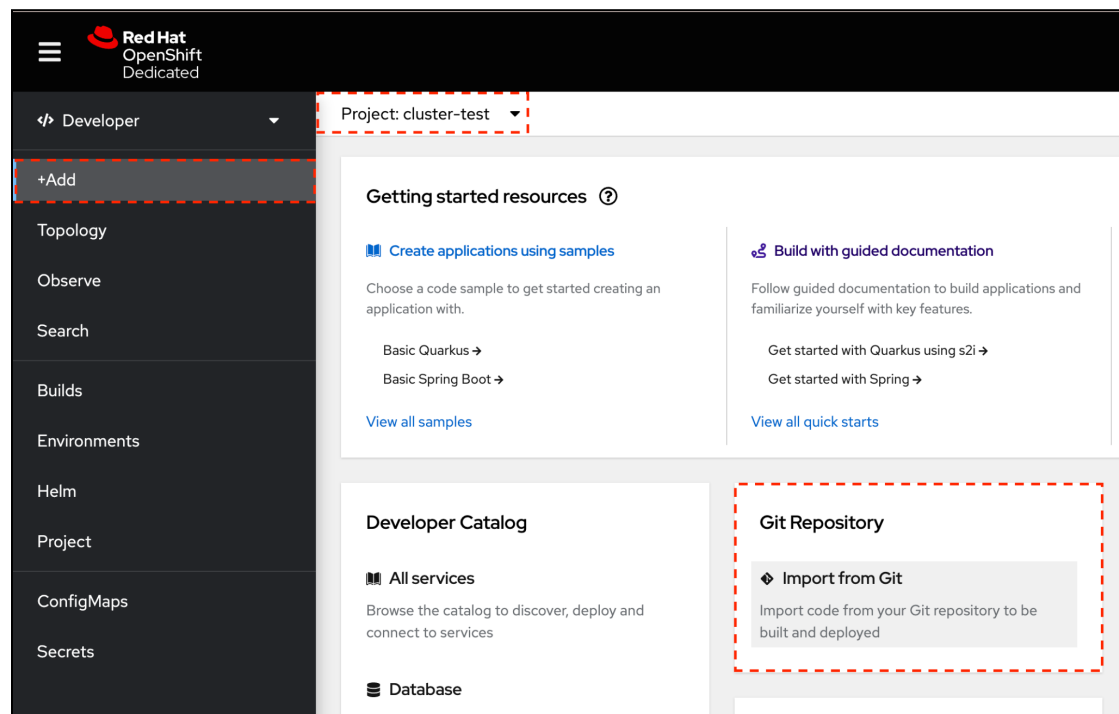
Note: The Launcher Icon is also visible in the OpenShift console. You can use this icon to move between the Red Hat OpenShift Data Science Platform and the Red Hat OpenShift Dedicated platform.



18. In the Project drop-down menu, select 'edge-anomaly-detection'. The 'edge-anomaly-detection' project is your work area within OpenShift. Make sure you are in it before proceeding to the following steps



19. Once you are in the edge-anomaly-detection project, you can begin to create your container. From the +Add menu, click the 'From Git' option.



20. In the Git Repo URL field, enter:

<https://github.com/Enterprise-Neurosystem/edge-prediction-failure>

Import from Git

Git


Git Repo URL *

https://github.com/Enterprise-Neurosystem/edge-prediction-failure

Validated

21. Next, change the BUILDER PYTHON to Python 3.8 (UBI7). Click 'Edit Import Strategy', then select 3.8 - ubi7 from the drop down list.

✓ **Builder Image detected.**
A Builder Image is recommended.

 **Python 3.9 (UBI 8)**
BUILDER PYTHON

Build and run Python 3.9 applications on UBI 8. For more considerations, see <https://github.com/sclorg/s2i-python>.
Sample repository: <https://github.com/sclorg/django-ex.g>

Click Edit Import Strategy

Edit Import Strategy

✓ **Builder Image detected.**
A Builder Image is recommended.

Import Strategy

Devfile

Dockerfile

Revert to recommended
Builder Image

Builder Image

Perl

PHP

NGINX
Nginx

Httpd

.NET

Go

nede
Node.js

Builder image version *

3.8-ubi8

3.9-ubi8

3.8-ubi7

3.8-ubi8

3.6-ubi8

Select 3.8 - ubi7 from the drop down list

22. If you continue to scroll down the page, you will see that everything is automatically selected to create a deployment of your application, as well as a route through which you will be able to access the application.

Make certain to name your app. For example, edge-failure-prediction

General

Application name

edge-prediction-failure-app


A unique name given to the Application grouping to label your resources.

Name *

edge-prediction-failure

A unique name given to the component that will be used to name associated resources.

Now we are ready to press the 'Create' button to create our containerized application. Press the 'Create' button.

 **Python 3.8 (UBI 7)**
BUILDER PYTHON

Build and run Python 3.8 applications on UBI 7. For more information about using this builder image, including OpenShift considerations, see <https://github.com/sclorg/s2i-python-container/blob/master/3.8/README.md>.
Sample repository: <https://github.com/sclorg/django-ex.git>

General

Application name

edge-prediction-failure-app

A unique name given to the Application grouping to label your resources.

Name *

edge-prediction-failure

A unique name given to the component that will be used to name associated resources.

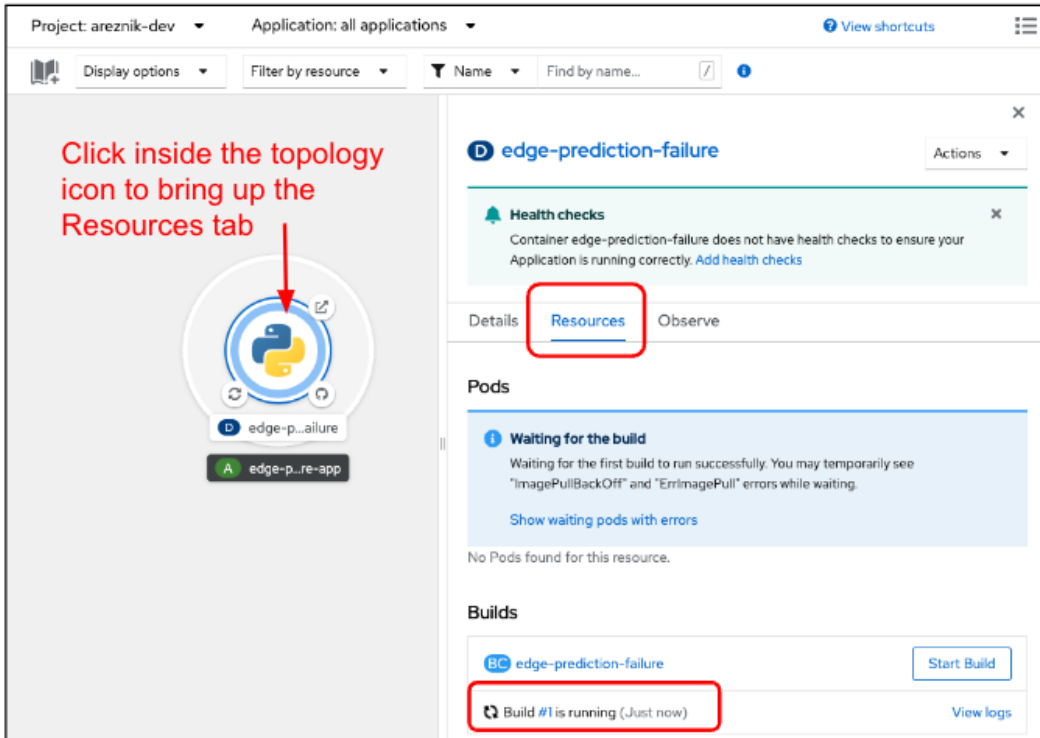
Resources

Select the resource type to generate

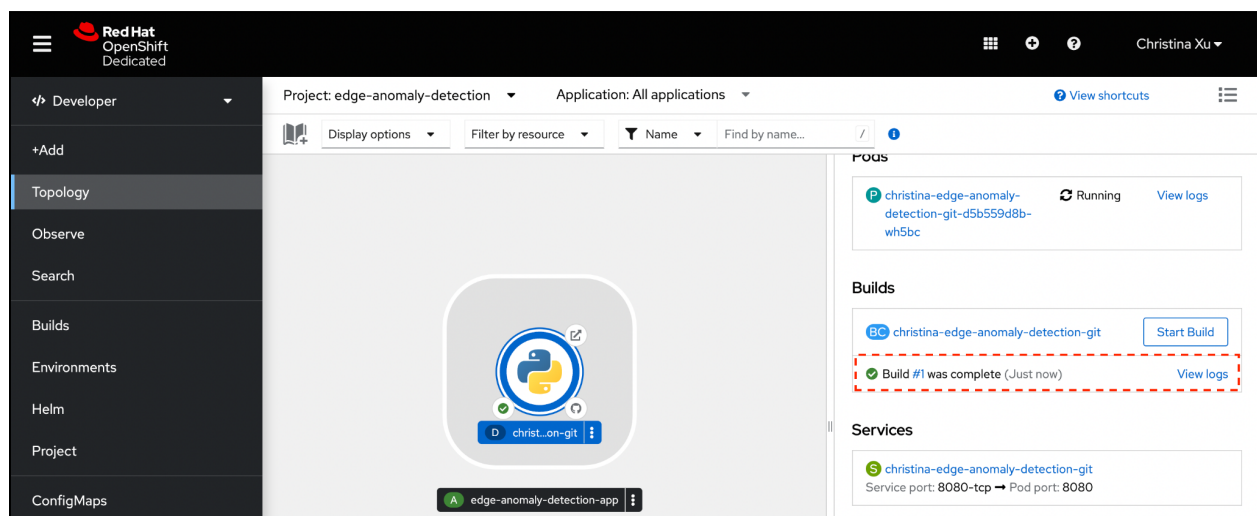
☒ **Deployment**
apps/Deployment
A Deployment enables declarative updates for Pods and ReplicaSets.

☐ **DeploymentConfig**
apps.openshift.io/DeploymentConfig
A DeploymentConfig defines the template for a Pod and manages deploying new images or configuration changes.

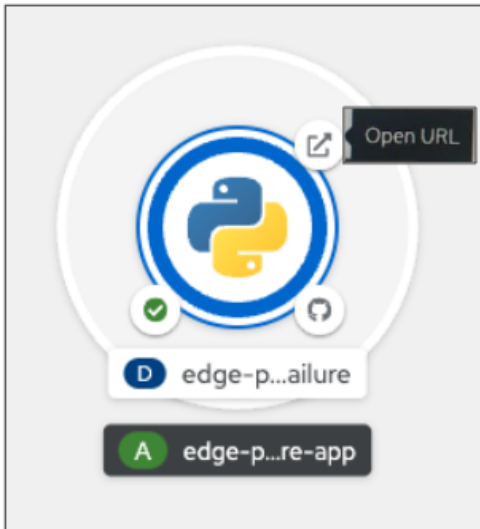
The automated process will take a few minutes. You can view the progress of your build by clicking on your application and then clicking on 'Resources' in the Topology view.



23. You will know that your build is complete once the following message under Builds:



24. You can display your containerized application in a browser by clicking on the URL icon in the Topology view.



25. Your containerized Failure Prediction application will now appear in a browser window.

