Red Hat OpenShift AI tutorial - Fraud detection example

Use OpenShift AI to train an example model in a Jupyter notebook, deploy the model, and refine the model by using automated pipelines.
Use OpenShift AI to train an example model in a Jupyter notebook, deploy the model, and refine the model by using automated pipelines.
Abstract

Step-by-step guidance for using OpenShift AI to train an example model in a Jupyter notebook, deploy the model, 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  7
   2.3.2. Running a script to install local object storage buckets and create data connections  10
2.4. ENABLING DATA SCIENCE PIPELINES  13

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  23
   3.3.1. Try it  24
3.4. TRAINING A MODEL  26

CHAPTER 4. DEPLOYING AND TESTING A MODEL ........................................ 27
4.1. PREPARING A MODEL FOR DEPLOYMENT  27
4.2. DEPLOYING A MODEL  28
   4.2.1. Deploying a model on a single-model server  28
   4.2.2. Deploying a model on a multi-model server  31
4.3. TESTING THE MODEL API  33

CHAPTER 5. IMPLEMENTING PIPELINES .................................................... 35
5.1. AUTOMATING WORKFLOWS WITH DATA SCIENCE PIPELINES  35
   5.1.1. Create a pipeline  35
   5.1.2. Add nodes to your pipeline  37
   5.1.3. Specify the training file as a dependency  38
   5.1.4. Create and store the ONNX-formatted output file  39
   5.1.5. Configure the data connection to the S3 storage bucket  40
   5.1.6. Run the Pipeline  43
5.2. RUNNING A DATA SCIENCE PIPELINE GENERATED FROM PYTHON CODE  44

CHAPTER 6. CONCLUSION .................................................................... 49
Welcome!

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

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

- Explore a pre-trained fraud detection model by using a Jupyter notebook.
- Deploy the model by using OpenShift AI model serving.
- Refine and train the model by using automated pipelines.

And you do not have 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 monitors 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

If you don’t already have an instance of Red Hat OpenShift AI, see the Red Hat OpenShift AI page on the Red Hat Developer website for information on setting up your environment. There, you can create an account and access the free OpenShift AI Sandbox or you can learn how to install OpenShift AI on your own OpenShift cluster.

IMPORTANT

If your cluster uses self-signed certificates, before you begin the tutorial, your OpenShift AI administrator must add self-signed certificates for OpenShift AI as described in Working with certificates.

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.

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:

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 want 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 prefer.

![Create data science project](image)

**Verification**

You can now see its initial state. Individual tabs provide more information about the project components and project access permissions:

- **Workbenches** are instances of your development and experimentation environment. They typically contain IDEs, such as JupyterLab, RStudio, and Visual Studio Code.

- **Pipelines** contain the data science pipelines that are executed within the project.

- **Models** 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.

- **Cluster storage** is a persistent volume that retains 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.

Permissions define which users and groups can access the project.

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. Click the **Data connections** tab, and then click **Add data connection**.

3. 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.

4. 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**

![Data connection form]

3. Click **Add data connection**

Verification

In the **Data connections** tab for the project, check to see that your data connections are listed.
Next steps

- Configure a pipeline server as described in Enabling data science pipelines
- Create a workbench and select a notebook image as described in Creating a workbench

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

For convenience, run a script (provided in the following procedure) that automatically completes these tasks:

- Creates a Minio instance in your project.
- Creates two storage buckets in that Minio instance.
- Generates a random user id and password for your Minio instance.
- Creates two data connections in your project, one for each bucket and both using the same credentials.
- Installs required network policies for service mesh functionality.

The script is based on a guide for deploying Minio.

IMPORTANT

The Minio-based Object Storage that the script creates is 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 click the ? icon next to the project name. A text box appears with information about the project, including its resource name:

![Fraud Detection Resource Name](image.png)

**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:

```
```

**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.

```yaml
---
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
```
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 steps**

- Configure a pipeline server as described in *Enabling data science pipelines*
- Create a workbench and select a notebook image as described in *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.

In this tutorial, you implement an example pipeline by using the JupyterLab Elyra extension. With Elyra, you can create a visual end-to-end pipeline workflow that can be executed in OpenShift AI.
Prerequisite

- You have installed local object storage buckets and created data connections, as described in Storing data with data connections.

Procedure

1. In the OpenShift AI dashboard, click Data Science Projects and then select Fraud Detection.
2. Click the Pipelines tab.
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 pipeline server**.

7. Wait until the spinner disappears and **No pipelines yet** is displayed.

**IMPORTANT**

You must wait until the pipeline configuration is complete before you continue and create your workbench. If you create your workbench before the pipeline server is ready, your workbench will not be able to submit pipelines to it.

**Verification**

1. Navigate to the **Pipelines** tab for the project.

2. Next to **Import pipeline**, click the action menu (⋮) and then select **View pipeline server configuration**.
An information box opens and displays the object storage connection information for the pipeline server.

**NOTE**

If you have waited more than 5 minutes, and the pipeline server configuration does not complete, you can try to delete the pipeline server and create it again.

You can also ask your OpenShift AI administrator to verify that self-signed certificates are added to your cluster as described in *Working with certificates*.

**Next step**

*Creating a workbench and selecting a notebook image*
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 a workbench you can select a notebook image for your data science work.

Prerequisites

- You created a My Storage data connection as described in Storing data with data connections.
- You configured a pipeline server as described in Enabling data science pipelines.

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 Workbenches tab, and then 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.

   **Notebook image**

   **Image selection** *

   | TensorFlow |
   |

   **Version selection** *

<table>
<thead>
<tr>
<th>2024.1</th>
</tr>
</thead>
</table>

   **A new image version is available. Select the latest image version to use Elyra for pipelines.**
   Hover an option to learn more information about the packages included.

5. Choose a small deployment.

   **Deployment size**

   **Container size**

   | Small |
   |

6. Leave the default environment variables and storage options.
Environment variables

+ Add variable

Cluster storage

- Cluster storage will mount to /

- Create new persistent storage
  This creates storage that is retained when logged out.
  
  **Name** *

  | Fraud Detection |

  **Description**

  | |

  **Persistent storage size**

  - 20 + GiB

- Use existing persistent storage
  This reuses a previously created persistent storage.

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

Data connections

- Use a data connection
  - Create new data connection
  - Use existing data connection

  **Data connection** *

  | My Storage |

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

Create workbench

Verification
In the **Workbenches** tab for the project, 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 valuable local resources such as 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.

   ![Image](image1)

   **Fraud Detection**
   
   My Fraud Detection project

   **Workbenches**

<table>
<thead>
<tr>
<th>Name</th>
<th>Notebook image</th>
<th>Container size</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraud Detection</td>
<td>Tensor Flow</td>
<td>Small</td>
<td>Running</td>
</tr>
</tbody>
</table>

   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:

   ![Image](image2)

   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.
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:

```
In [ ]:
```

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 **CTRL + ENTER** to run a cell or press **SHIFT + ENTER** 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:

```python
In [2]: def print_some_text(entered_text):
    print('This is what you entered:' + entered_text)

my_text = 'Hello world!

print_some_text(my_text)
```

```
This is what you entered:Hello world!
```

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:

   ![Python 3 Notebook](image)

   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.

Data connections

- Use a data connection
  - Create new data connection
  - Use existing data connection

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Connected workbenches</th>
<th>Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>My Storage</td>
<td>Object storage</td>
<td>No connections</td>
<td>AWS S3</td>
</tr>
<tr>
<td>Pipeline Artifacts</td>
<td>Object storage</td>
<td>No connections</td>
<td>AWS S3</td>
</tr>
</tbody>
</table>

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/fraud1/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 offers two options for model serving:

- **Single-model serving** - Each model in the project is deployed on its own model server. This platform is suitable for large models or models that need dedicated resources.

- **Multi-model serving** - All models in the project are deployed on the same model server. This platform is suitable for sharing resources amongst deployed models.

**Note:** For each project, you can specify only one model serving platform. If you want to change to the other model serving platform, you must create a new project.

For this tutorial, since you are only deploying only one model, you can select either serving type. The steps for deploying the fraud detection model depend on the type of model serving platform you select:

- Deploying a model on a single-model server
- Deploying a model on a multi-model server

4.2.1. Deploying a model on a single-model server

OpenShift AI single-model servers host only one model. You create a new model server and deploy your model to it.

**Prerequisite**

- A user with **admin** privileges has enabled the single-model serving platform on your OpenShift cluster.

**Procedure**

1. In the OpenShift AI dashboard, navigate to the project details page and click the **Models** tab.
2. In the Single-model serving platform tile, click Deploy model.

3. In the form, provide the following values:
   
a. For Model Name, type fraud.
   
b. For Serving runtime, select OpenVINO Model Server.
   
c. For Model framework, select onnx-1.
   
d. For Existing data connection select My Storage.
   
e. Type the path that leads to the version folder that contains your model file: models/fraud
   
f. Leave the other fields with the default settings.
4. Click **Deploy**.

**Verification**

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

Testing the model API

4.2.2. Deploying a model on a multi-model server

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

Prerequisite

- A user with admin privileges has enabled the multi-model serving platform on your OpenShift cluster.

Procedure

1. In the OpenShift AI dashboard, navigate to the project details page and click the Models tab.

2. In the Multi-model serving platform tile, click Add model server.

3. In the form, provide the following values:
   a. For Model server name, type a name, for example Model Server.
   b. For Serving runtime, 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, provide the following values:
   
   a. For Model Name, type fraud.
   
   b. For Model framework, select onnx-1.
   
   c. For Existing data connection select My Storage.
   
   d. Type the path that leads to the version folder that contains your model file: models/fraud
e. Leave the other fields with the default settings.

**Deploy model**

Configure properties for deploying your model

**Project**

Fraud detection

**Model name**

fraud

**Model server**

Model Server

**Model framework (name - version)**

onnx - 1

**Model location**

- Existing data connection

- Not seeing what you’re looking for?

**Name**

My Storage

**Path**

models/fraud

Enter a path to a model or folder. This path cannot point to a root folder.

- New data connection

  **Deploy**  **Cancel**

7. Click **Deploy**.

**Verification**

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

<table>
<thead>
<tr>
<th>Models and model servers</th>
<th>Add model server</th>
<th>Deploy model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Server Name</td>
<td>Serving Runtime</td>
<td>Deployed models</td>
</tr>
<tr>
<td>Model Server</td>
<td>OpenVINO Model Server</td>
<td>1</td>
</tr>
</tbody>
</table>

**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.
Procedure

1. In the OpenShift AI dashboard, navigate to the project details page and click the **Models** tab.

2. Take note of the model's Inference endpoint. You need this information when you test the model API.

   ![Models and model servers table]

<table>
<thead>
<tr>
<th>Model name</th>
<th>Serving runtime</th>
<th>Inference endpoint</th>
<th>API protocol</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>fraud</td>
<td>OpenVINO Model Server</td>
<td><a href="https://fraud-fraud-detect">https://fraud-fraud-detect</a> ...</td>
<td>REST</td>
<td></td>
</tr>
</tbody>
</table>

3. Return to the Jupyter environment and try out your new endpoint.
   If you deployed your model with multi-model serving, follow the directions in **3_rest_requests_multi_model.ipynb** to try a REST API call and **4_grpc_requests_multi_model.ipynb** to try a gRPC API call.

   If you deployed your model with single-model serving, follow the directions in **5_rest_requests_single_model.ipynb** to try a REST API call.

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.

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

Note: You can run and use 6 Train Save.pipeline. To explore the pipeline editor, complete the steps in the following procedure to create your own pipeline.

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**.

   **File Dependencies**

   ```
   data/card_transdata.csv
   ```

   ![Include Subdirectories](onnx)

6. Save the pipeline.

5.1.4. Create and store the ONNX-formatted output file

In node 1, the notebook creates the `models/fraud/1/model.onnx` file. In node 2, the notebook uploads that file to the S3 storage bucket. You must set `models/fraud/1/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/1/model.onnx` and then click **Add**.
4. Repeat steps 1-3 for node 2.

5. Save the pipeline.

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 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

**Kubernetes Secrets**

Environment Variable*

AWS_ACCESS_KEY_ID

Secret Name*

aws-connection-my-storage

Secret Key*

AWS_ACCESS_KEY_ID

Remove

Add
c. Repeat Steps 2a and 2b 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 your cluster 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 View runs and then view the pipeline run in progress.

The result should be a `models/fraud/1/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` 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` command. With this package, you can use Python code to create a pipeline and then compile it to YAML format. 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. For your convenience, the output of the **build.sh** script is provided in the **7_get_data_train_upload.yaml** file. The **7_get_data_train_upload.yaml** output file is located in the top-level **fraud-detection** 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. Click the **Pipelines** tab and then click **Import pipeline**.

   ![Pipelines tab](image)

   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.

   ![Import pipeline dialog](image)

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

4. Expand the pipeline item, click the action menu (⋮), and then select **View runs**.
5. Click **Create run**.

6. On the **Create run** page, provide the following values:
   
   a. For **Name**, type any name, for example **Run 1**.
   
   b. For **Pipeline**, select the pipeline that you uploaded. You can leave the other fields with their default values.
Create run
Create a run from a pipeline.

7. Click **Create** to create the run.
A new run starts immediately. The **Details** page shows a pipeline created in Python that is running in OpenShift AI.
Run 1

Details
- Name: Run 1
- Pipeline version: Python Pipeline
- Project: Fraud Detection 2
- Run ID: 1052c7bf-3802-425e-b412-079ac04983f
- Workflow name: Run 1
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 a Jupyter notebook.
- Deployed the model by using OpenShift AI model serving.
- Refined and trained the model by using automated pipelines.