Red Hat OpenShift AI Self-Managed 2.6

OpenShift AI tutorial - Fraud detection example

Last Updated: 2024-02-26

Legal Notice

Abstract

Step-by-step guidance for using OpenShift AI to develop and train an example model in Jupyter Notebooks, deploy the model, integrate the model into a fraud detection application, and refine the model by using automated pipelines.

Table of Contents

CHAPTER 1. INTRODUCTION 3
1.1. ABOUT THE EXAMPLE FRAUD DETECTION MODEL 3
1.2. BEFORE YOU BEGIN 3

CHAPTER 2. SETTING UP A PROJECT AND STORAGE 4
2.1. NAVIGATING TO THE OPENSHIFT AI DASHBOARD 4
2.2. SETTING UP YOUR DATA SCIENCE PROJECT 5
2.3. STORING DATA WITH DATA CONNECTIONS 7
2.3.1. Creating data connections to your own S3-compatible object storage 8
2.3.2. Running a script to install local object storage buckets and create data connections 11
2.4. ENABLING DATA SCIENCE PIPELINES 14

CHAPTER 3. CREATING A WORKBENCH AND A NOTEBOOK 17
3.1. CREATING A WORKBENCH AND SELECTING A NOTEBOOK IMAGE 17
3.2. IMPORTING THE TUTORIAL FILES INTO THE JUPYTER ENVIRONMENT 20
3.3. RUNNING CODE IN A NOTEBOOK 24
3.3.1. Try it 25
3.4. TRAINING A MODEL 26

CHAPTER 4. DEPLOYING AND TESTING A MODEL 28
4.1. PREPARING A MODEL FOR DEPLOYMENT 28
4.2. DEPLOYING A MODEL 29
4.3. TESTING THE MODEL API 32

CHAPTER 5. IMPLEMENTING PIPELINES 34
5.1. AUTOMATING WORKFLOWS WITH DATA SCIENCE PIPELINES 34
5.1.1. Create a pipeline 34
5.1.2. Add nodes to your pipeline 36
5.1.3. Specify the training file as a dependency 37
5.1.4. Create and store the ONNX-formatted output file 38
5.1.5. Configure the data connection to the S3 storage bucket 39
5.1.6. Run the Pipeline 42
5.2. RUNNING A DATA SCIENCE PIPELINE GENERATED FROM PYTHON CODE 44

CHAPTER 6. CONCLUSION 47

CHAPTER 1. INTRODUCTION

Welcome!

In this tutorial, you'll learn how to incorporate data science and artificial intelligence and machine learning (AI/ML) into an OpenShift development workflow.

You'll use an example fraud detection model to complete the following tasks:

- Explore a pre-trained fraud detection model by using Jupyter Notebooks.
- Deploy the model by using OpenShift AI model serving.
- Integrate the model into a real-time fraud detection application.
- Refine and train the model by using automated pipelines.

And you'll do all of this without having to install anything on your own computer, thanks to Red Hat OpenShift AI.

1.1. ABOUT THE EXAMPLE FRAUD DETECTION MODEL

The example fraud detection model mentors credit card transactions for potential fraudulent activity. It analyzes the following credit card transaction details:

- The geographical distance from the previous credit card transaction.
- The price of the current transaction, compared to the median price of all the user's transactions.
- Whether the user completed the transaction by using the hardware chip in the credit card, entered a PIN number, or for an online purchase.

Based on this data, the model outputs the likelihood of the transaction being fraudulent.

1.2. BEFORE YOU BEGIN

Set up your Red Hat OpenShift AI environment

If you don't already have an instance of Red Hat OpenShift AI, find out more on the developer page. There, you can spin up your own account on the **free OpenShift Data Science Sandbox**or learn about installing on **your own OpenShift cluster**.

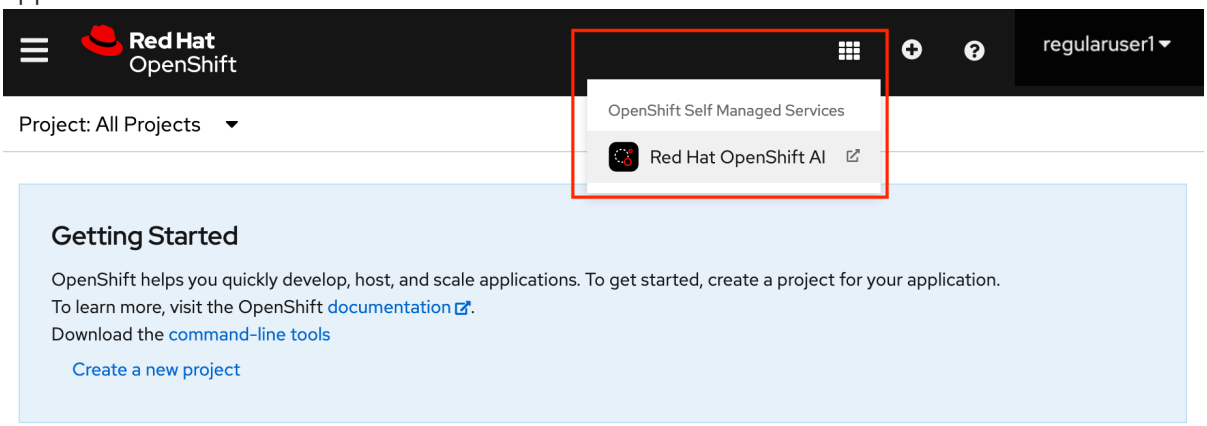
If you're ready, start the tutorial!

CHAPTER 2. SETTING UP A PROJECT AND STORAGE

2.1. NAVIGATING TO THE OPENSHIFT AI DASHBOARD

Procedure

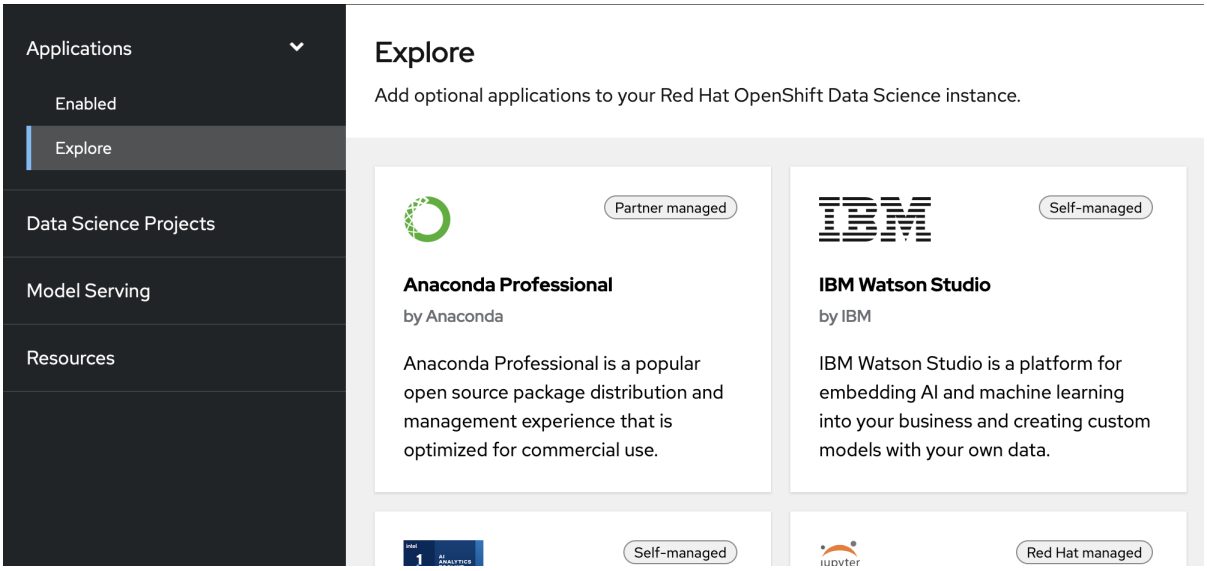
1. After you log in to the OpenShift console, access the OpenShift AI dashboard by clicking the application launcher icon on the header.



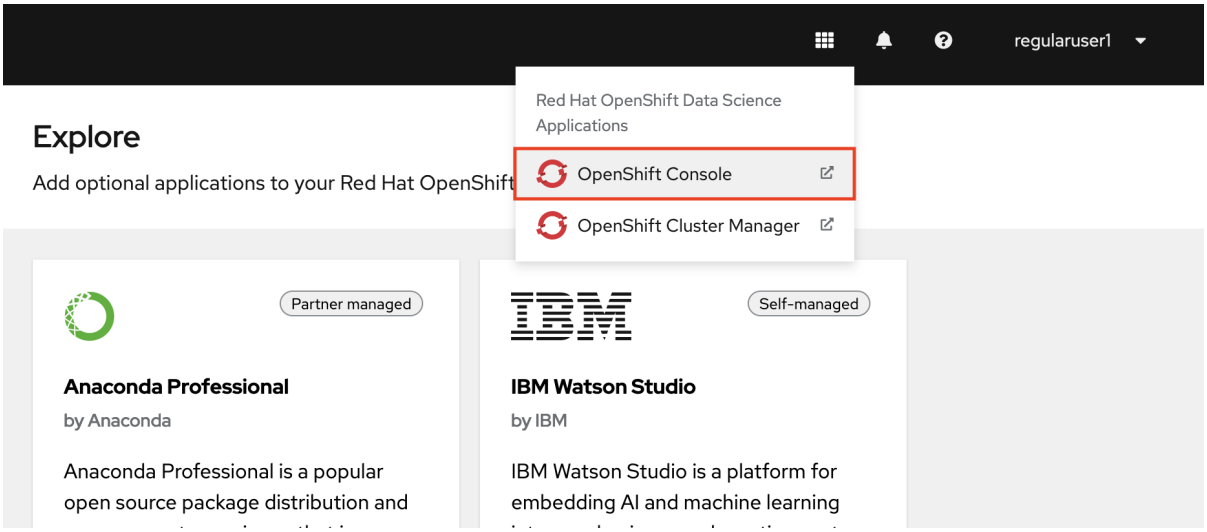
2. When prompted, log in to the OpenShift AI dashboard by using your OpenShift credentials. OpenShift AI uses the same credentials as OpenShift for the dashboard, notebooks, and all other components.

The OpenShift AI dashboard shows the status of any installed and enabled applications.

3. Optionally, click **Explore** to view other available application integrations.



Note: You can navigate back to the OpenShift console in a similar fashion. Click the application launcher to access the OpenShift console.



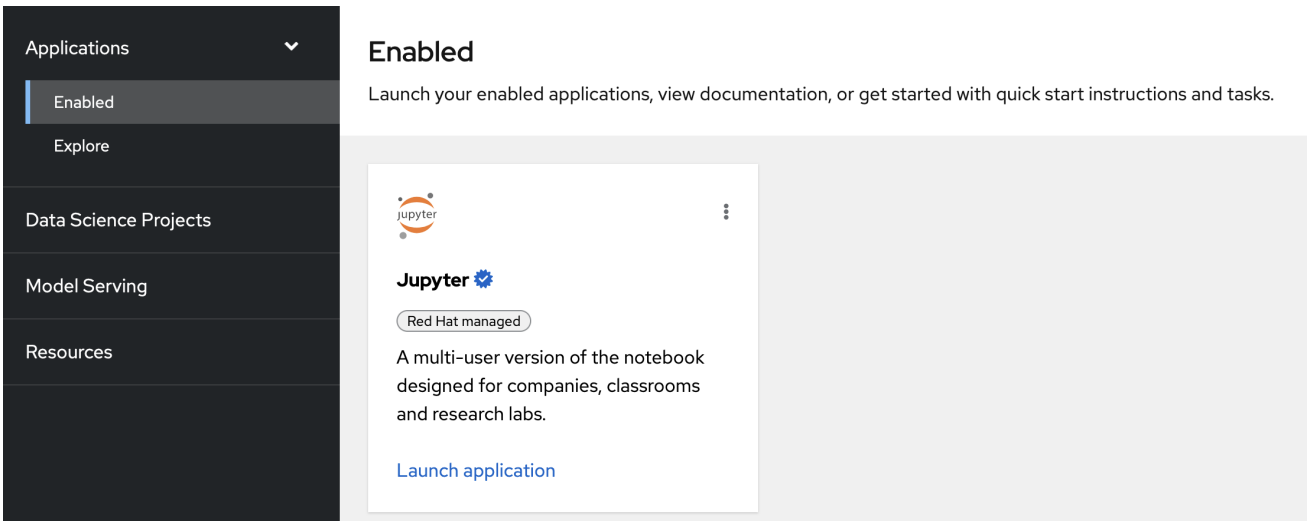
For now, stay in the OpenShift AI dashboard.

Next step

Setting up your data science project

2.2. SETTING UP YOUR DATA SCIENCE PROJECT

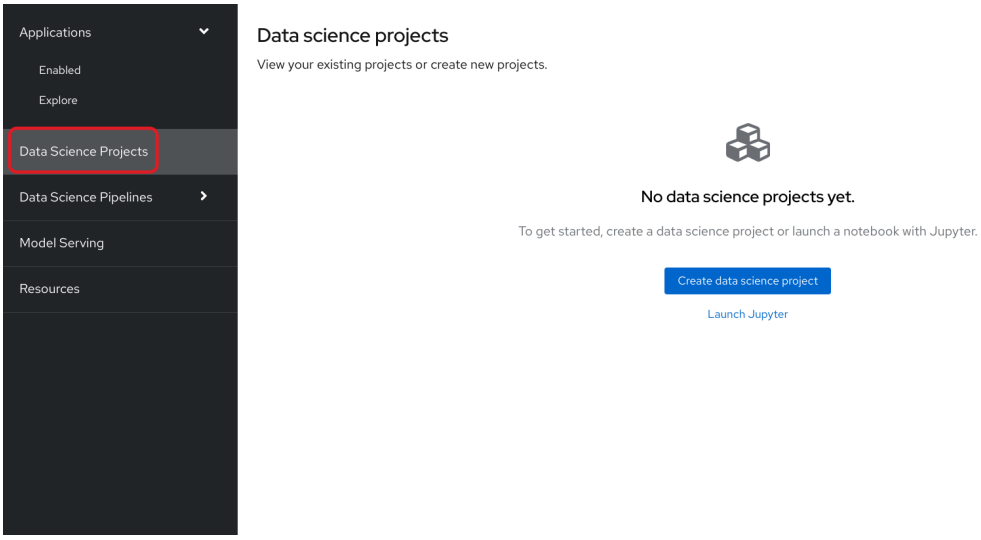
Before you begin, make sure that you are logged in to **Red Hat OpenShift AI** and that you can see the dashboard:



Note that you can start a Jupyter notebook from here, but it would be a one-off notebook run in isolation. To implement a data science workflow, you must create a data science project. Projects allow you and your team to organize and collaborate on resources within separated namespaces. From a project you can create multiple workbenches, each with their own Jupyter notebook environment, and each with their own data connections and cluster storage. In addition, the workbenches can share models and data with pipelines and model servers.

Procedure

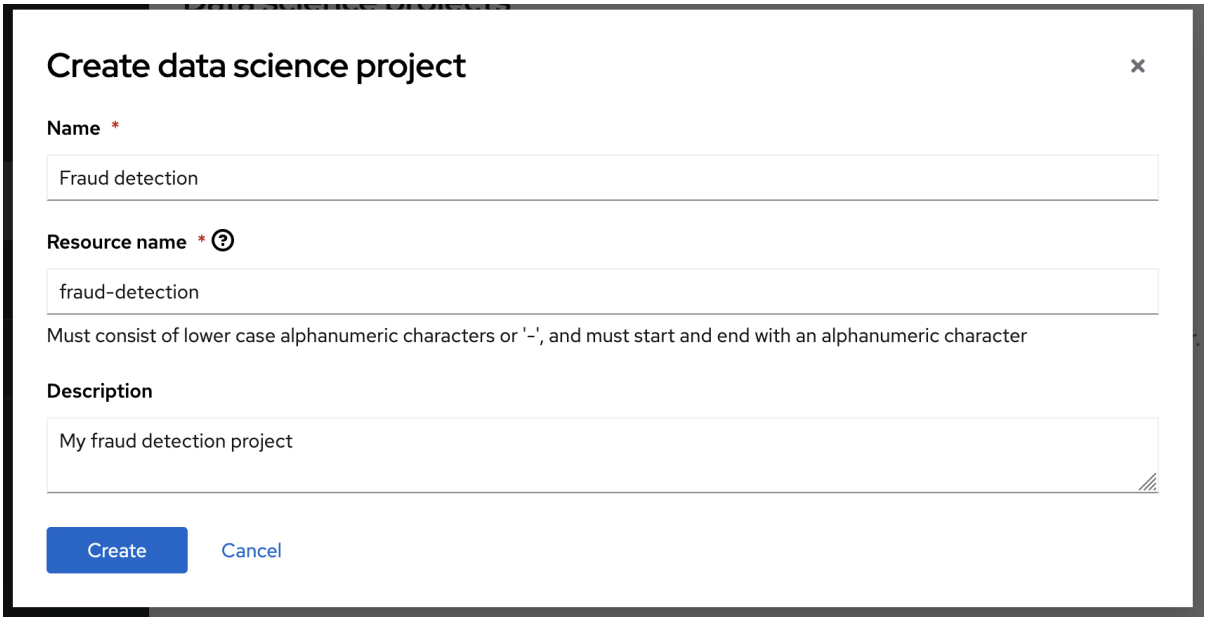
1. On the navigation menu, select **Data Science Projects**. This page lists any existing projects that you have access to. From this page, you can select an existing project (if any) or create a new one.



   If you already have an active project that you'd like to use, select it now and skip ahead to the next section, Storing data with data connections. Otherwise, continue to the next step.

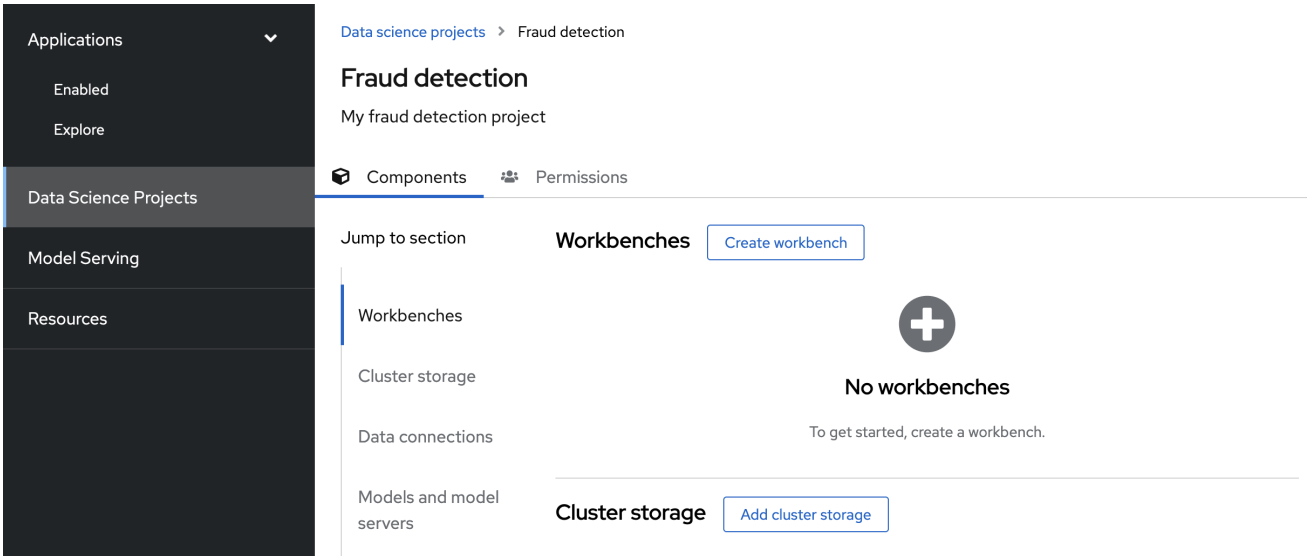
2. Click **Create data science project**

3. Enter a display name and description. Based on the display name, a resource name is automatically generated, but you can change it if you'd like.



Verification

You can now see its initial state. There are five types of project components:



- **Workbenches** are instances of your development and experimentation environment. They typically contain IDEs, such as JupyterLab, RStudio, and Visual Studio Code.
- A **Cluster storage** is a volume that persists the files and data you're working on within a workbench. A workbench has access to one or more cluster storage instances.
- **Data connections** contain configuration parameters that are required to connect to a data source, such as an S3 object bucket.
- **Pipelines** contain the Data Science pipelines that are executed within the project.
- **Models and model servers** allow you to quickly serve a trained model for real-time inference. You can have multiple model servers per data science project. One model server can host multiple models.

Next step

Storing data with data connections

2.3. STORING DATA WITH DATA CONNECTIONS

For this tutorial, you need two S3-compatible object storage buckets, such as Ceph, Minio, or AWS S3:

- **My Storage** - Use this bucket for storing your models and data. You can reuse this bucket and its connection for your notebooks and model servers.
- **Pipelines Artifacts** - Use this bucket as storage for your pipeline artifacts. A pipeline artifacts bucket is required when you create a pipeline server. For this tutorial, create this bucket to separate it from the first storage bucket for clarity.

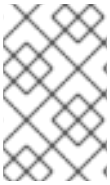
You can use your own storage buckets or run a provided script that creates local Minio storage buckets for you.

Also, you must create a data connection to each storage bucket. A data connection is a resource that contains the configuration parameters needed to connect to an object storage bucket.

You have two options for this tutorial, depending on whether you want to use your own storage buckets or use a script to create local Minio storage buckets:

- If you want to use your own S3-compatible object storage buckets, create data connections to them as described in Creating data connections to your own S3-compatible object storage .
- If you want to run a script that installs local Minio storage buckets and creates data connections to them, for the purposes of this tutorial, follow the steps in Running a script to install local object storage buckets and create data connections.

2.3.1. Creating data connections to your own S3-compatible object storage



NOTE

If you do not have your own s3-compatible storage, or if you want to use a disposable local Minio instance instead, skip this section and follow the steps in Running a script to install local object storage buckets and create data connections.

Prerequisite

To create data connections to your existing S3-compatible storage buckets, you need the following credential information for the storage buckets:

- Endpoint URL
- Access key
- Secret key
- Region
- Bucket name

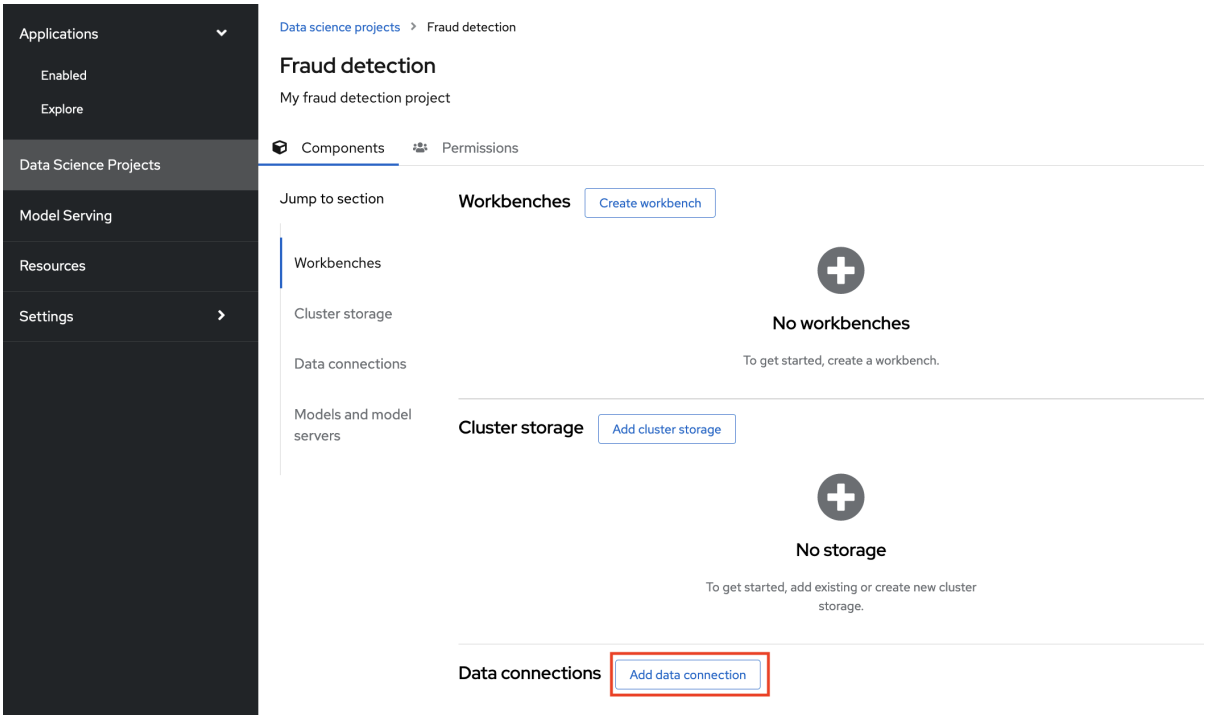
If you don't have this information, contact your storage administrator.

Procedures

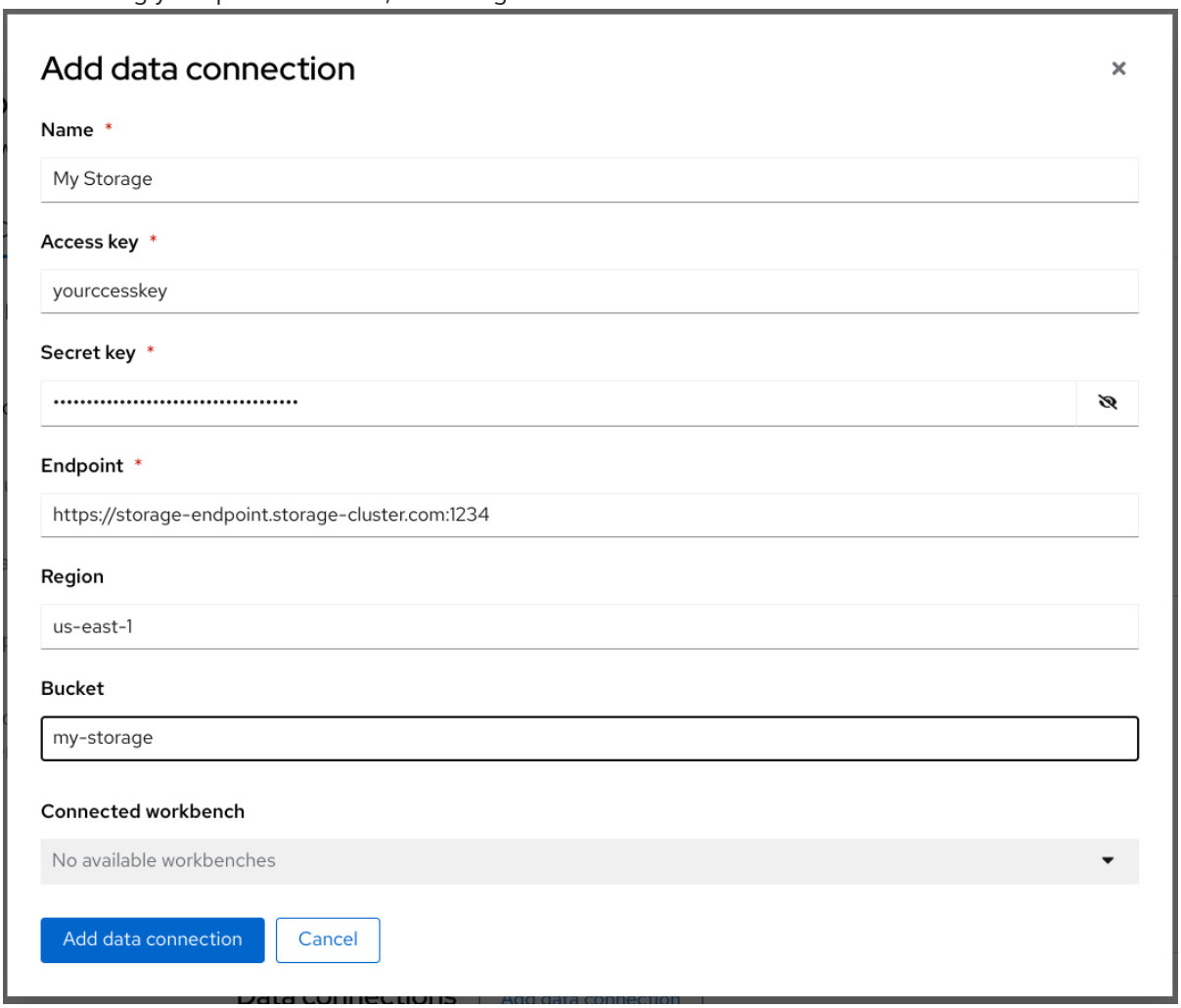
Create data connections to your two storage buckets.

Create a data connection for saving your data and models

1. In the OpenShift AI dashboard, navigate to the page for your data science project.
2. Under **Components**, click **Data connections**.
3. Click **Add data connection**



4. Fill out the **Add data connection** form and name your connection **My Storage**. This connection is for saving your personal work, including data and models.



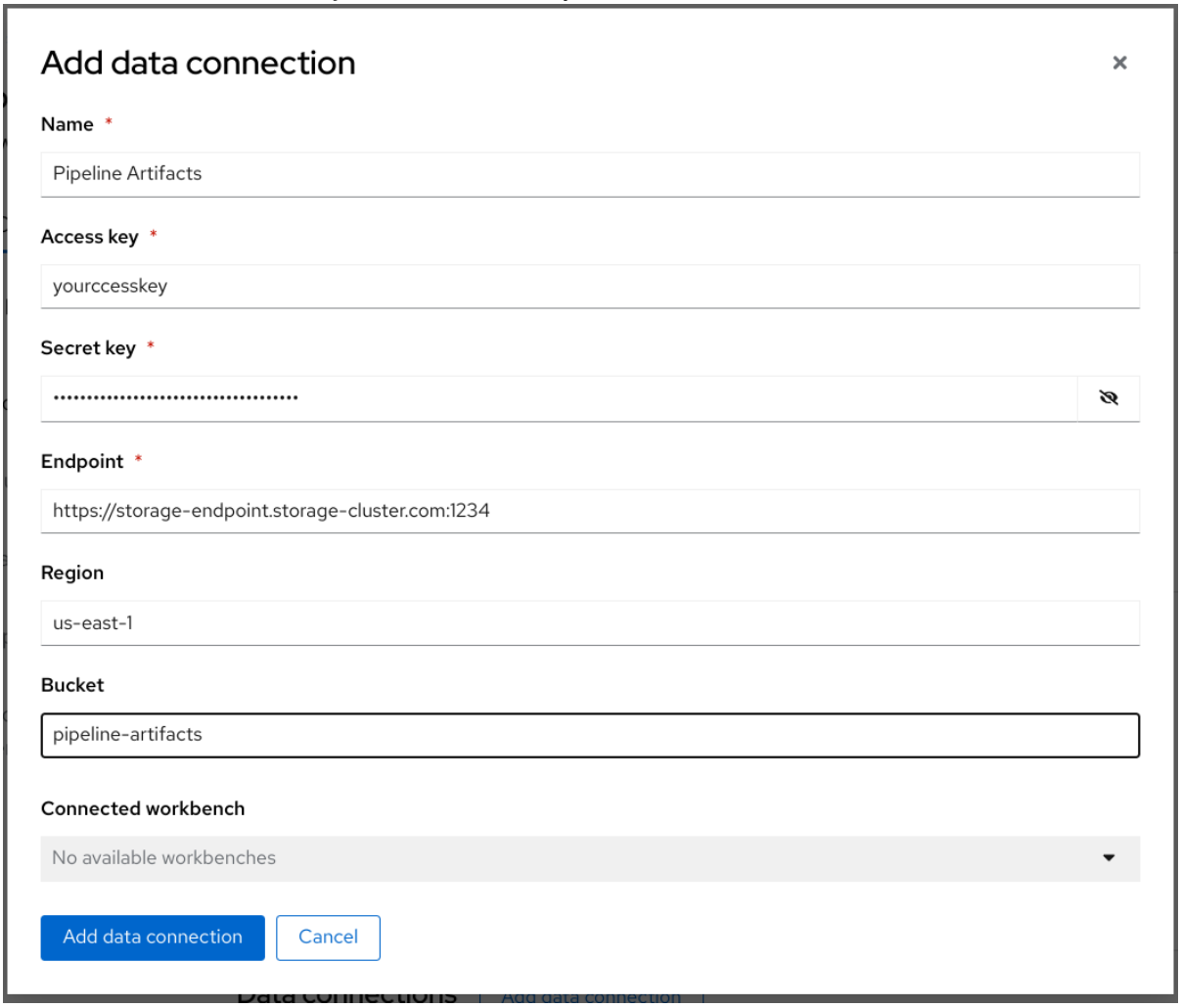
5. Click **Add data connection**.

Create a data connection for saving pipeline artifacts

NOTE

If you do not intend to complete the pipelines section of the tutorial, you can skip this step.

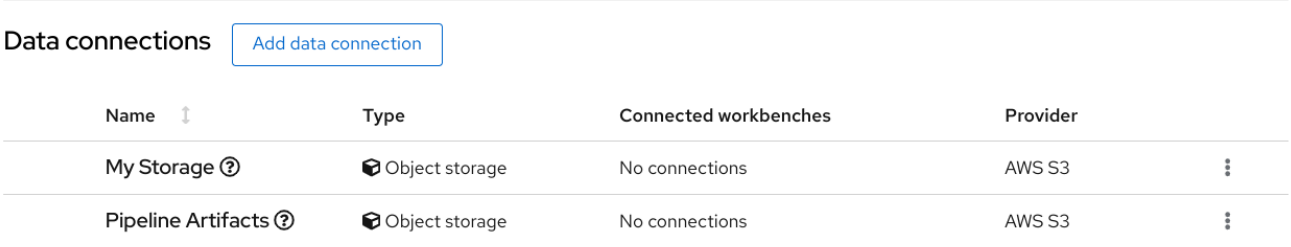
1. Click **Add data connection**.
2. Fill out the form and name your connection **Pipeline Artifacts**.



3. Click **Add data connection**.

Verification

Check to see that your data connections are listed in the project.



Next step

Creating a workbench

2.3.2. Running a script to install local object storage buckets and create data connections

For convenience, the provided script installs two data connections (and associated secrets) and two Minio buckets as s3-compatible storage The script creates a random user and password for security. This script is based on the instructions for installing Minio in this guide.

IMPORTANT

The storage buckets that this script creates are **not** meant for production usage.

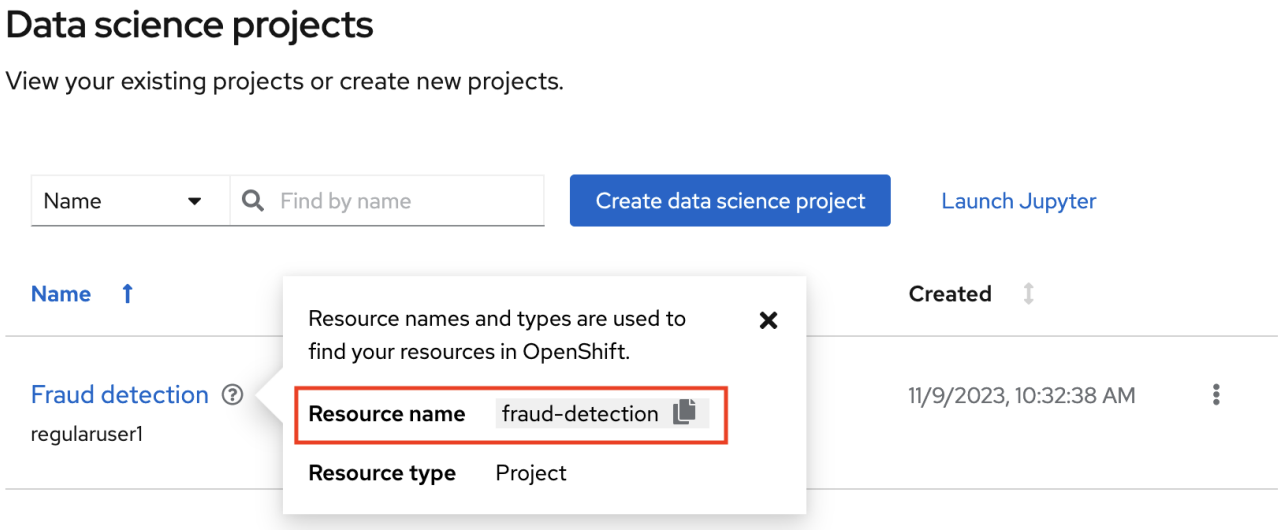
NOTE

If you want to connect to your own storage, see Creating data connections to your own S3-compatible object storage.

Prerequisite

You must know the OpenShift resource name for your data science project so that you run the provided script in the correct project. To get the project's resource name:

In the OpenShift AI dashboard, select **Data Science Projects** and then hover your cursor over the **?** icon next to the project name. A text box appears with information about the project, including it's resource name:



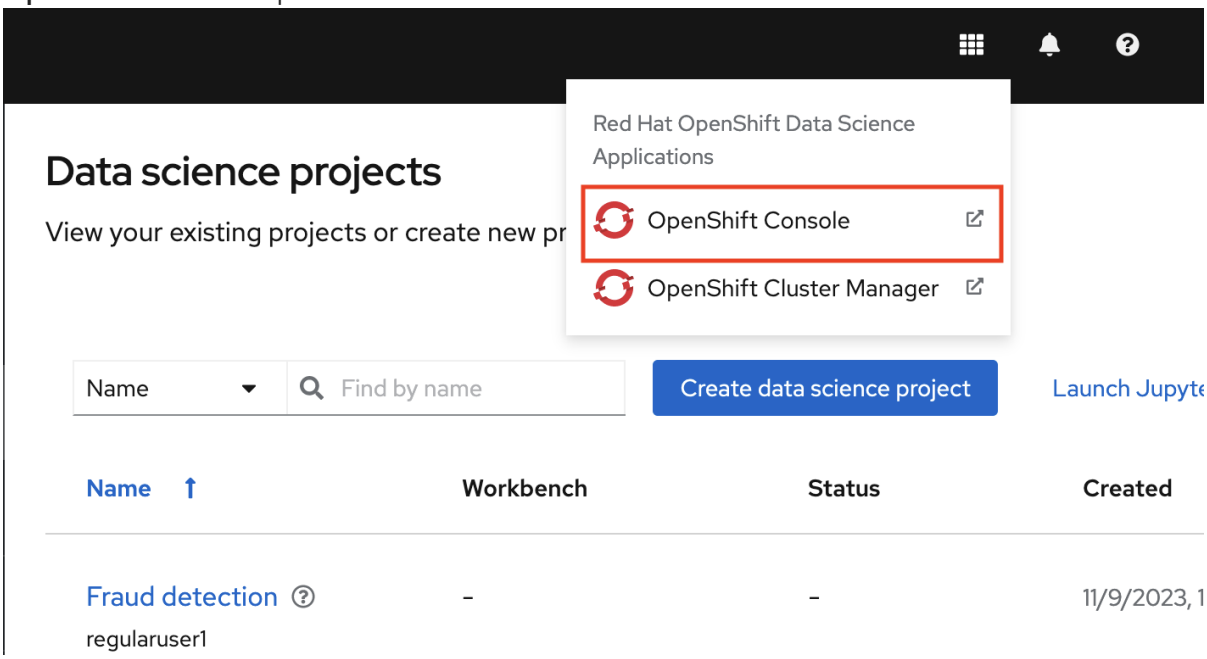
NOTE

The following procedure describes how to run the script from the OpenShift console. If you are knowledgeable in OpenShift and can access the cluster from the command line, instead of following the steps in this procedure, you can use the following command to run the script:

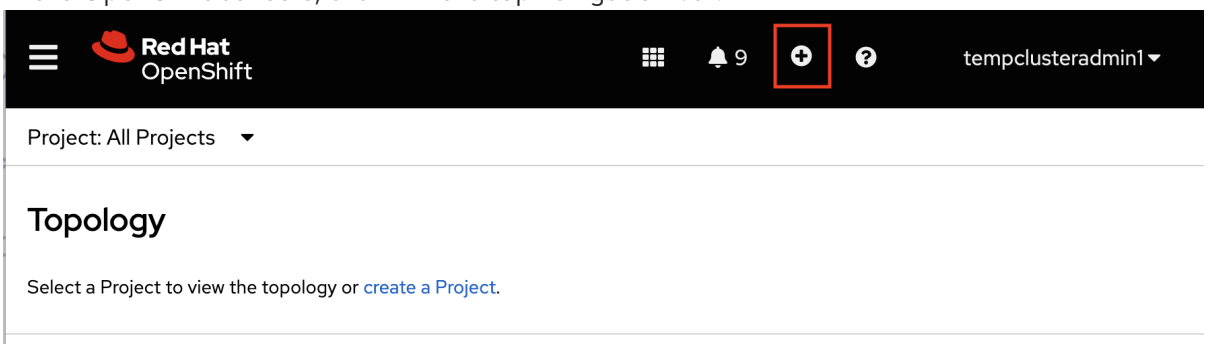
```
oc apply -n <your-project-name/> -f https://github.com/rh-aiservices-bu/fraud-detection/raw/main/setup/setup-s3.yaml
```

Procedure

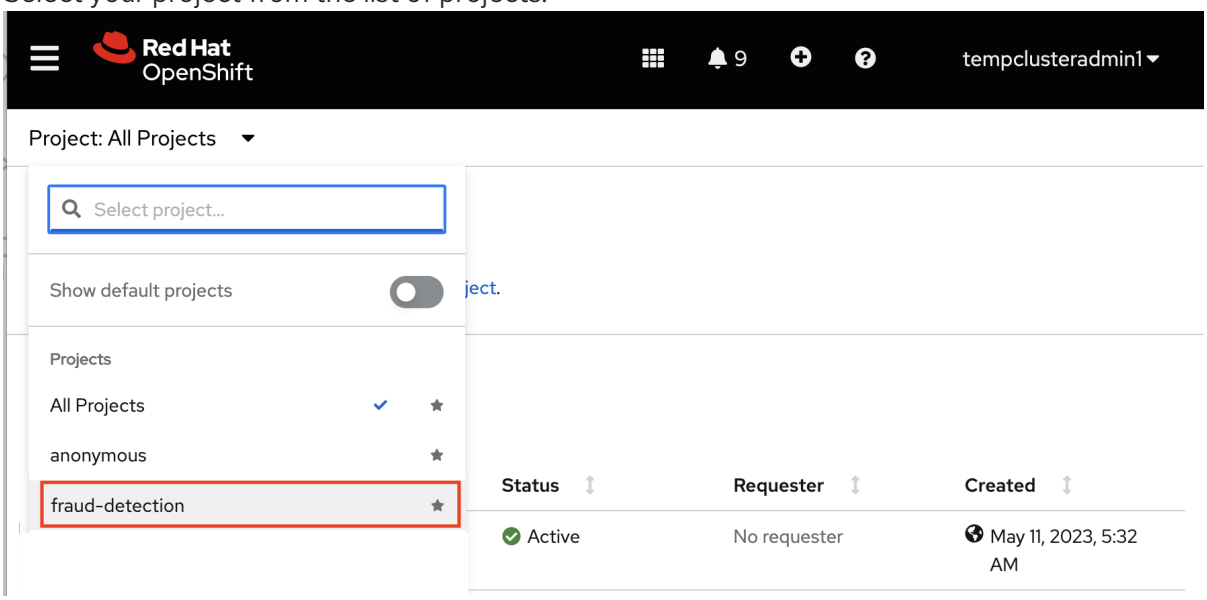
1. In the OpenShift AI dashboard, click the application launcher icon and then select the **OpenShift Console** option.



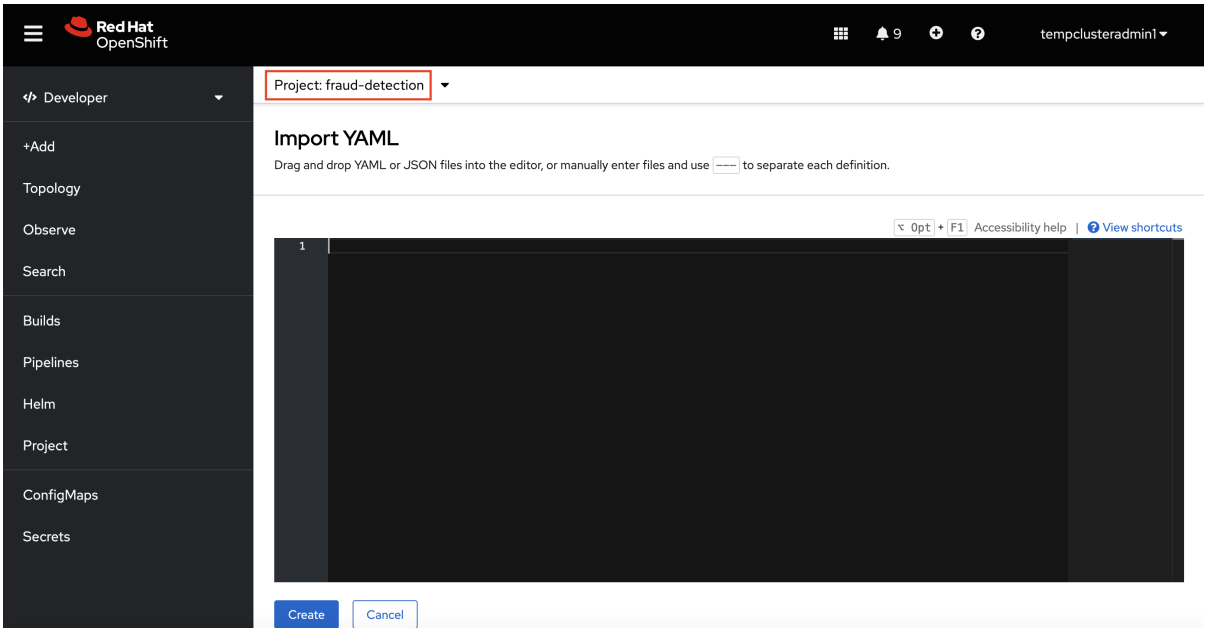
2. In the OpenShift console, click + in the top navigation bar.



3. Select your project from the list of projects.



4. Verify that you selected the correct project.



5. Copy the following code and paste it into the **Import YAML** editor.
Note: This code gets and applies the **setup-s3-no-sa.yaml** file.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: demo-setup
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: demo-setup-edit
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
  - kind: ServiceAccount
    name: demo-setup
---
apiVersion: batch/v1
kind: Job
metadata:
  name: create-s3-storage
spec:
  selector: {}
  template:
    spec:
      containers:
        - args:
            - -ec
            - |-
              echo -n 'Setting up Minio instance and data connections'
              oc apply -f https://github.com/rh-aiservices-bu/fraud-detection/raw/main/setup/setup-
s3-no-sa.yaml
          command:
```

```
      - /bin/bash
    image: image-registry.openshift-image-registry.svc:5000/openshift/tools:latest
    imagePullPolicy: IfNotPresent
    name: create-s3-storage
  restartPolicy: Never
  serviceAccount: demo-setup
  serviceAccountName: demo-setup
```

6. Click **Create**.

Verification

You should see a "Resources successfully created" message and the following resources listed:

- **demo-setup**
- **demo-setup-edit**
- **create s3-storage**

Next step

Creating a workbench

2.4. ENABLING DATA SCIENCE PIPELINES

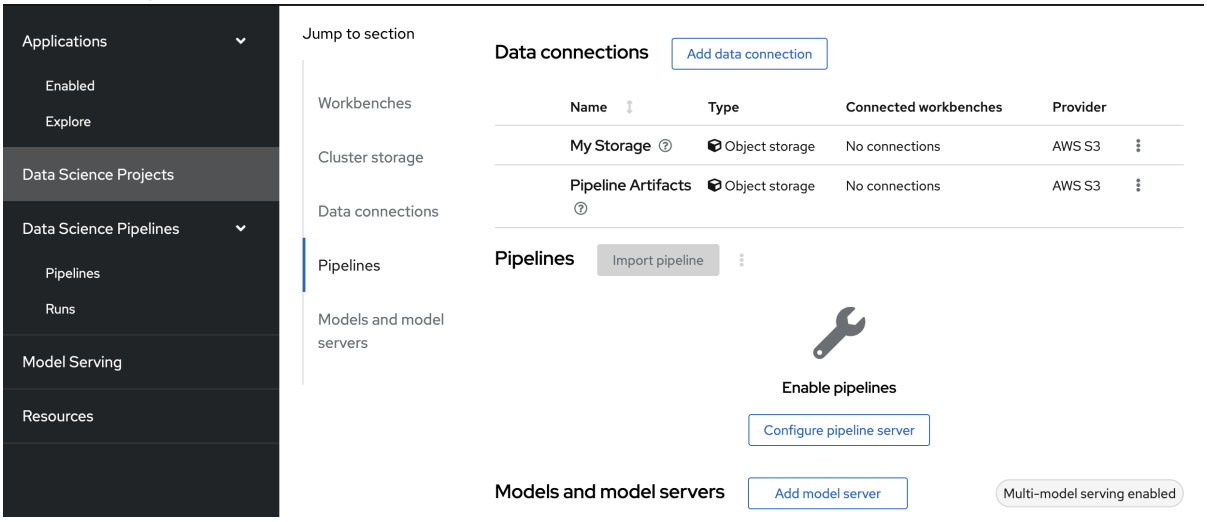
NOTE

If you do not intend to complete the pipelines section of the workshop you can skip this step and move on to the next section, Create a Workbench.

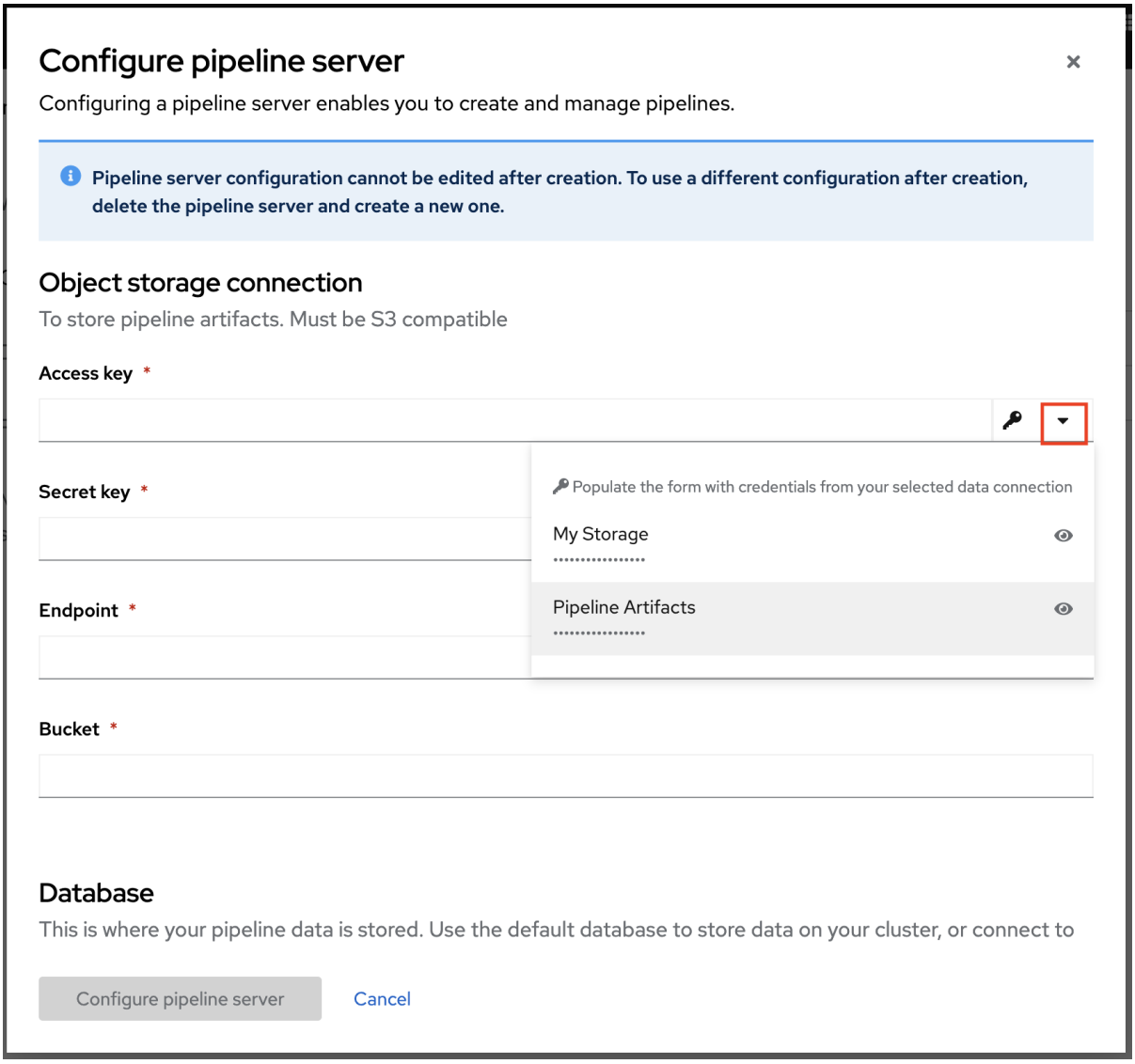
In this section, you prepare your tutorial environment so that you can use data science pipelines.

Procedure

1. In the OpenShift AI dashboard, click **Data Science Projects** and then select **Fraud Detection**.
2. Navigate to the **Pipelines** section.
3. Click **Configure pipeline server**.



4. In the **Configure pipeline server** form, in the **Access key** field next to the key icon, click the dropdown menu and then click **Pipeline Artifacts** to populate the **Configure pipeline server** form with credentials for the data connection.

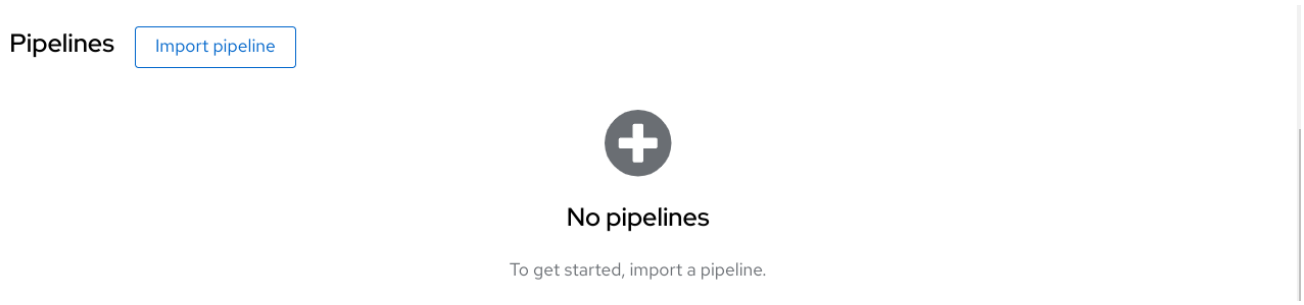


5. Leave the database configuration as the default.

6. Click **Configure**.

Verification

Check the **Pipelines** page. Pipelines are enabled when the **Pipeline server actions** option appears and the **Create pipeline server** button no longer appears.



Next step

Automating workflows with data science pipelines

Running a data science pipeline generated from Python code

CHAPTER 3. CREATING A WORKBENCH AND A NOTEBOOK

3.1. CREATING A WORKBENCH AND SELECTING A NOTEBOOK IMAGE

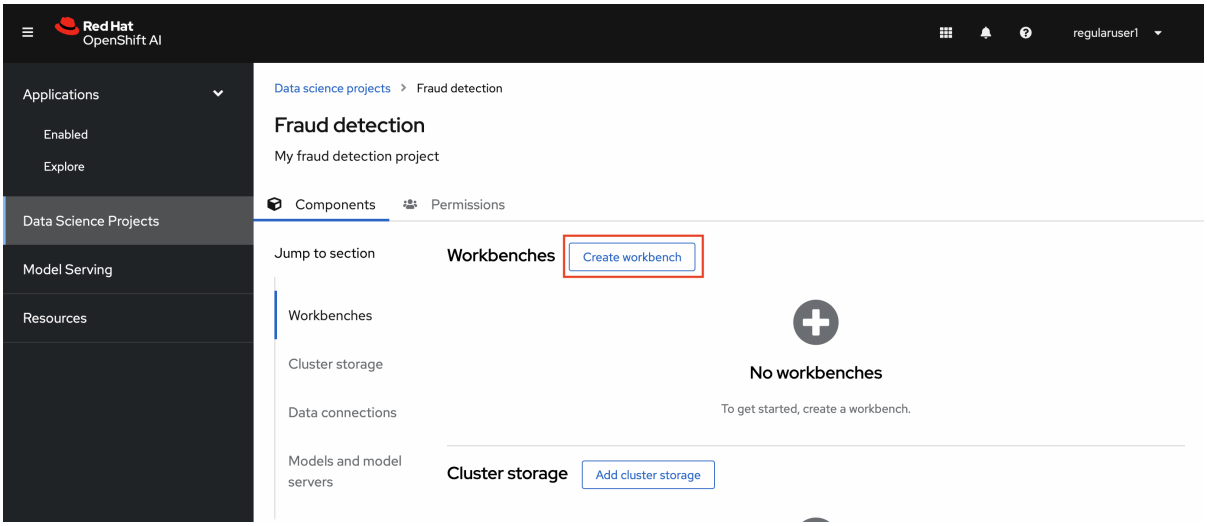
A workbench is an instance of your development and experimentation environment. Within the workbench you can select a notebook image for your data science work.

Prerequisite

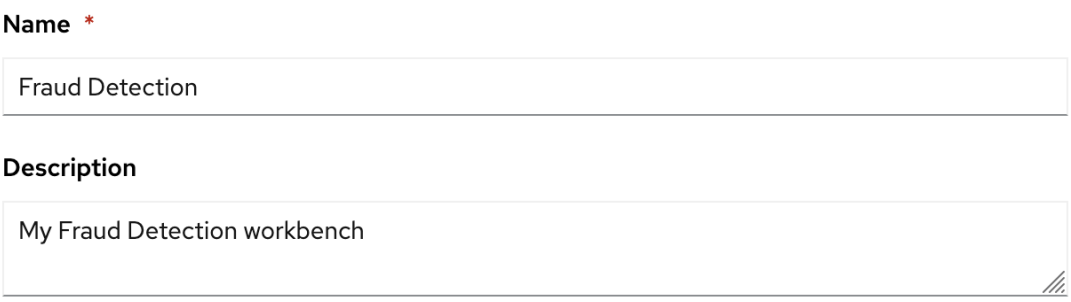
- You created a **My Storage** data connection as described in Storing data with data connections.

Procedure

1. Navigate to the project detail page for the data science project that you created in Setting up your data science project.
2. Click the **Create workbench** button.

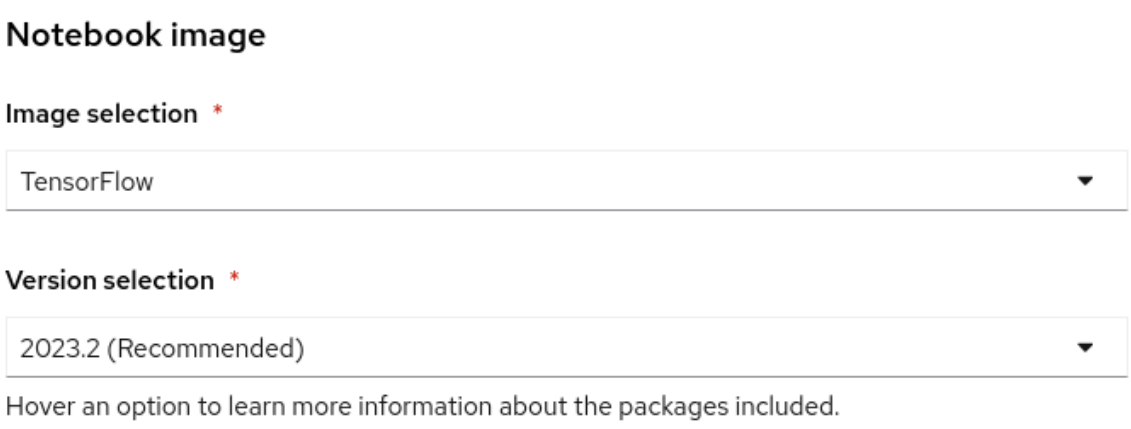


3. Fill out the name and description.



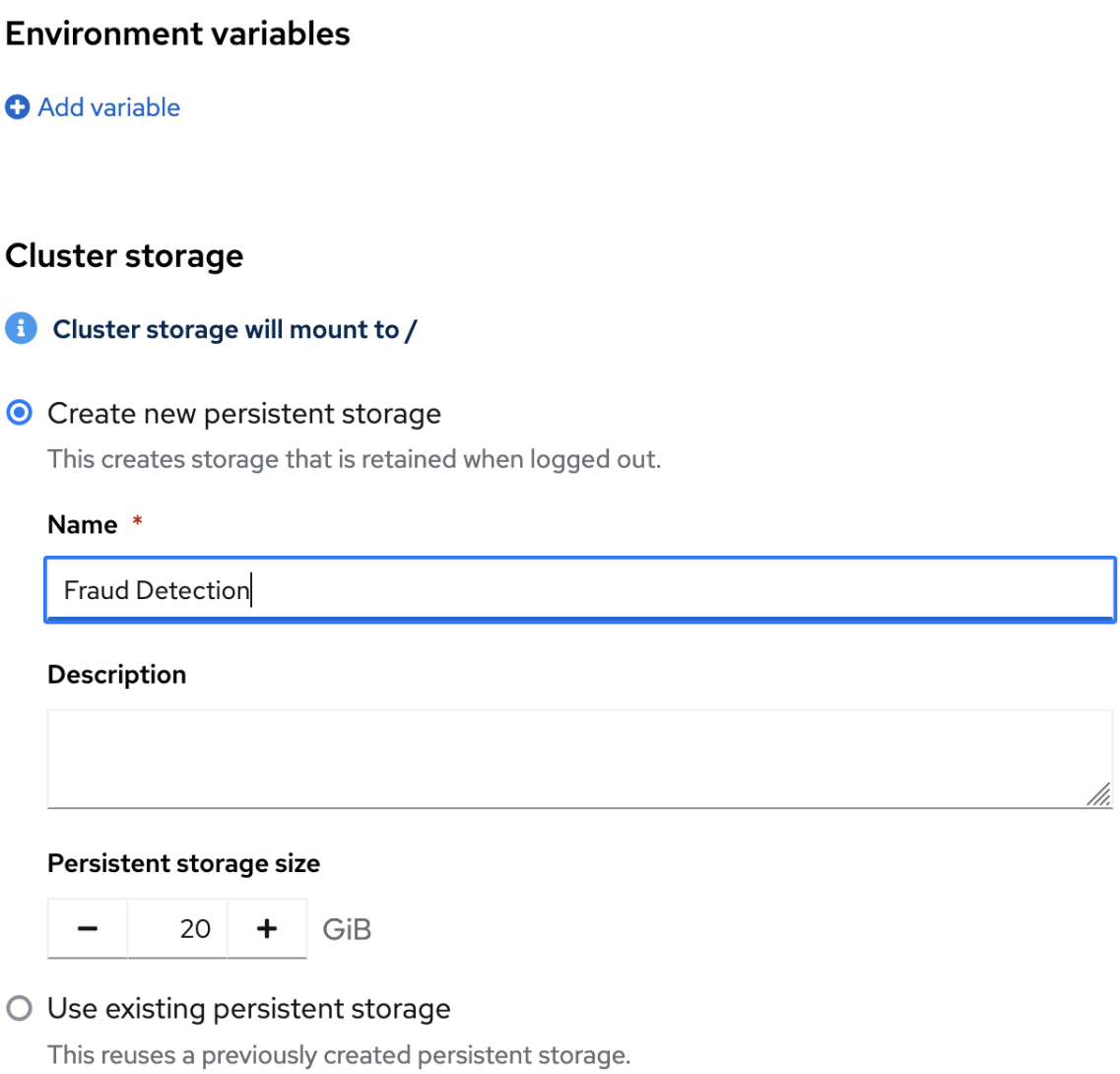
Red Hat provides several supported notebook images. In the **Notebook image** section, you can choose one of these images or any custom images that an administrator has set up for you. The **Tensorflow** image has the libraries needed for this tutorial.

4. Select the latest **Tensorflow** image.

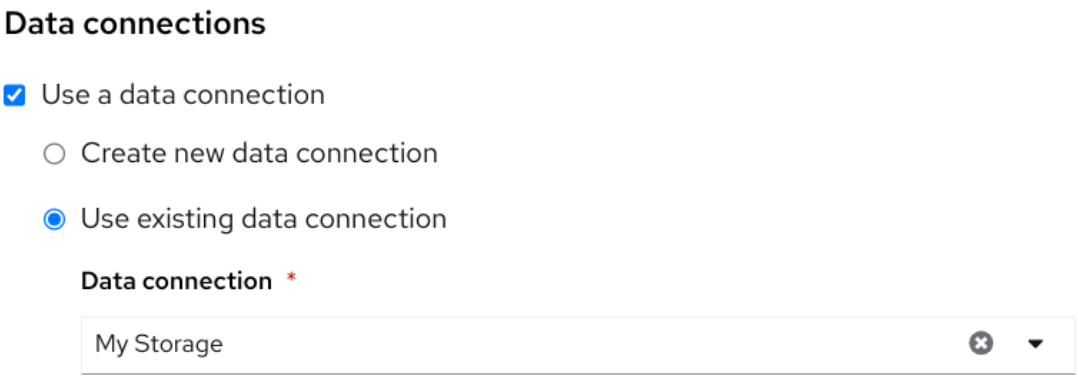


5. Select a small deployment.

6. Leave the default environment variables and storage options.



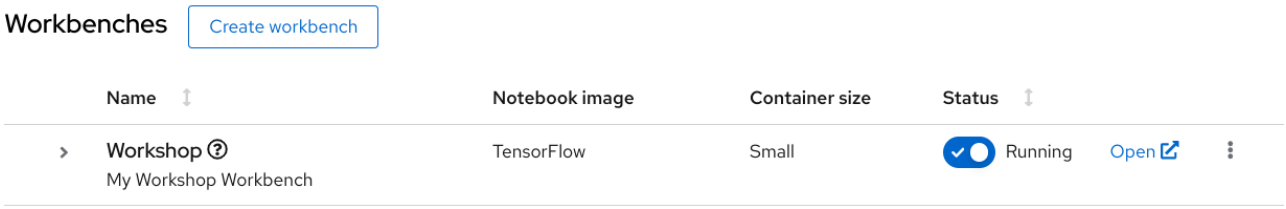
7. Under **Data connections**, select **Use existing data connection** and select **My Storage** (the object storage that you configured previously) from the list.



8. Click the **Create workbench** button.

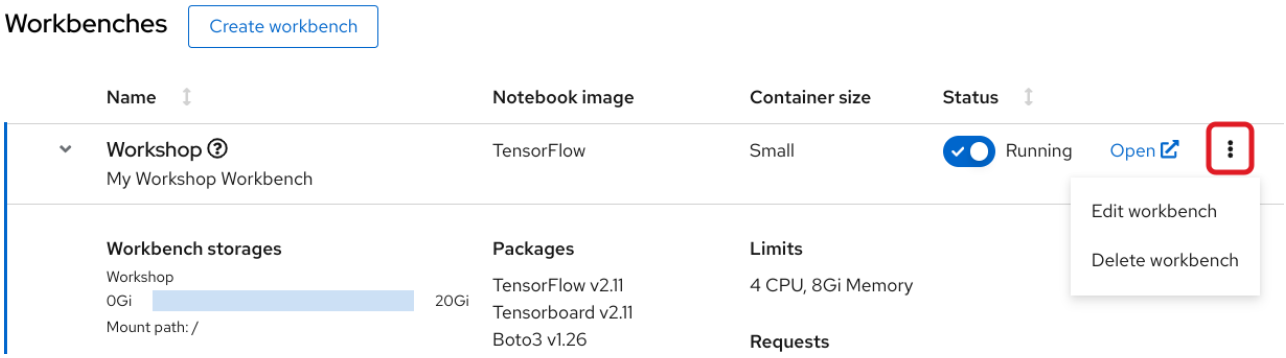
Verification

In the project details page, the status of the workbench changes from **Starting** to **Running**.



NOTE

If you made a mistake, you can edit the workbench to make changes.



Next step

Importing the tutorial files into the Jupyter environment

3.2. IMPORTING THE TUTORIAL FILES INTO THE JUPYTER ENVIRONMENT

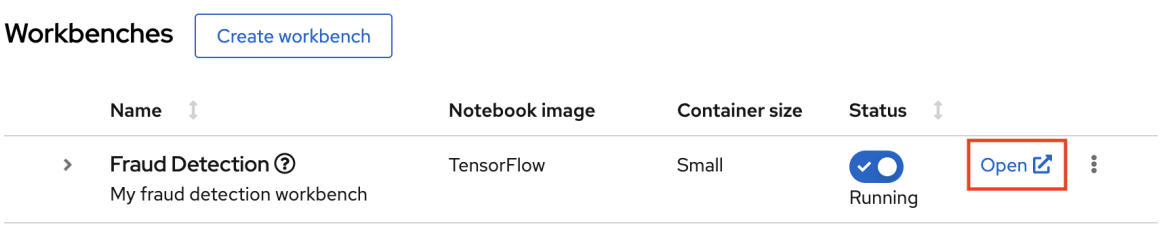
The Jupyter environment is a web-based environment, but everything you do inside it happens on **Red Hat OpenShift AI** and is powered by the **OpenShift** cluster. This means that, without having to install and maintain anything on your own computer, and without disposing of lots of local resources like CPU, GPU and RAM, you can conduct your Data Science work in this powerful and stable managed environment.

Prerequisite

You created a workbench, as described in Creating a workbench and selecting a Notebook image .

Procedure

1. Click the **Open** link next to your workbench. If prompted, log in and allow the Notebook to authorize your user.

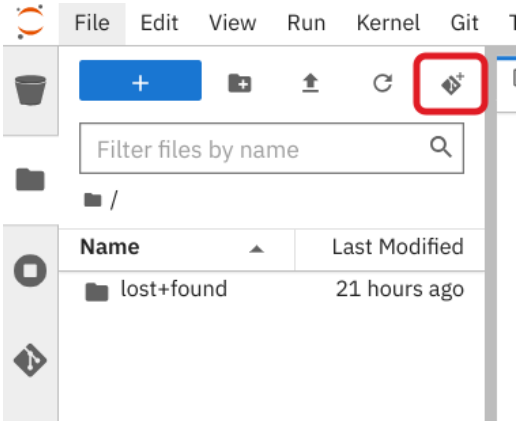


Your Jupyter environment window opens.

This file-browser window shows the files and folders that are saved inside your own personal space in OpenShift AI.

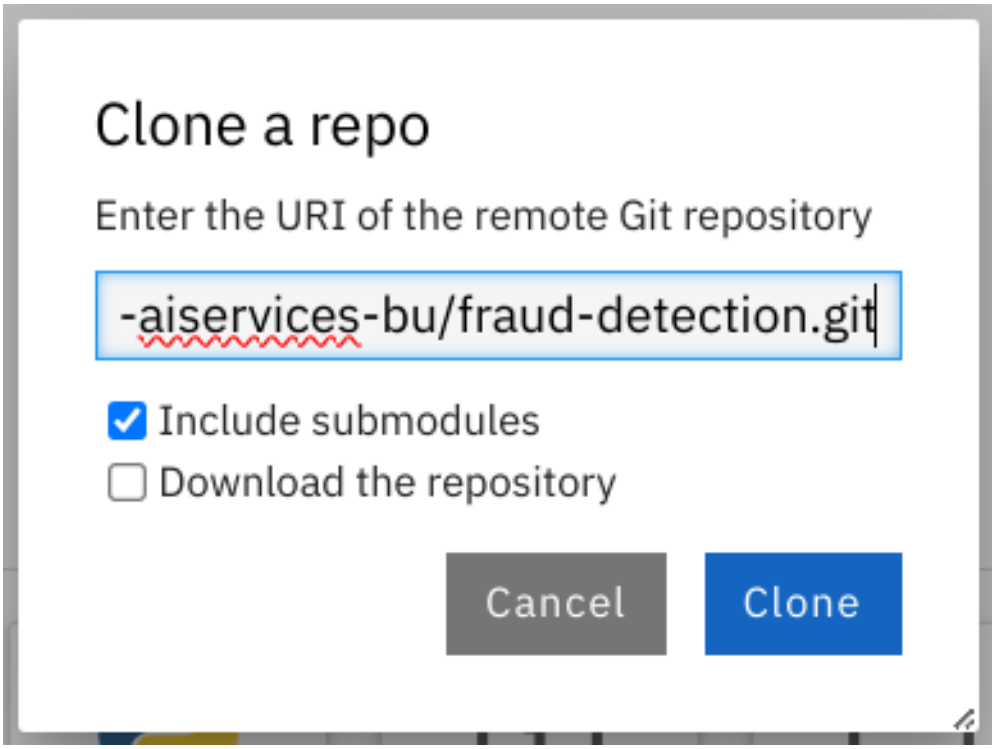
2. Bring the content of this tutorial inside your Jupyter environment:

   a. On the toolbar, click the **Git Clone** icon:



   b. Enter the following tutorial Git **https** URL:

```
https://github.com/rh-aiservices-bu/fraud-detection.git
```

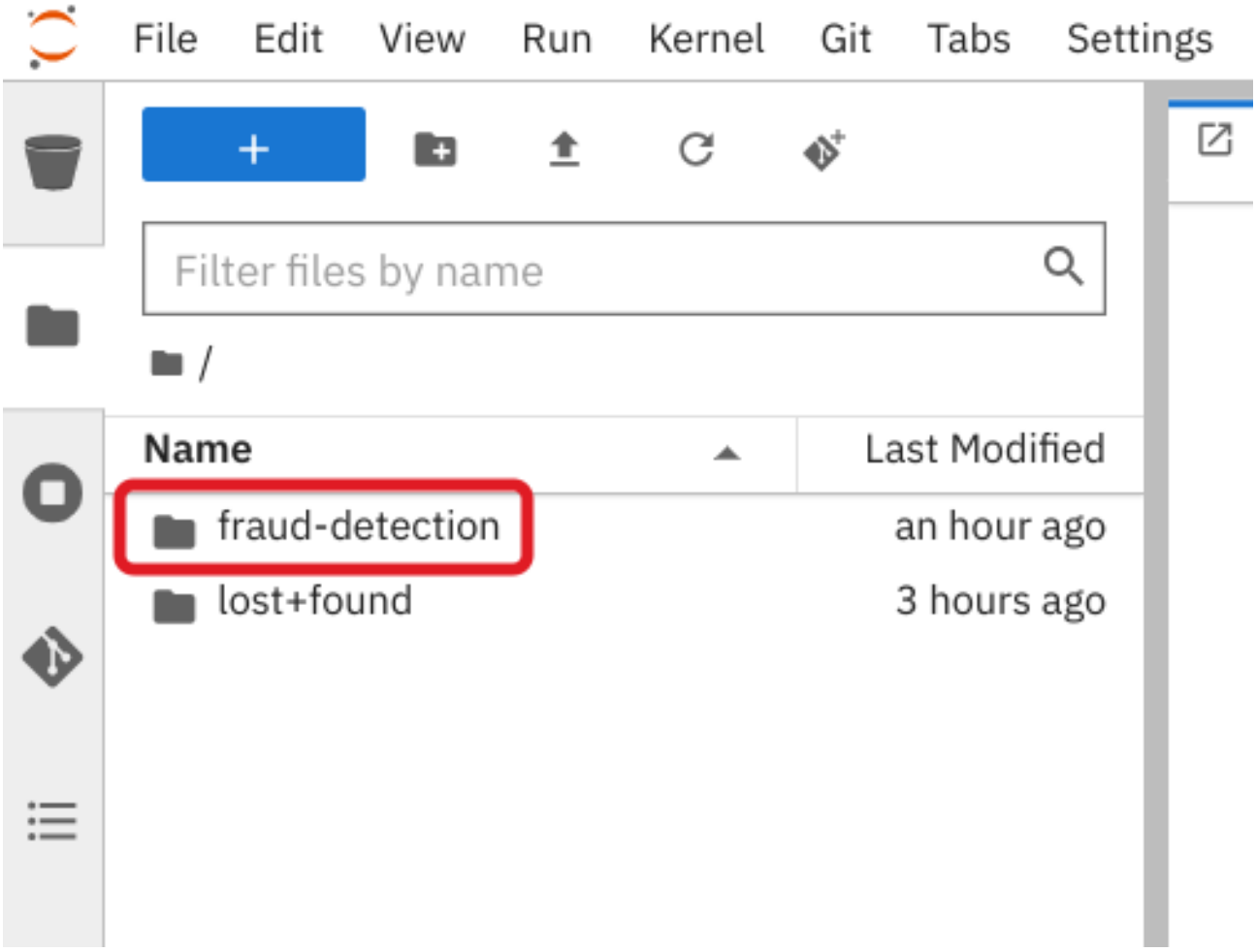


   c. Check the **Include submodules** option.

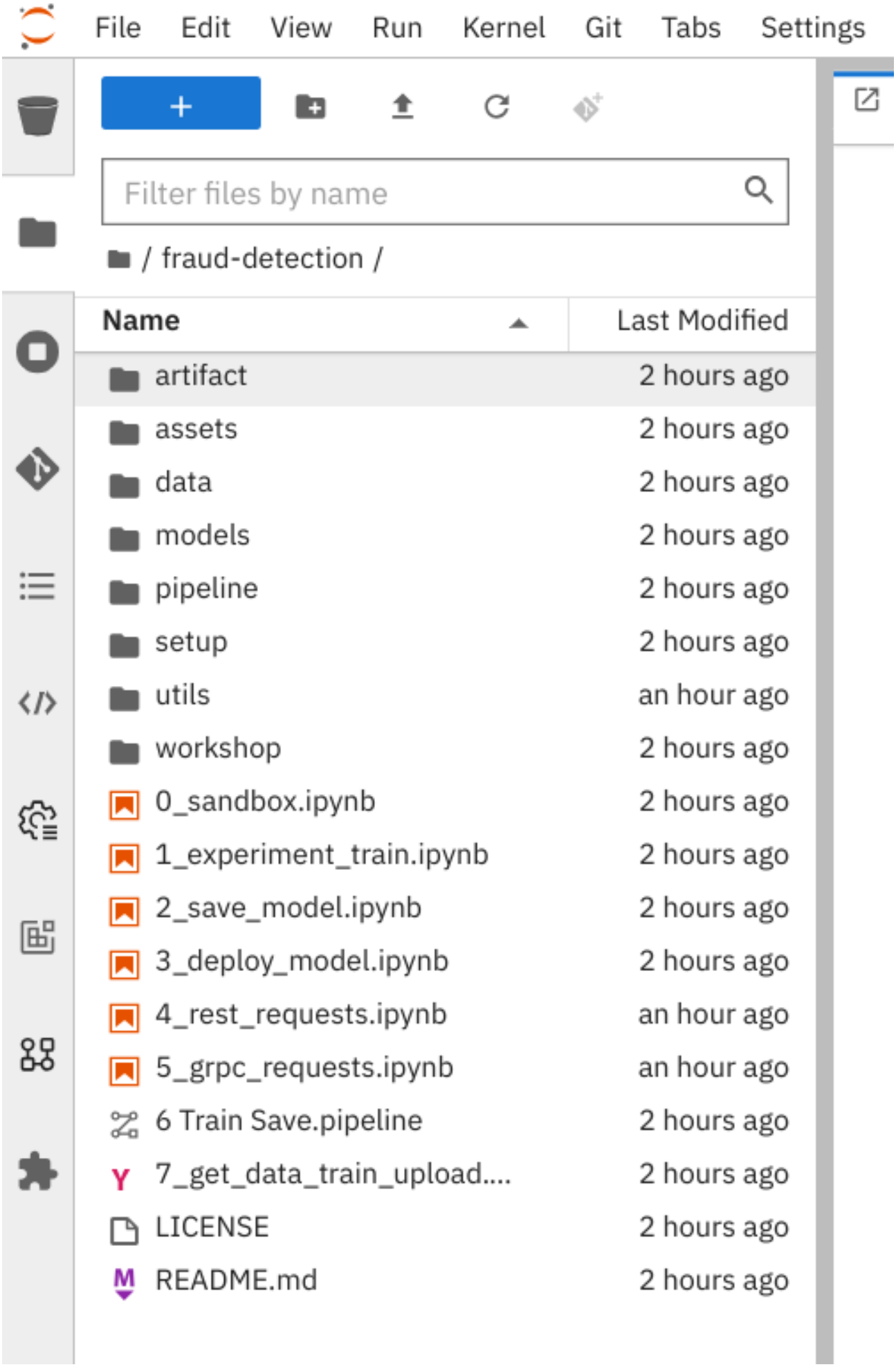
   d. Click **CLONE**.

Verification

Double-click the newly-created folder, **fraud-detection**:



In the file browser, you should see the notebooks that you cloned from Git.


File Edit View Run Kernel Git Tabs Settings
Filter files by name
/ fraud-detection /
Name
Last Modified
artifact 2 hours ago
assets 2 hours ago
data 2 hours ago
models 2 hours ago
pipeline 2 hours ago
setup 2 hours ago
utils an hour ago
workshop 2 hours ago
0_sandbox.ipynb 2 hours ago
1_experiment_train.ipynb 2 hours ago
2_save_model.ipynb 2 hours ago
3_deploy_model.ipynb 2 hours ago
4_rest_requests.ipynb an hour ago
5_grpc_requests.ipynb an hour ago
6 Train Save.pipeline 2 hours ago
7_get_data_train_upload.... 2 hours ago
LICENSE 2 hours ago
README.md 2 hours ago

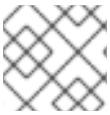
Next step

Running code in a notebook

or

Training a model

3.3. RUNNING CODE IN A NOTEBOOK



NOTE

If you're already at ease with Jupyter, you can skip to the next section .

A notebook is an environment where you have *cells* that can display formatted text or code.

This is an empty cell:

This is a cell with some code:

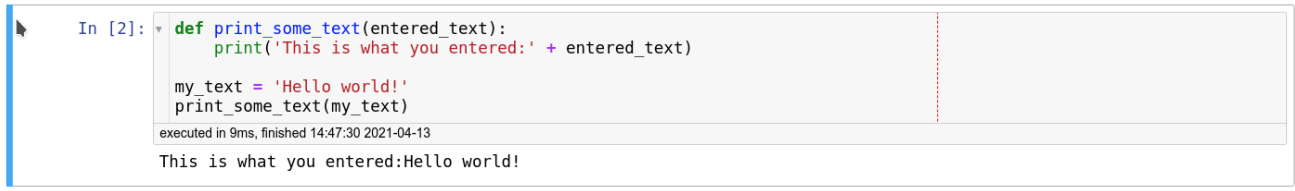
Code cells contain Python code that you can run interactively. You can modify the code and then run it. The code does not run on your computer or in the browser, but directly in the environment that you are connected to, **Red Hat OpenShift AI** in our case.

You can run a code cell from the notebook interface or from the keyboard:

- **From the user interface:**Select the cell (by clicking inside the cell or to the left side of the cell) and then click **Run** from the toolbar.

- **From the keyboard:**Press **CTRLENTER to run a cell or press SHIFTENTER** to run the cell and automatically select the next one.

After you run a cell, you can see the result of its code as well as information about when the cell was run, as shown in this example:



When you save a notebook, the code and the results are saved. You can reopen the notebook to look at the results without having to run the program again, while still having access to the code.

Notebooks are so named because they are like a physical *notebook*: you can take notes about your experiments (which you will do), along with the code itself, including any parameters that you set. You can see the output of the experiment inline (this is the result from a cell after it's run), along with all the notes that you want to take (to do that, from the menu switch the cell type from **Code** to **Markdown**).

3.3.1. Try it

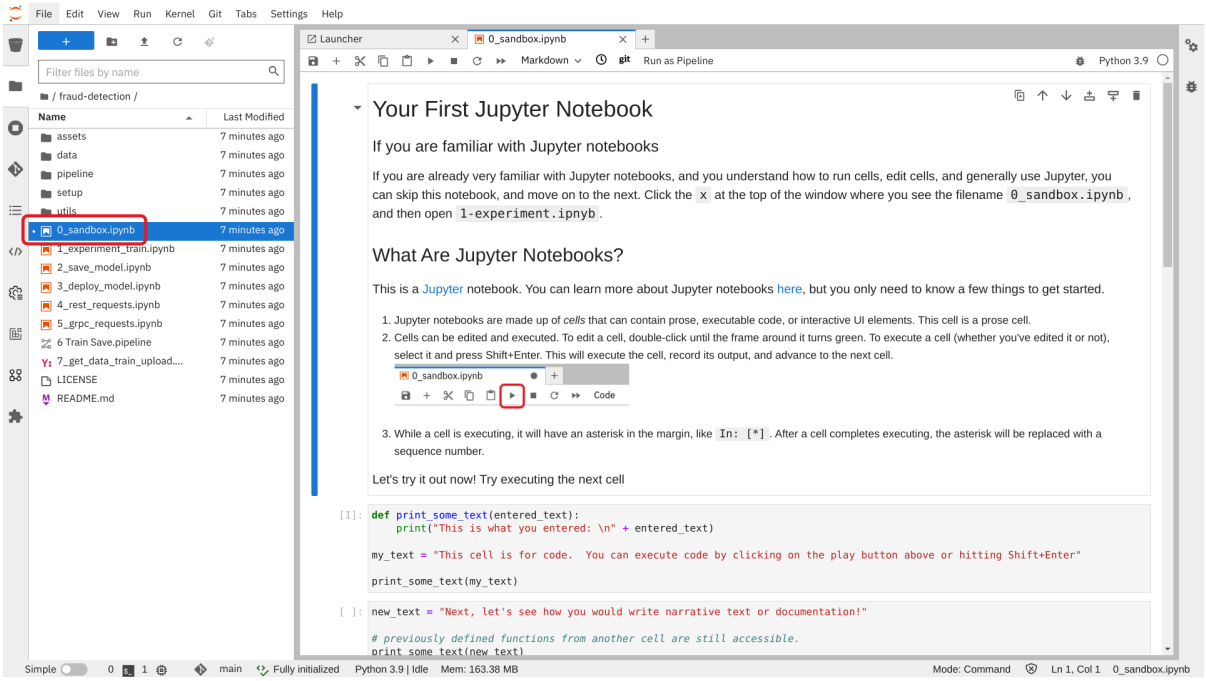
Now that you know the basics, give it a try!

Prerequisite

- You have imported the tutorial files into your Jupyter environment as described in Importing the tutorial files into the Jupyter environment.

Procedure

1. In your Jupyter environment, locate the **0_sandbox.ipynb** file and double-click it to launch the notebook. The notebook opens in a new tab in the content section of the environment.



2. Experiment by, for example, running the existing cells, adding more cells and creating functions. You can do what you want - it's your environment and there is no risk of breaking anything or impacting other users. This environment isolation is also a great advantage brought by OpenShift AI.

3. Optionally, create a new notebook in which the code cells are run by using a Python 3 kernel:

    a. Create a new notebook by either selecting **File →New →Notebook** or by clicking the Python 3 tile in the Notebook section of the launcher window:

You can use different kernels, with different languages or versions, to run in your notebook.

Additional resource

To learn more about notebooks, go to the Jupyter site.

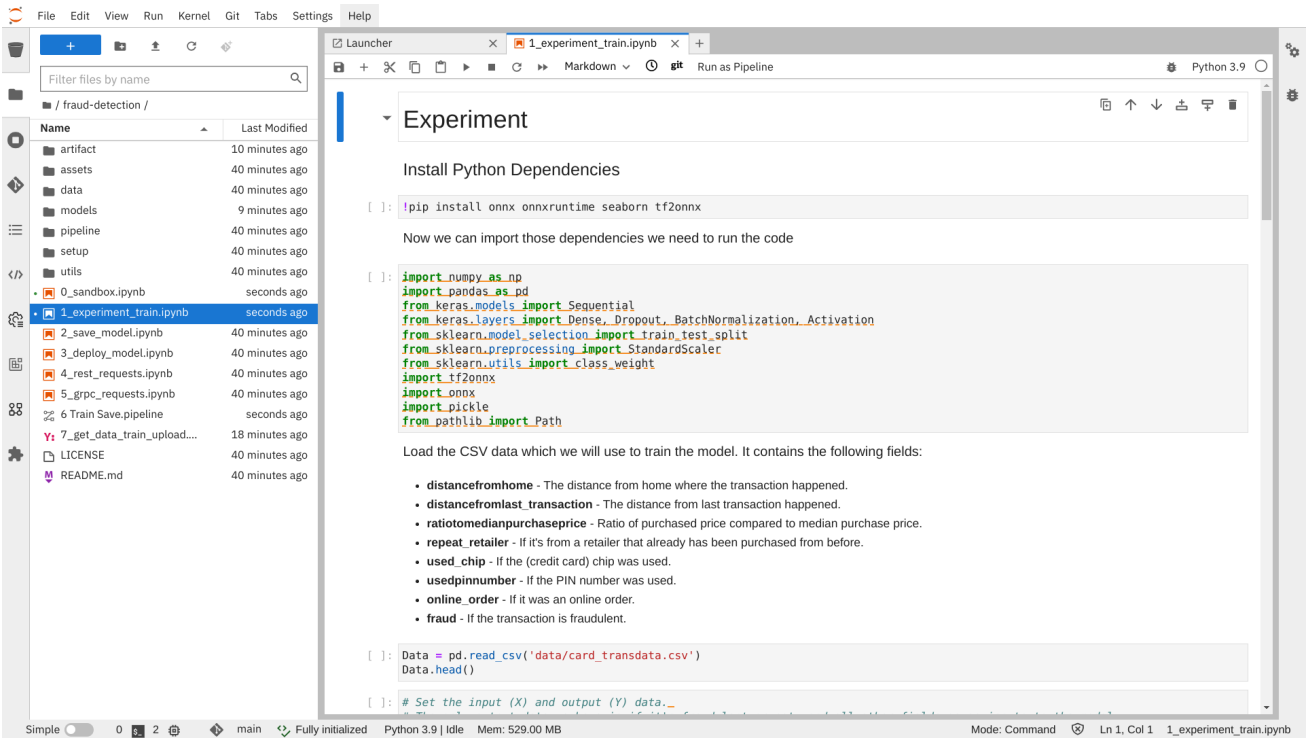
Next step

Training a model

3.4. TRAINING A MODEL

Now that you know how the Jupyter notebook environment works, the real work can begin!

In your notebook environment, open the **1_experiment_train.ipynb** file and follow the instructions directly in the notebook. The instructions guide you through some simple data exploration, experimentation, and model training tasks.



Next step

Preparing a model for deployment

CHAPTER 4. DEPLOYING AND TESTING A MODEL

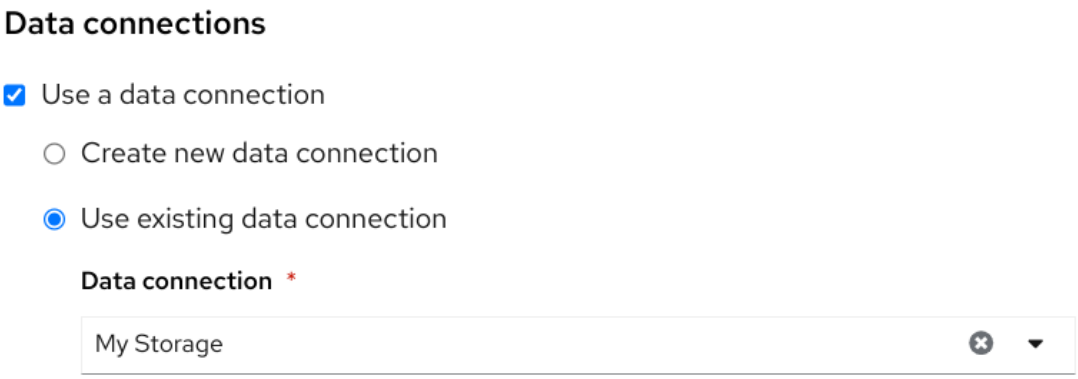
4.1. PREPARING A MODEL FOR DEPLOYMENT

After you train a model, you can deploy it by using the OpenShift AI model serving capabilities.

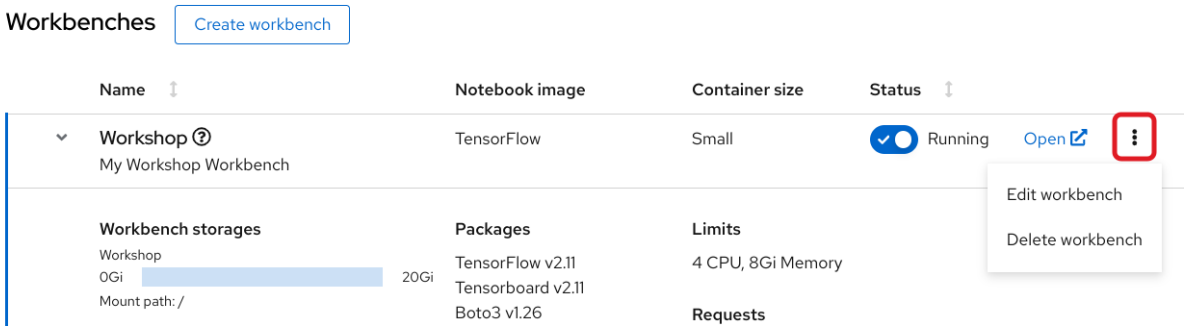
To prepare a model for deployment, you must move the model from your workbench to your S3-compatible object storage. You use the data connection that you created in the Storing data with data connections section and upload the model from a notebook. You also convert the model to the portable ONNX format. ONNX allows you to transfer models between frameworks with minimal preparation and without the need for rewriting the models.

Prerequisites

- You created the data connection **My Storage**.

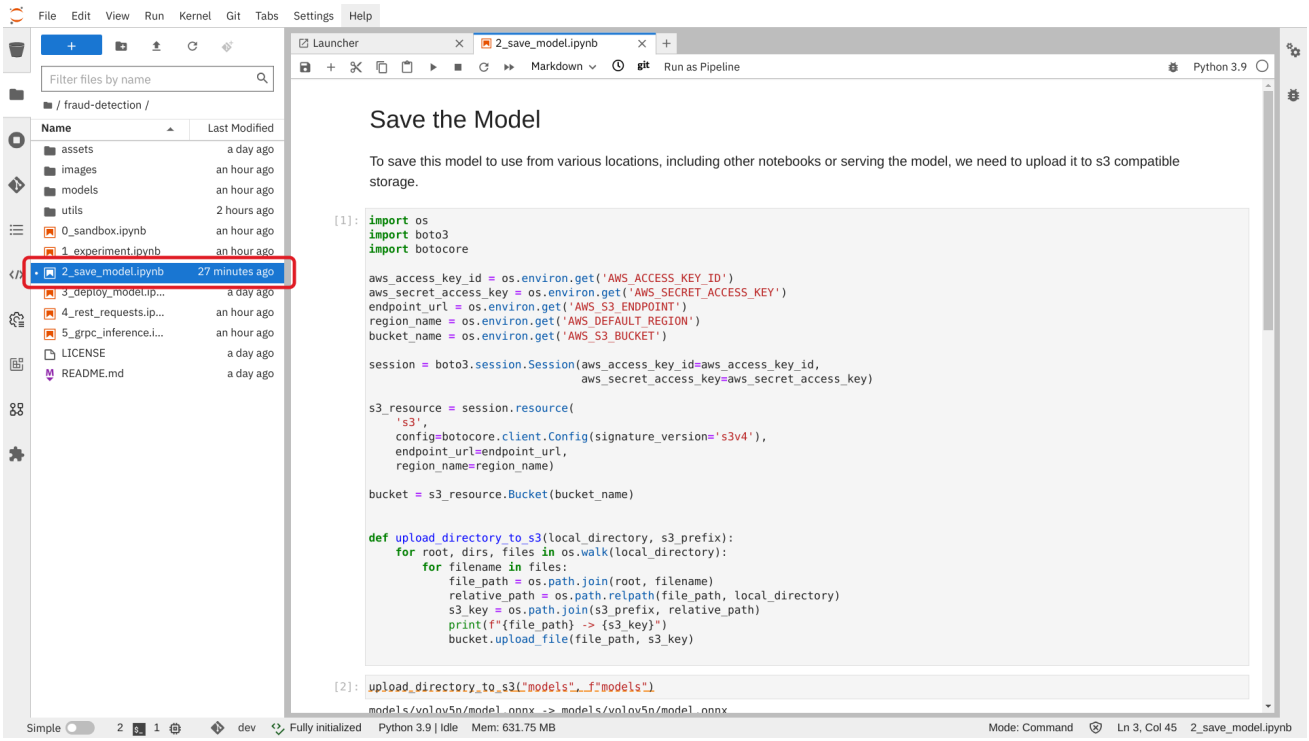


- You added the **My Storage** data connection to your workbench.



Procedure

1. In your Jupyter environment, open the **2_save_model.ipynb** file.
2. Follow the instructions in the notebook to make the model accessible in storage and save it in the portable ONNX format.



Verification

When you have completed the notebook instructions, the **models/fraud/model.onnx** file is in your object storage and it is ready for your model server to use.

Next step

Deploying a model

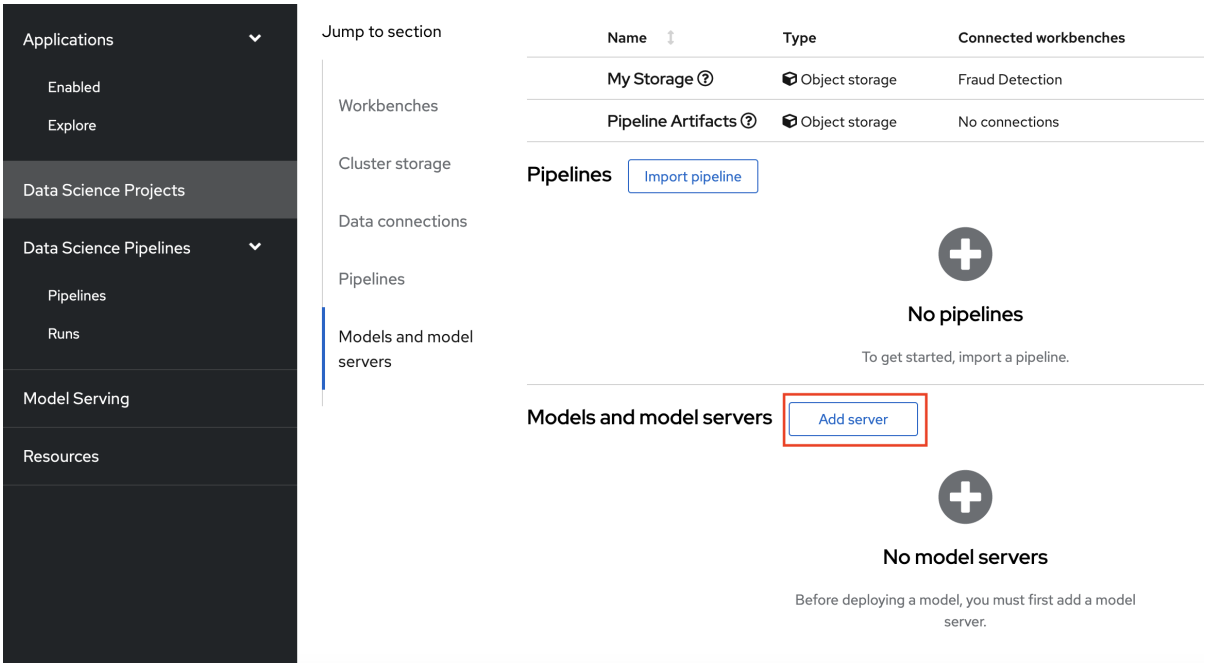
4.2. DEPLOYING A MODEL

Now that the model is accessible in storage and saved in the portable ONNX format, you can use an OpenShift AI model server to deploy it as an API.

OpenShift AI multi-model servers can host several models at once. You create a new model server and deploy your model to it.

Procedure

1. In the OpenShift AI dashboard, navigate to **Models and model servers**.
2. Click **Add server**.

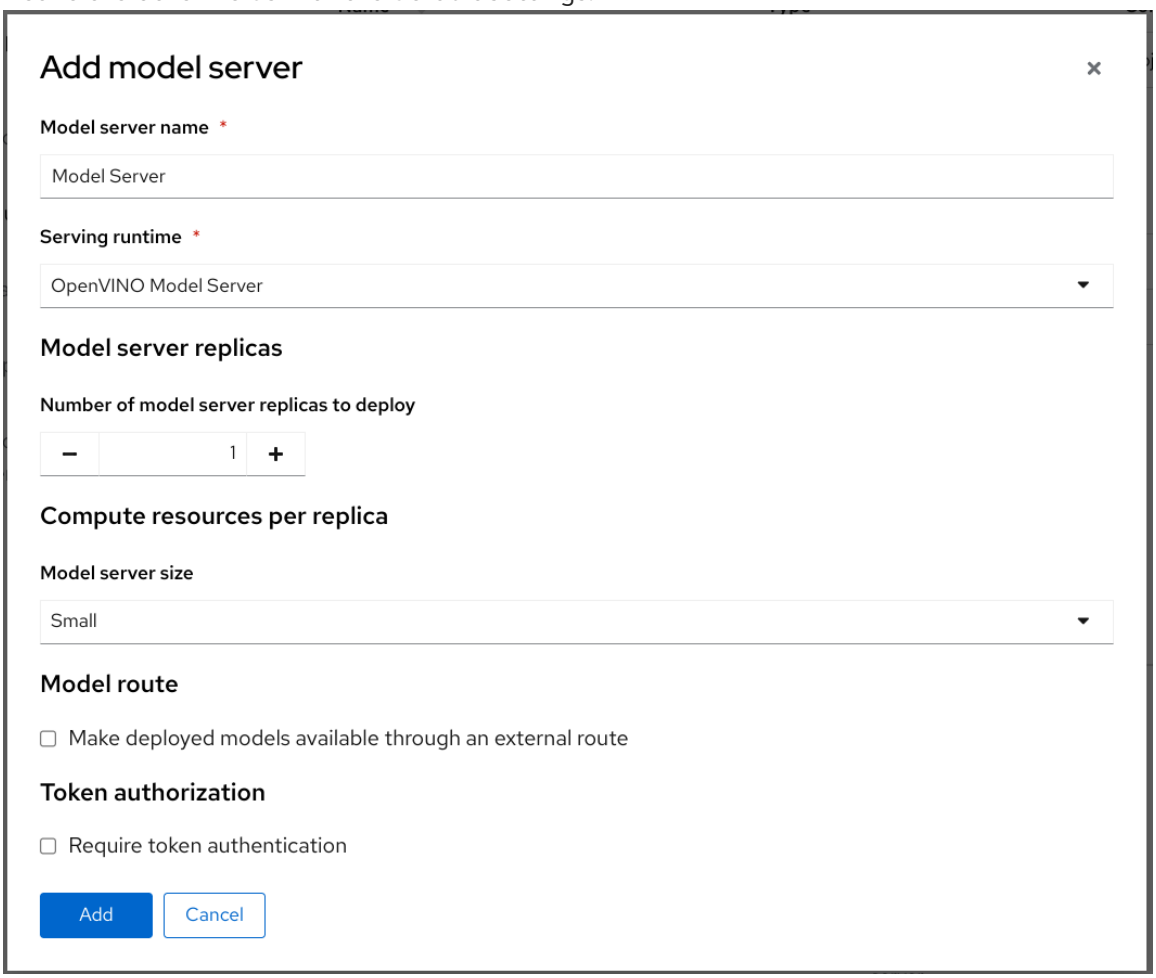


3. In the form:

   a. Fill out the **Model server name**, for example **Model Server**.

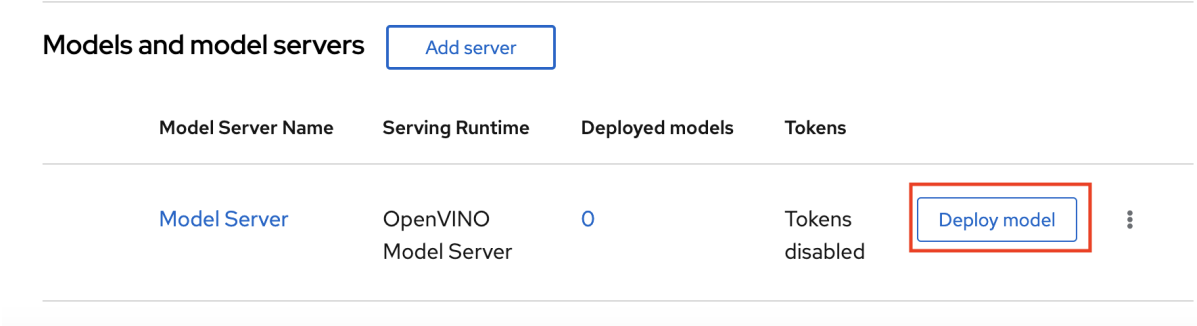
   b. Select **OpenVINO Model Server**.

   c. Leave the other fields with the default settings.



4. Click **Add**.

5. In the **Models and model servers** list, next to the new model server, click **Deploy model**.



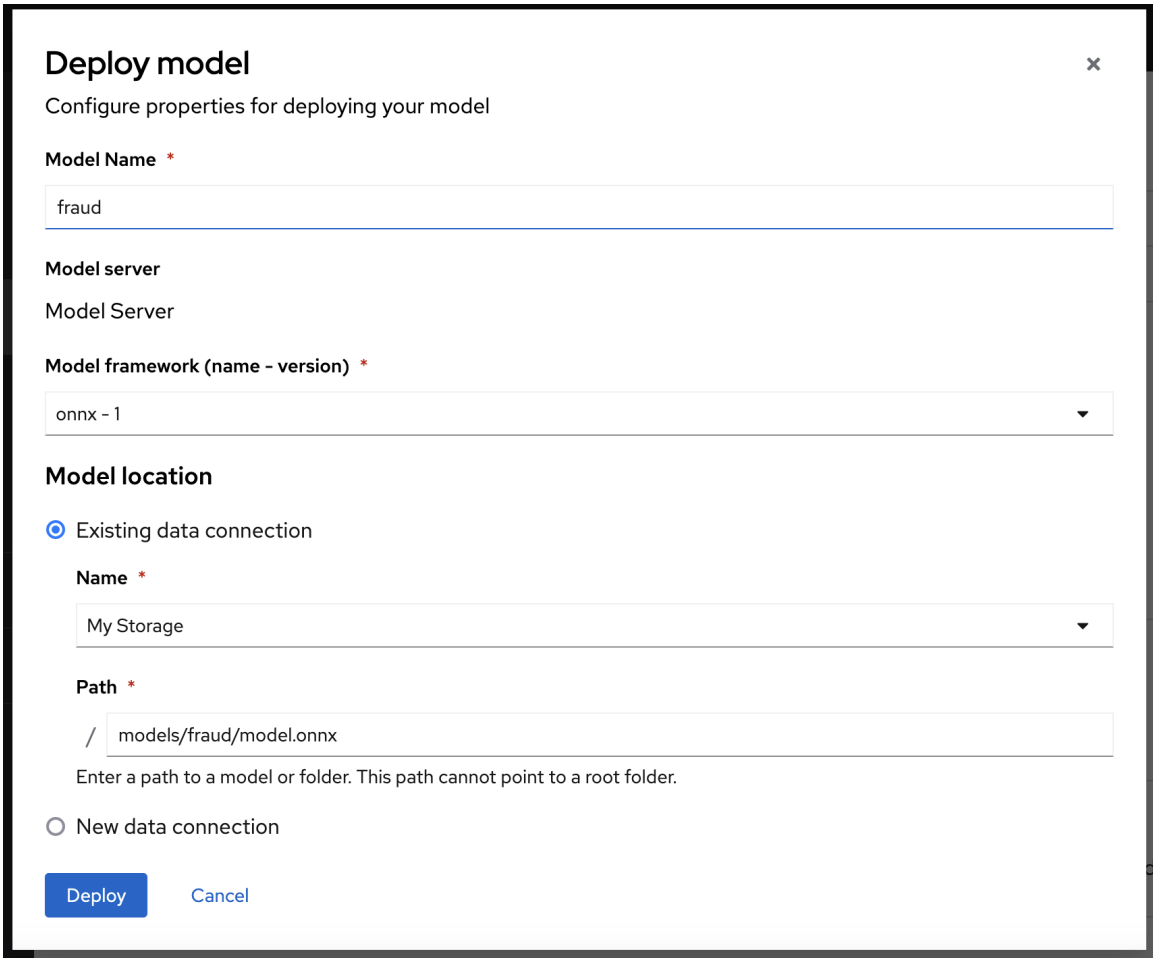
6. In the form:

   a. Fill out the **Model Name** with the value **fraud**.

   b. Select the server that you created (for example, **Model Server**).

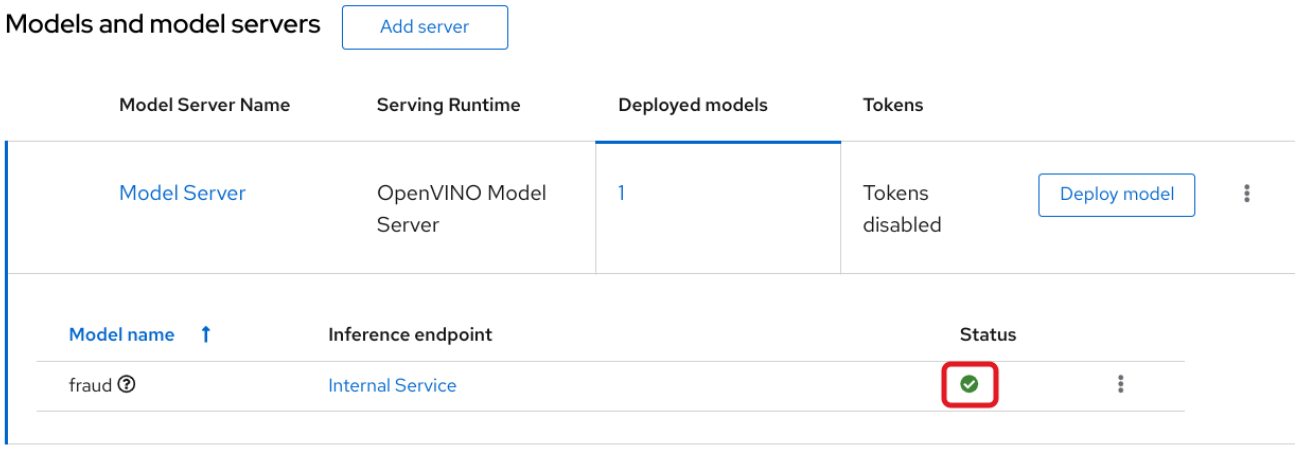
   c. Select **Existing data connection** **My Storage**.

   d. Enter the path to your uploaded model: **models/fraud/model.onnx**



Verification

Wait for the model to deploy and for the **Status** to show a green checkmark.



Next step

Testing the model API

4.3. TESTING THE MODEL API

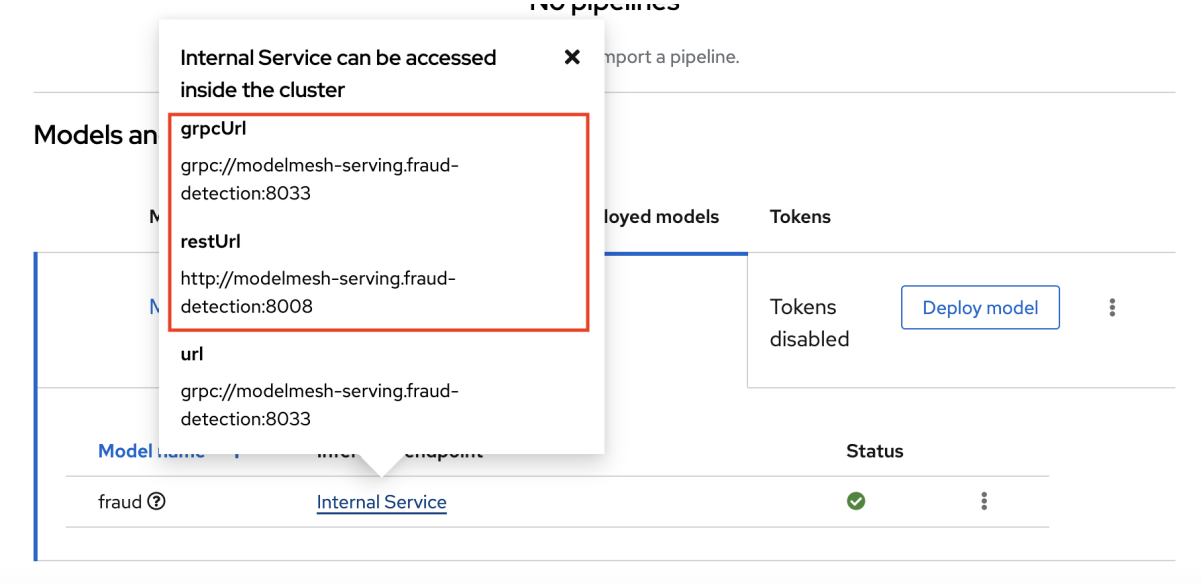
Now that you've deployed the model, you can test its API endpoints.

When you created the model server, you did **not** create a route for external access to the API and you did not protect it with an authentication token. By default, if you do not specify external access, the model server provides an internal endpoint with no authentication.

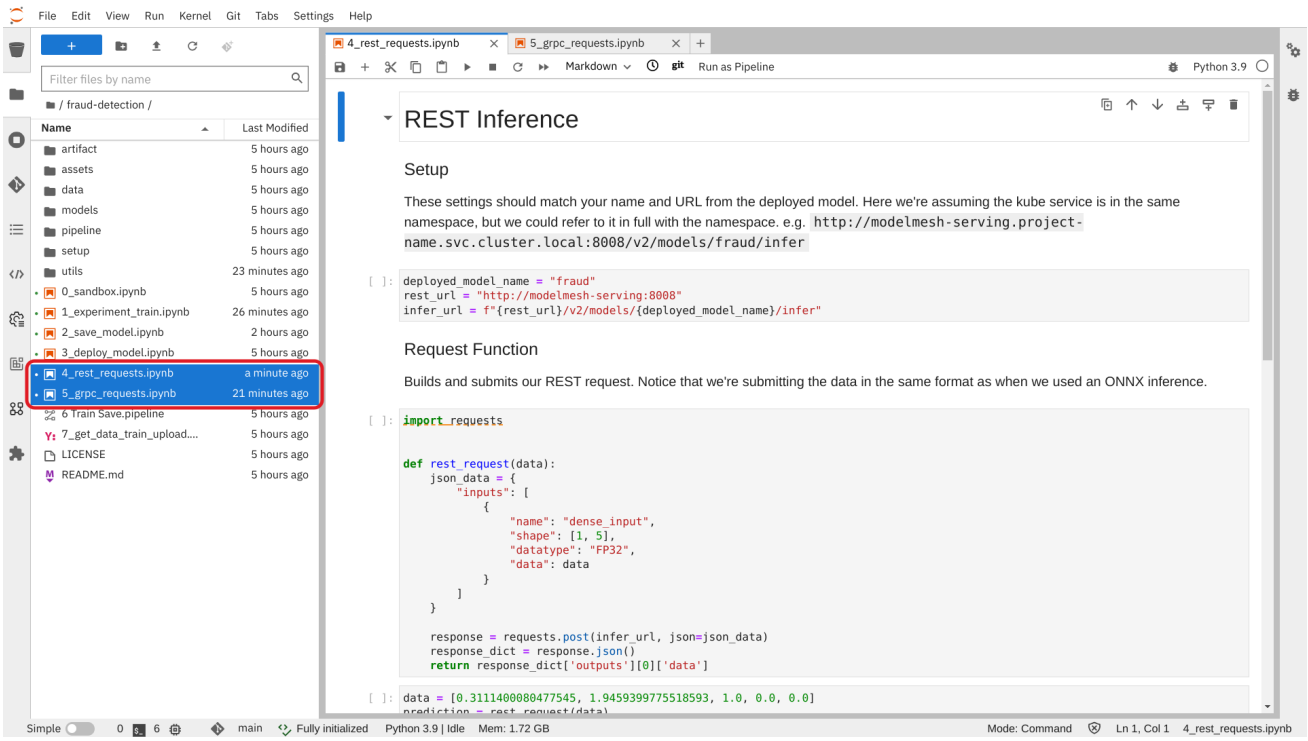
You can communicate directly with this internal service in the same way that an application in your project would. An easy way to test it is from a notebook in the same project.

Procedure

1. In the OpenShift AI dashboard, navigate to the project details page and scroll down to the **Models and model servers** section.
2. Take note of the model's resource name (API endpoint name) and the internal service's grpcURL and restURL. You need this information when you test the model API.



3. Return to the Jupyter environment and try out your new endpoint. You'll try REST API calls in **4_rest_requests.ipynb** and gRPC requests in **5_grpc_requests.ipynb**.



Next step

Automating workflows with data science pipelines

Running a data science pipeline generated from Python code

CHAPTER 5. IMPLEMENTING PIPELINES

5.1. AUTOMATING WORKFLOWS WITH DATA SCIENCE PIPELINES

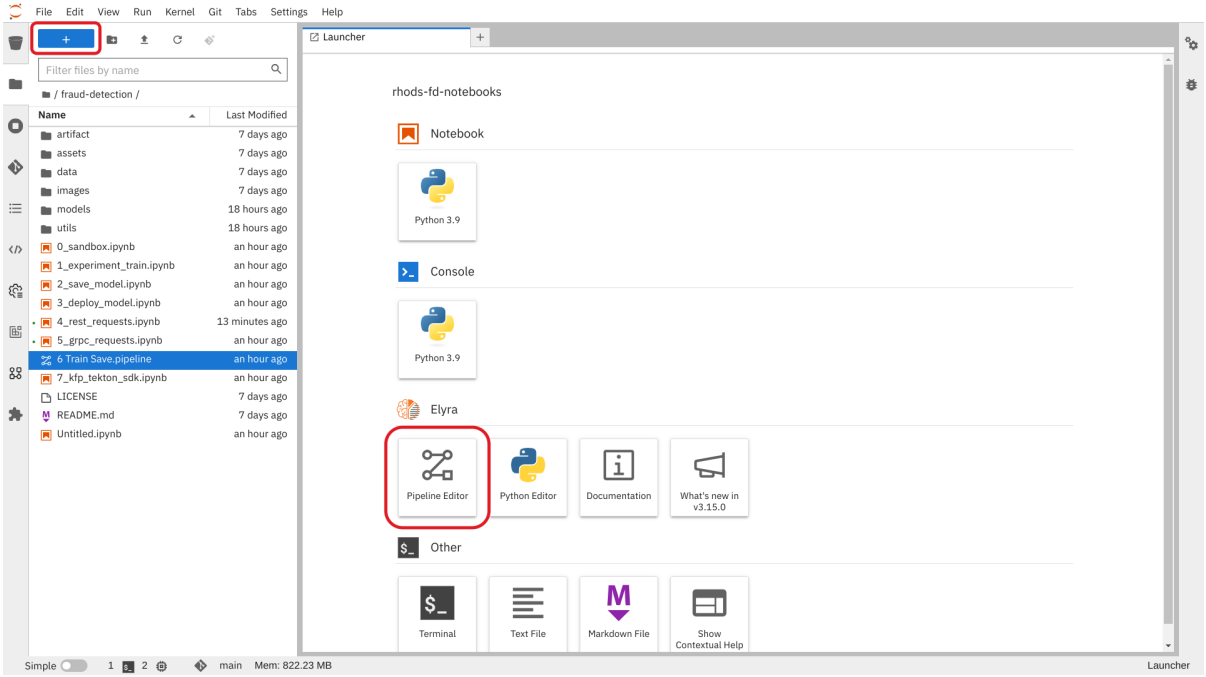
In previous sections of this tutorial, you used a notebook to train and save your model. Optionally, you can automate these tasks by using Red Hat OpenShift AI pipelines. Pipelines offer a way to automate the execution of multiple notebooks and Python code. By using pipelines, you can execute long training jobs or retrain your models on a schedule without having to manually run them in a notebook.

In this section, you create a simple pipeline by using the GUI pipeline editor. The pipeline uses the notebook that you used in previous sections to train a model and then save it to S3 storage.

Note: Your completed pipeline should look like the one in the **6 Train Save.pipeline** file.

5.1.1. Create a pipeline

1. Open your workbench's JupyterLab environment. If the launcher is not visible, click **+** to open it.

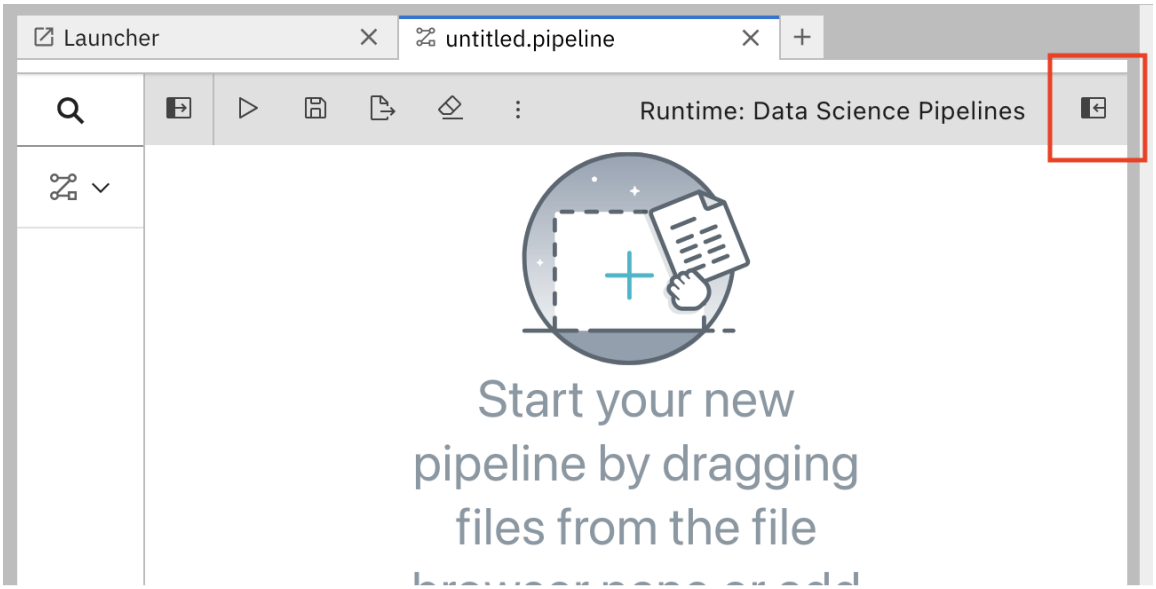


2. Click **Pipeline Editor**.

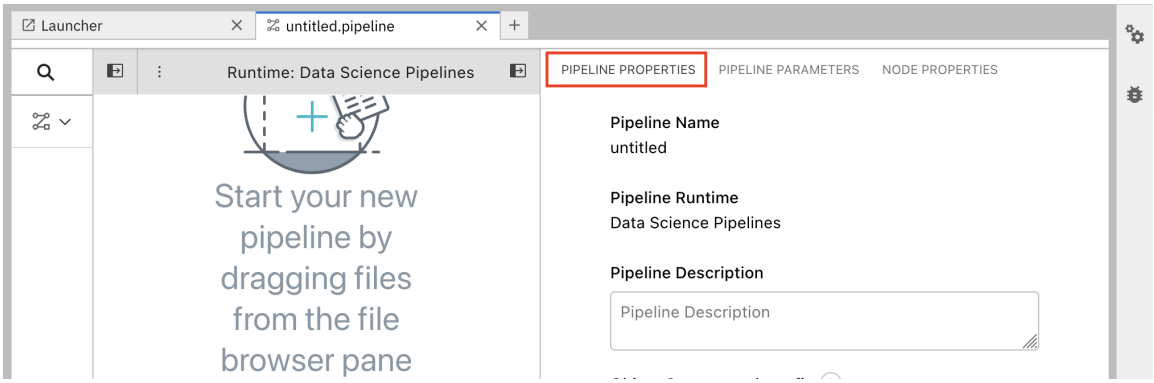
You've created a blank pipeline!

3. Set the default runtime image for when you run your notebook or Python code.

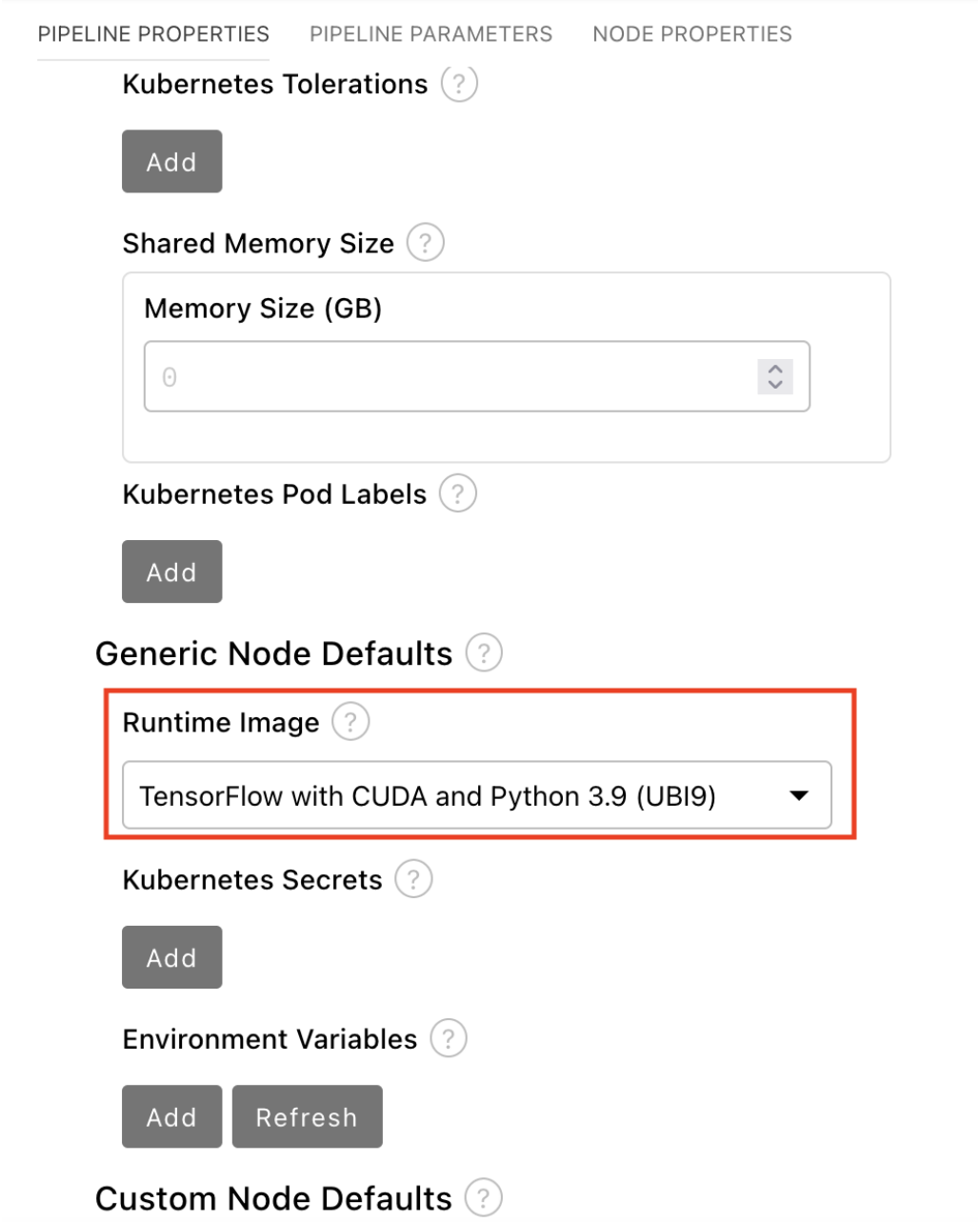
   a. In the pipeline editor, click **Open Panel**



b. Select the **Pipeline Properties** tab.



c. In the **Pipeline Properties** panel, scroll down to **Generic Node Defaults** and **Runtime Image**. Set the value to **Tensorflow with Cuda and Python 3.9 (UBI 9)**.

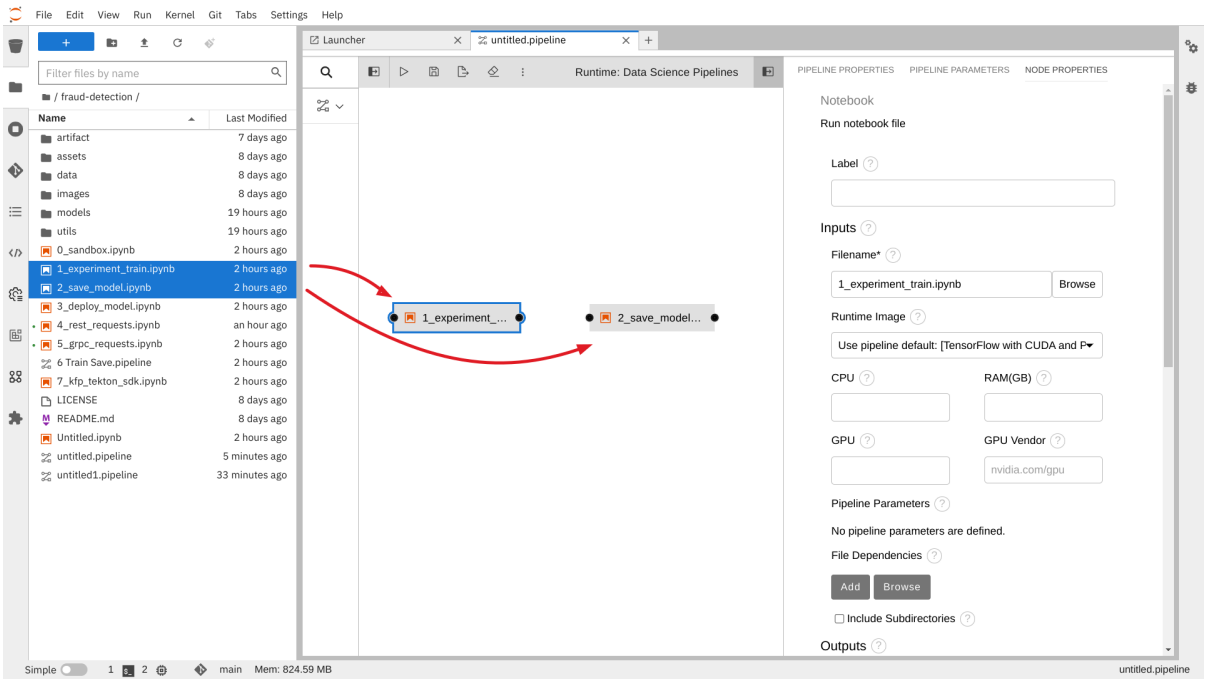


4. Save the pipeline.

5.1.2. Add nodes to your pipeline

Add some steps, or **nodes** in your pipeline. Your two nodes will use the **1_experiment_train.ipynb** and **2_save_model.ipynb** notebooks.

1. From the file-browser panel, drag the **1_experiment_train.ipynb** and **2_save_model.ipynb** notebooks onto the pipeline canvas.



2. Click the output port of **1_experiment_train.ipynb** and drag a connecting line to the input port of **2_save_model.ipynb**.

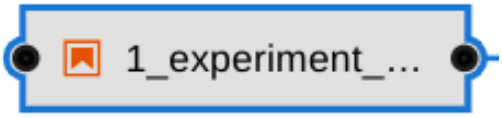
3. Save the pipeline.

5.1.3. Specify the training file as a dependency

Set node properties to specify the training file as a dependency.

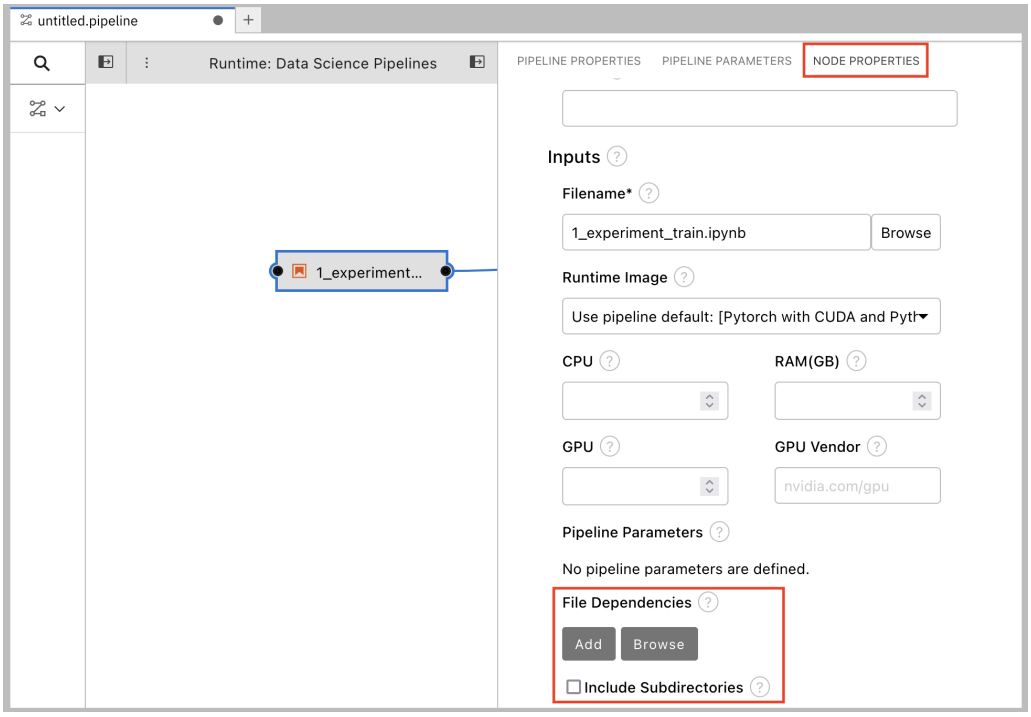
Note: If you don't set this file dependency, the file is not included in the node when it runs and the training job fails.

1. Click the **1_experiment_train.ipynb** node.

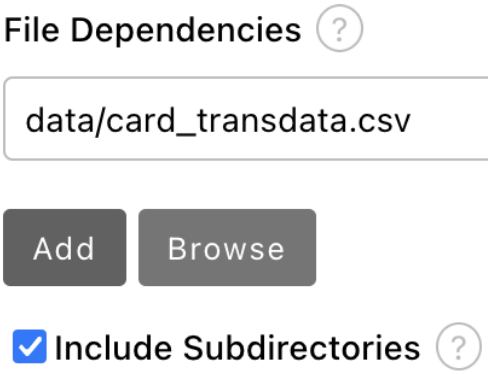


2. In the **Properties** panel, click the **Node Properties** tab.

3. Scroll down to the **File Dependencies** section and then click **Add**.



4. Set the value to **data/card_transdata.csv** which contains the data to train your model.
5. Select the **Include Subdirectories** option and then click **Add**.



6. Save the pipeline.

5.1.4. Create and store the ONNX-formatted output file

In node 1, the notebook creates the **models/fraud/model.onnx** file. In node 2, the notebook uploads that file to the S3 storage bucket. You must set **models/fraud/model.onnx** file as the output file for both nodes.

1. Select node 1 and then select the **Node Properties** tab.
2. Scroll down to the **Output Files** section, and then click **Add**.
3. Set the value to **models/fraud/model.onnx** and then click **Add**.

4. Repeat steps 1-3 for node 2.

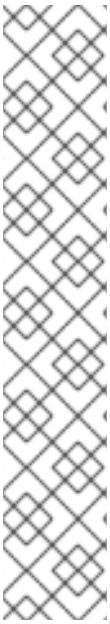
5.1.5. Configure the data connection to the S3 storage bucket

In node 2, the notebook uploads the model to the S3 storage bucket.

You must set the S3 storage bucket keys by using the secret created by the **My Storage** data connection that you set up in the Storing data with data connections section of this tutorial.

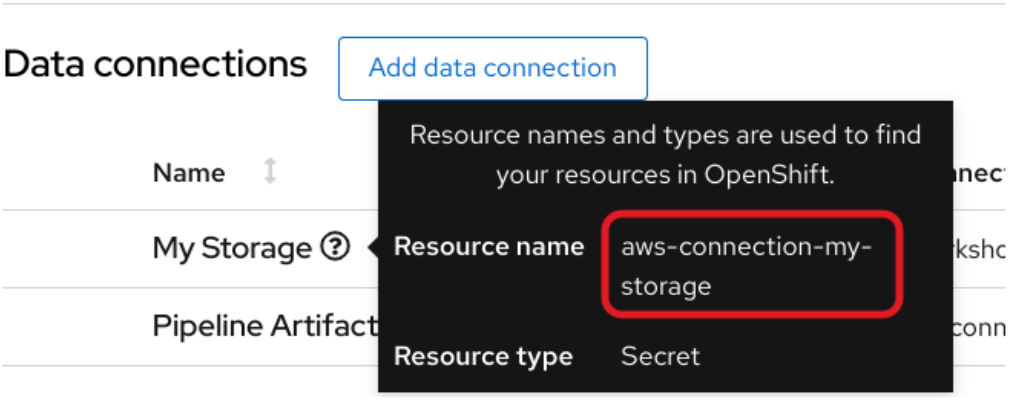
You can use this secret in your pipeline nodes without having to save the information in your pipeline code. This is important, for example, if you want to save your pipelines - without any secret keys - to source control.

The secret is named **aws-connection-my-storage**.



NOTE

If you named your data connection something other than **My Storage**, you can obtain the secret name in the OpenShift AI dashboard by hovering over the resource information icon **?** in the **Data Connections** tab.



The **aws-connection-my-storage** secret includes the following fields:

- **AWS_ACCESS_KEY_ID**
- **AWS_DEFAULT_REGION**
- **AWS_S3_BUCKET**
- **AWS_S3_ENDPOINT**

- **AWS_SECRET_ACCESS_KEY**

You must set the secret name and key for each of these fields.

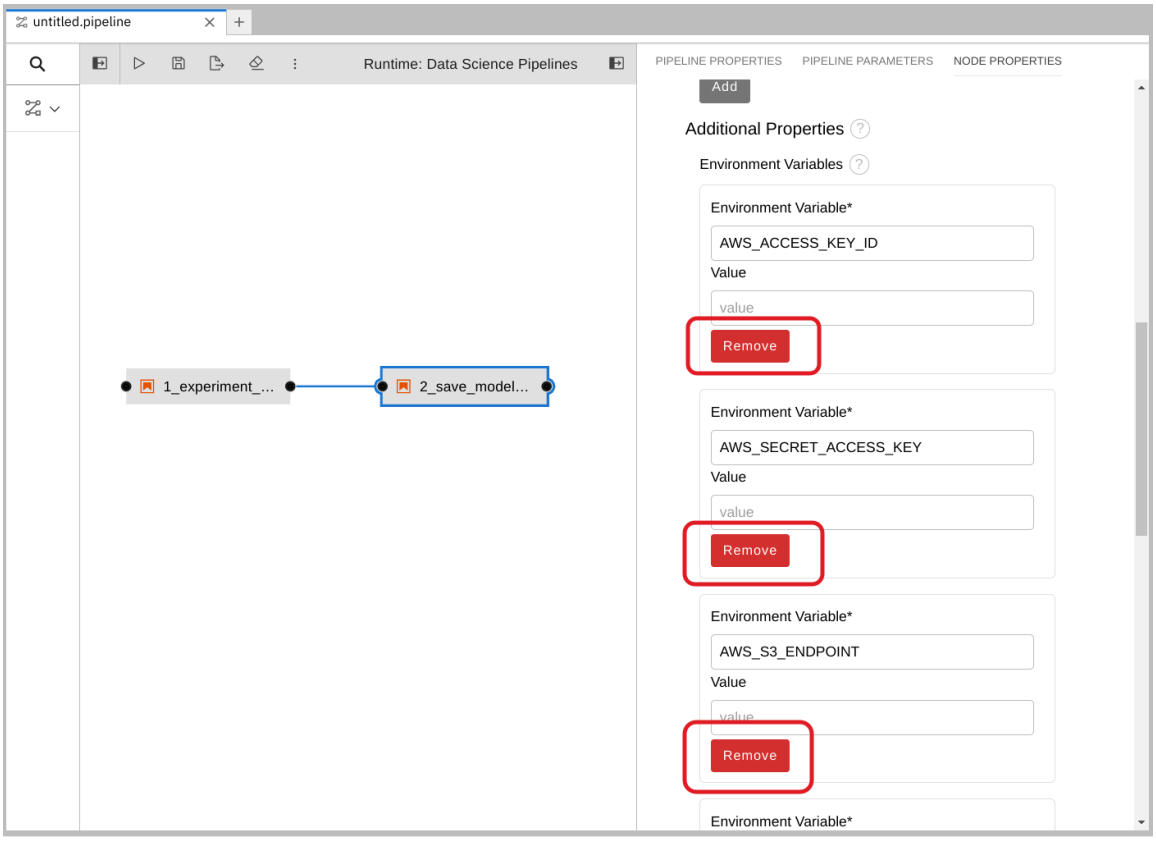
Procedure

1. Remove any pre-filled environment variables.

   a. Select node 2, and then select the **Node Properties** tab.
   Under **Additional Properties**, note that some environment variables have been pre-filled. The pipeline editor inferred that you'd need them from the notebook code.

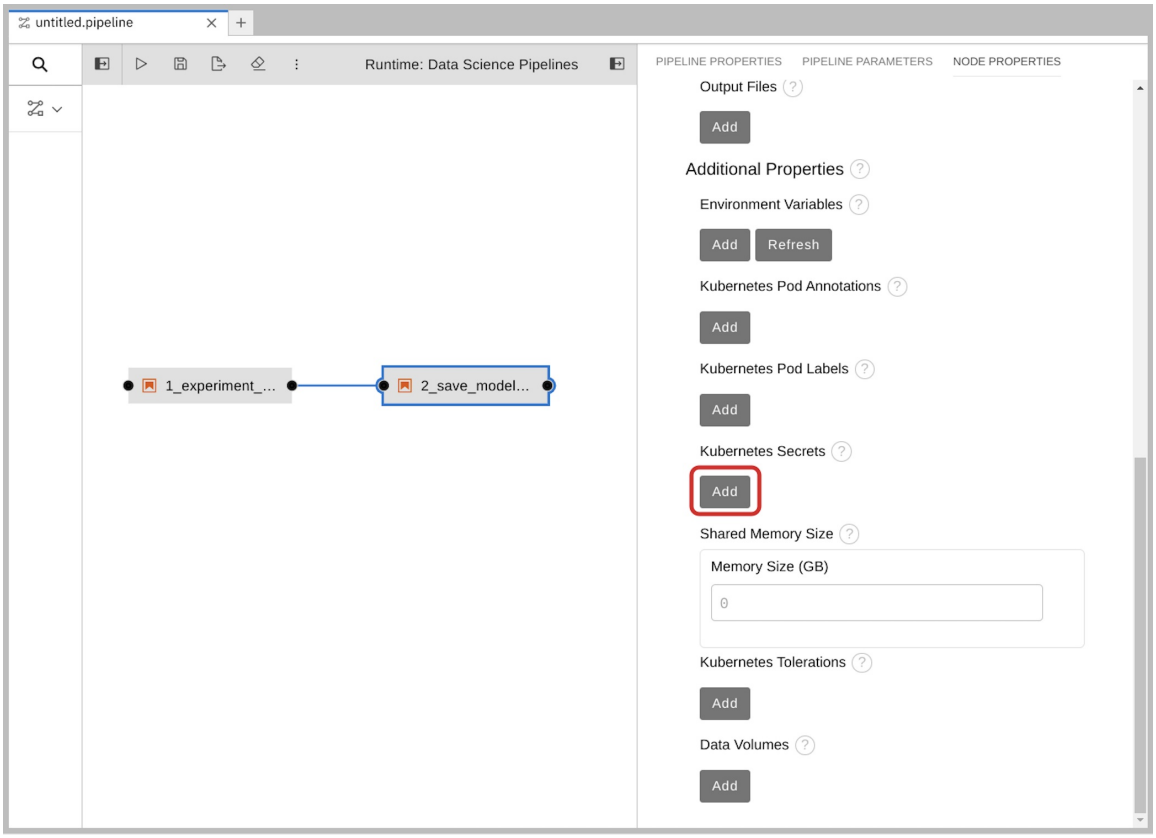
   Since you don't want to save the value in your pipelines, remove all of these environment variables.

   b. Click **Remove** for each of the pre-filled environment variables.



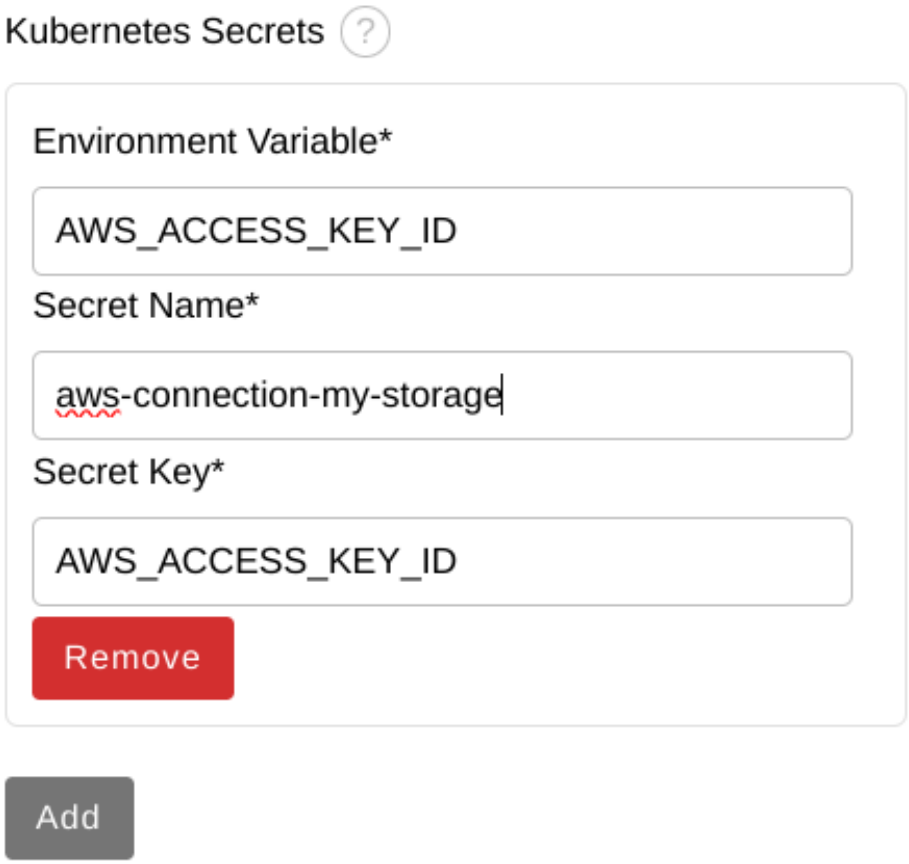
2. Add the S3 bucket and keys by using the Kubernetes secret.

   a. Under **Kubernetes Secrets**, click **Add**.



b. Enter the following values and then click **Add**.

- Environment Variable: **AWS_ACCESS_KEY_ID**
- Secret Name: **aws-connection-my-storage**
- Secret Key: **AWS_ACCESS_KEY_ID**



c. Repeat Steps 3a and 3b for each set of these Kubernetes secrets:

- Environment Variable: **AWS_SECRET_ACCESS_KEY**
  - Secret Name: **aws-connection-my-storage**
  - Secret Key: **AWS_SECRET_ACCESS_KEY`**
- Environment Variable: **AWS_S3_ENDPOINT**
  - Secret Name: **aws-connection-my-storage**
  - Secret Key: **AWS_S3_ENDPOINT**
- Environment Variable: **AWS_DEFAULT_REGION**
  - Secret Name: **aws-connection-my-storage**
  - Secret Key: **AWS_DEFAULT_REGION**
- Environment Variable: **AWS_S3_BUCKET**
  - Secret Name: **aws-connection-my-storage**
  - Secret Key: **AWS_S3_BUCKET**

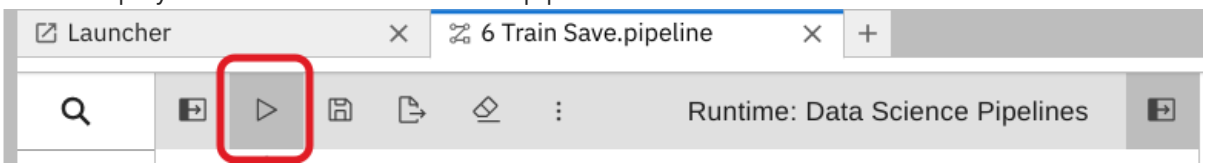
3. **Save** and **Rename** the **.pipeline** file.

5.1.6. Run the Pipeline

Upload the pipeline on the cluster itself and run it. You can do so directly from the pipeline editor. You can use your own newly created pipeline for this or **6 Train Save.pipeline**.

Procedure

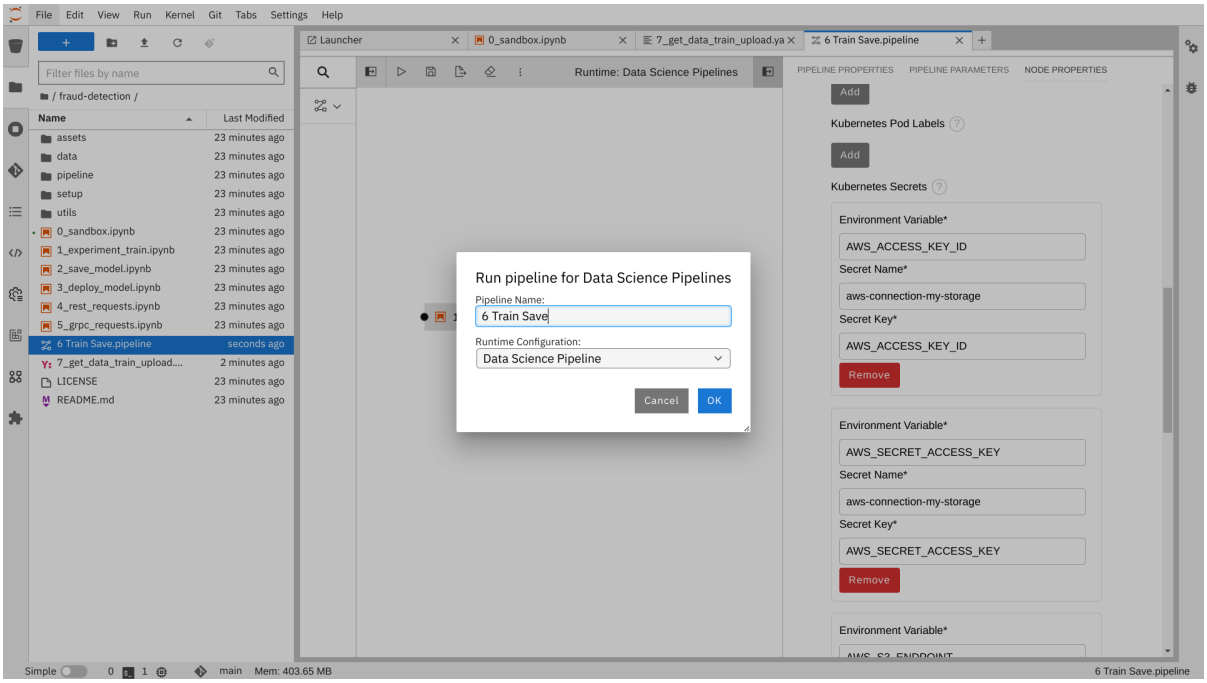
1. Click the play button in the toolbar of the pipeline editor.



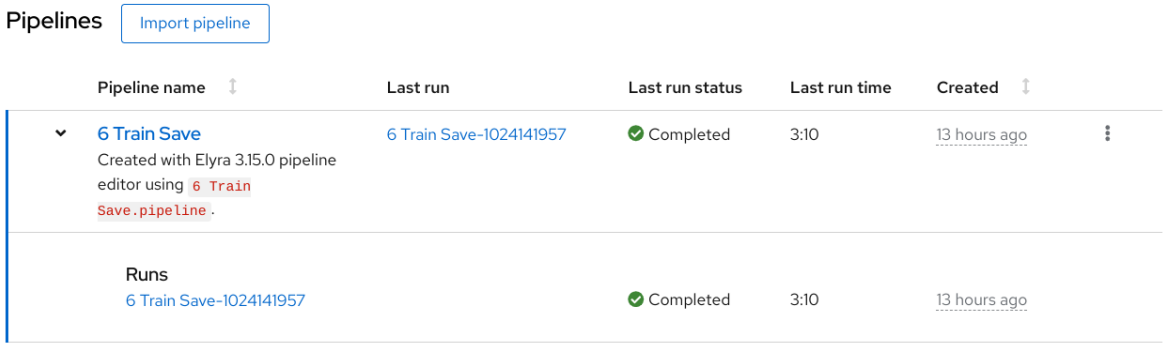
2. Enter a name for your pipeline.
3. Verify the **Runtime Configuration:** is set to **Data Science Pipeline**.
4. Click **OK**.

NOTE

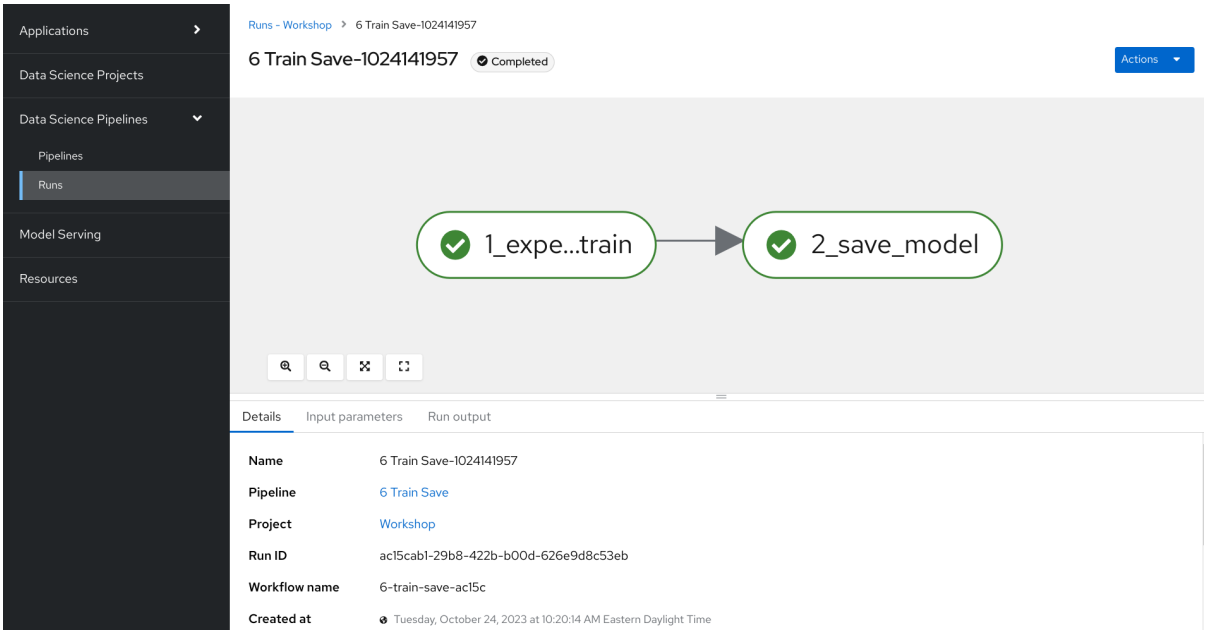
If **Data Science Pipeline** is not available as a runtime configuration, you may have created your notebook before the pipeline server was available. You can restart your notebook after the pipeline server has been created in your data science project.



5. Return to your data science project and expand the newly created pipeline.



6. Click the pipeline or the pipeline run and then view the pipeline run in progress.



The result should be a **models/fraud/model.onnx** file in your S3 bucket which you can serve, just like you did manually in the Preparing a model for deployment section.

Next step

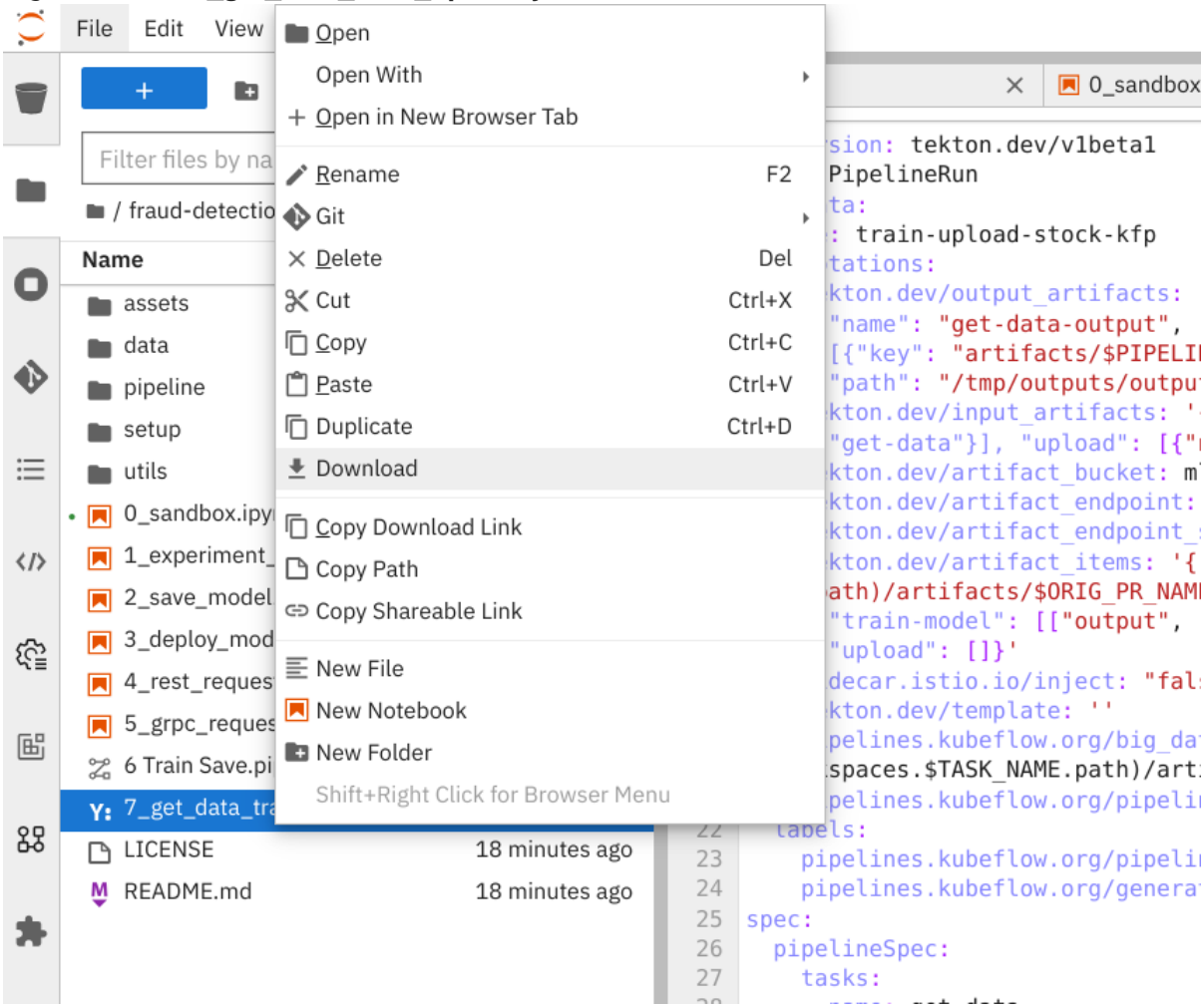
(optional) Automating workflows with data science pipelines

5.2. RUNNING A DATA SCIENCE PIPELINE GENERATED FROM PYTHON CODE

In the previous section, you created a simple pipeline by using the GUI pipeline editor. It's often desirable to create pipelines by using code that can be version-controlled and shared with others. The kfp-tekton SDK provides a Python API for creating pipelines. The SDK is available as a Python package that you can install by using the **pip install kfp-tekton~=1.5.9** command. With this package, you can use Python code to create a pipeline and then compile it to Tekton YAML. Then you can import the YAML code into OpenShift AI.

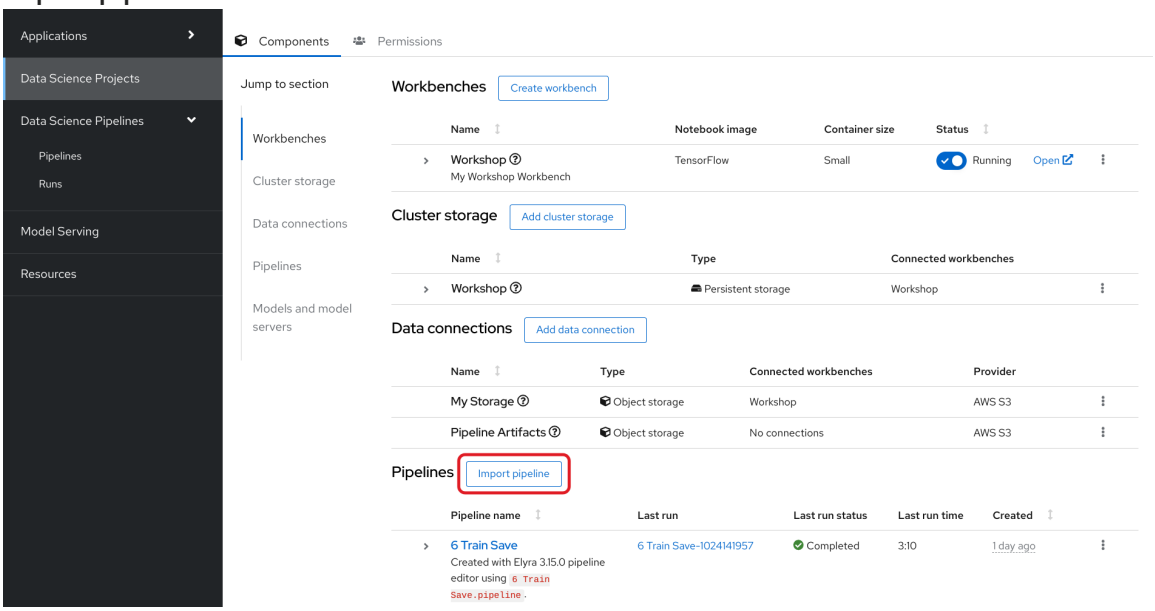
This tutorial does not delve into the details of how to use the SDK. Instead, it provides the files for you to view and upload.

1. Optionally, view the provided Python code in your Jupyter environment by navigating to the **fraud-detection-notebooks** project's **pipeline** directory. It contains the following files:
   - **7_get_data_train_upload.py** is the main pipeline code.
   - **get_data.py**, **train_model.py**, and **upload.py** are the three components of the pipeline.
   - **build.sh** is a script that builds the pipeline and creates the YAML file. The generated **7_get_data_train_upload.yaml** file is located in the **fraud-detection-notebooks** directory.
2. Right-click the **7_get_data_train_upload.yaml** file and then click **Download**.



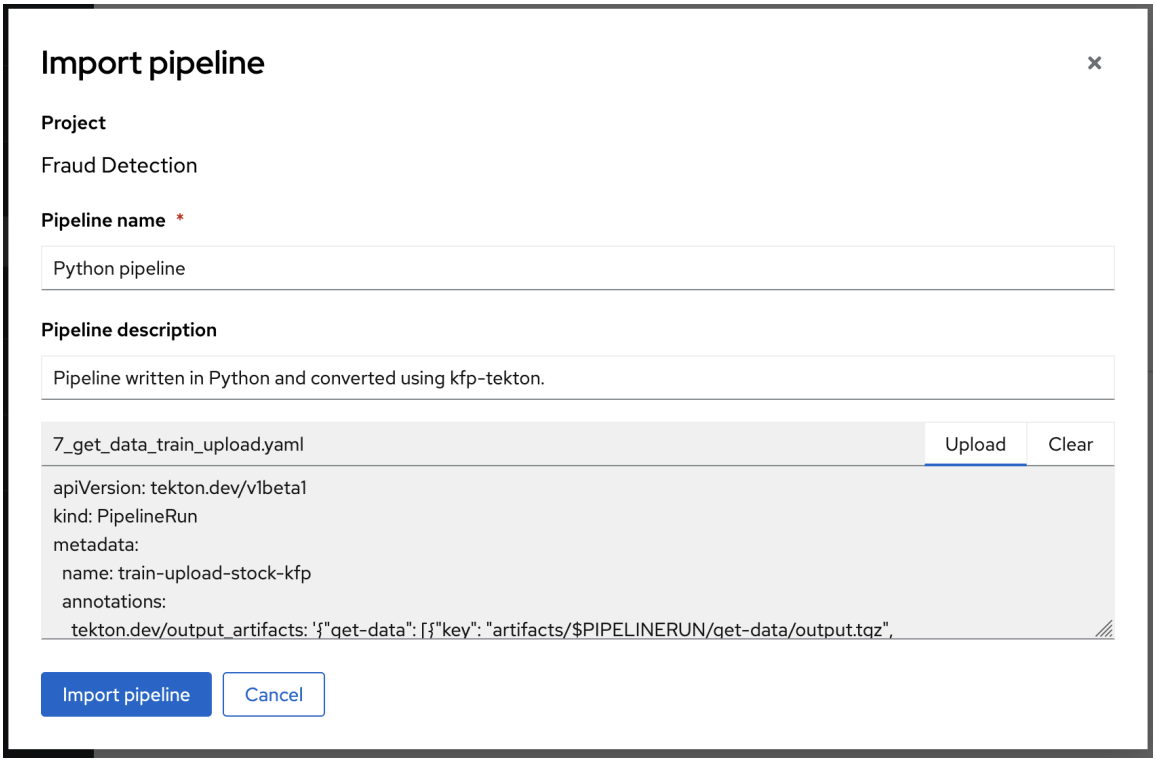
3. Upload the **7_get_data_train_upload.yaml** file to OpenShift AI.

a. In the OpenShift AI dashboard, navigate to your data science project page and then click **Import pipeline**.



b. Enter values for **Pipeline name** and **Pipeline description**.

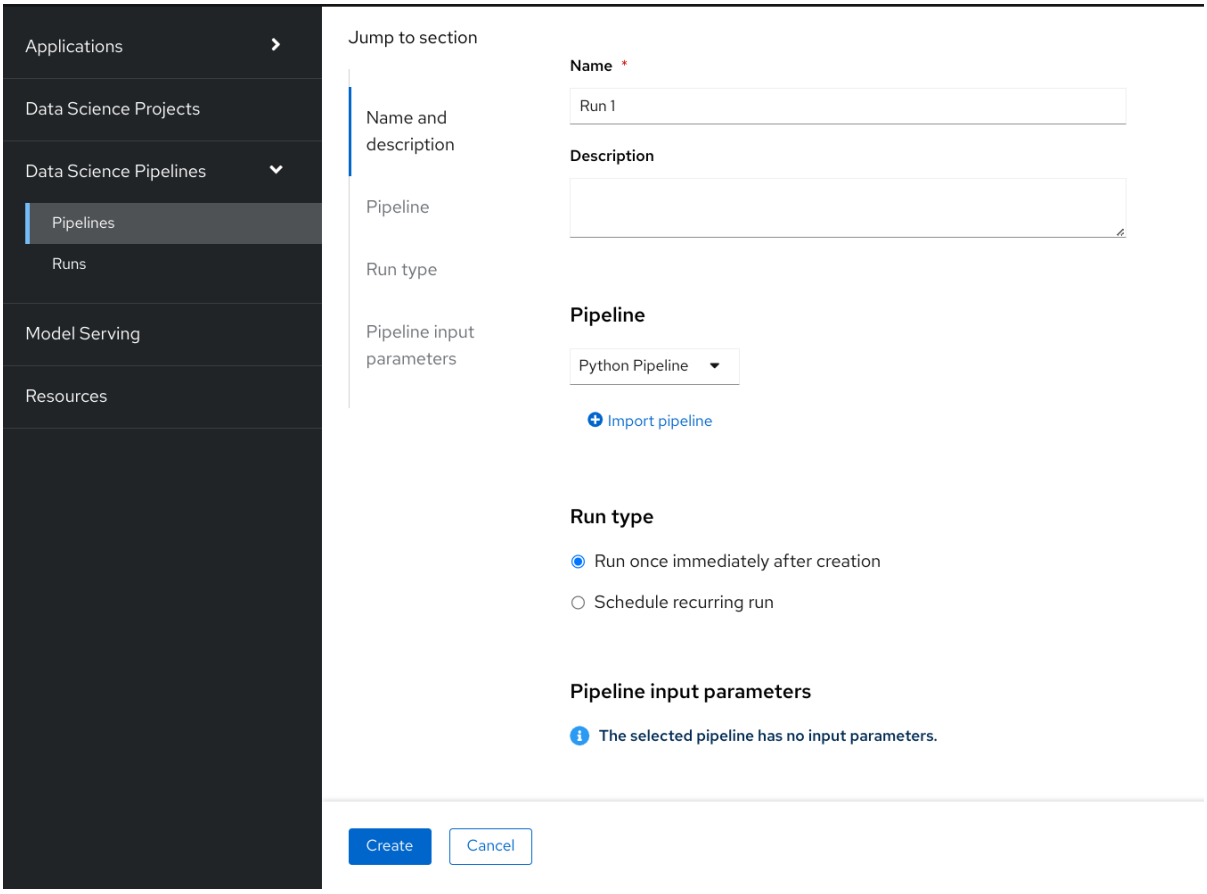
c. Click **Upload** and then select **7_get_data_train_upload.yaml** from your local files to upload the pipeline.



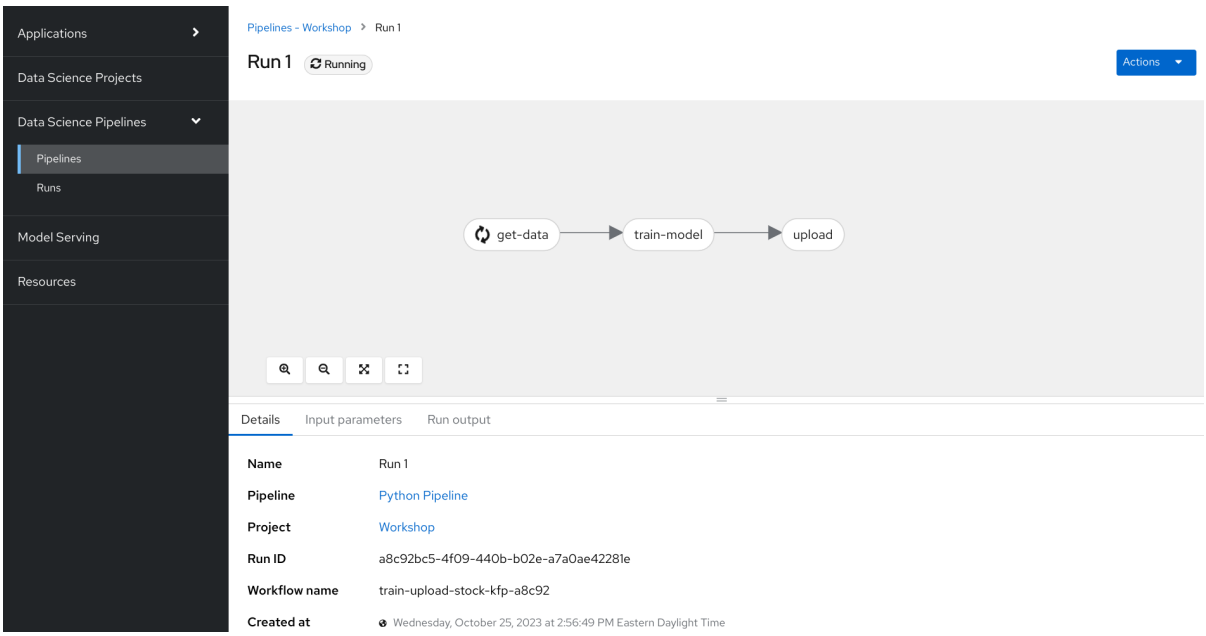
d. Click **Import pipeline** to import and save the pipeline.
The pipeline shows in the list of pipelines.

4. Expand the pipeline item and then click **Create run**.

5. On the **Create run** page, enter a **Name**. You can leave the other fields with their default values.



6. Click **Create** to create the run.
   A new run starts immediately and opens the run details page.



There you have it: a pipeline created in Python that is running in OpenShift AI.

CHAPTER 6. CONCLUSION

Congratulations!

In this tutorial, you learned how to incorporate data science and artificial intelligence (AI) and machine learning (ML) into an OpenShift development workflow.

You used an example fraud detection model and completed the following tasks:

- Explored a pre-trained fraud detection model by using Jupyter Notebooks.
- Deployed the model by using OpenShift AI model serving.
- Integrated the model into a real-time fraud detection application.
- Refined and trained the model by using automated pipelines.