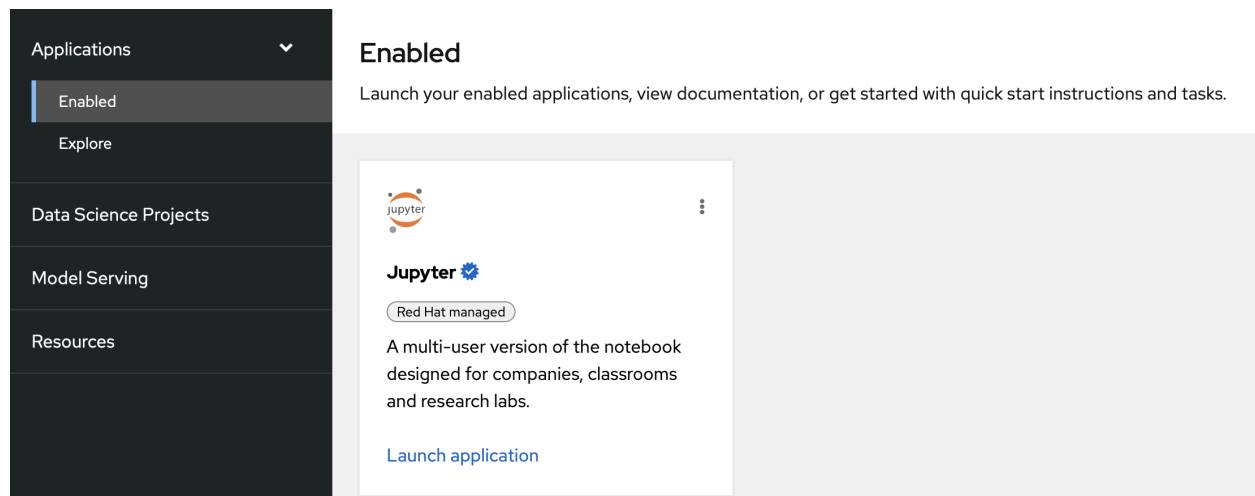# Create projects

Introduction

In RHOAI, a data science project is the preferred way to organize resources when working on an AI/ML application. Similarly to how you use projects in OpenShift for other workloads, you can use and should use data science projects to organize the different elements that you need for your AI applications, such as workbenches, model servers, or persistent storage.

## Exercise: 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 do data science as part of a 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 shows a list of any existing projects that you have access to. From this page, you can select an existing project (if any) or create a new one.
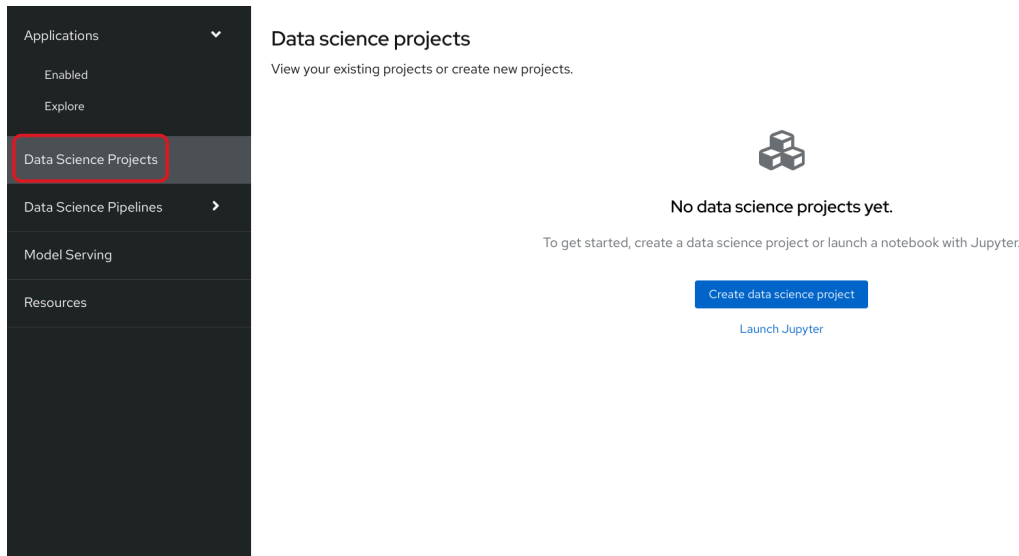
2.

3. Click Create data science project.

4. Enter a display name and description.
   - Based on the display name, a resource name is automatically generated
   - Beware that you are able to update the display name later on, but not the resource name.

The project name should not be userX like in the screenshot. X should match your user id/number. This way all attendants will have a unique project name.

○

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

6.

- ○ 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.

Deleting Data Science Projects

NOTE:
You don't need to remove the project, this is just information on how you would go about it.

To delete a data science project, navigate to the Data science projects page. Then, locate the project to be deleted and click its ⋮ button, then click Delete project. Finally, type the project name to confirm the deletion.
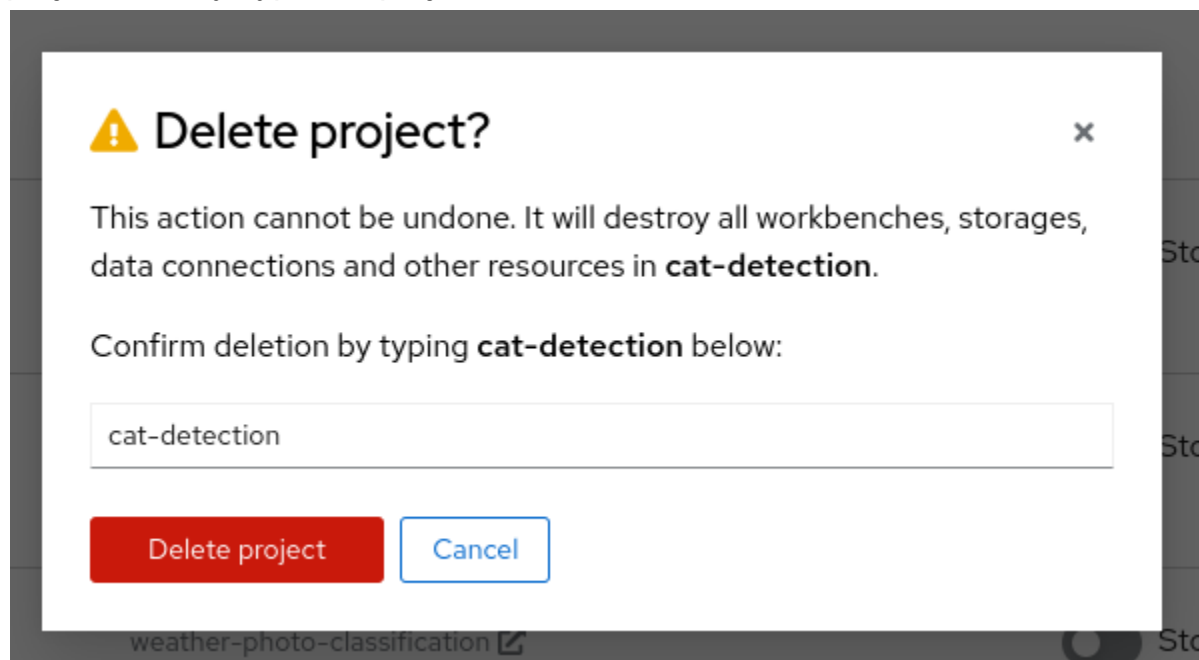
Workbenches

Introduction

In RHOAI, a workbench is a containerized working environment that runs as any other application in OpenShift, as a pod. Workbenches typically include a collection of common AI/ML libraries and a JupyterLab server. RHOAI exposes the JupyterLab URL through an OpenShift route so that you can use and interact with the workbench via the web browser.
Data science projects can include more than one workbench. This is useful when you need to run different experiments that belong to the same project, but with different tooling. For example, you might want to train a simple model with Scikit-learn and another model with PyTorch. Or you might have a workbench only for data exploration and another workbench for model training.

> Thanks to being connected to persistent storage, workbenches are stateful, meaning that you can stop and restart them without losing your work. Restarting them will clean up everything that was not stored in the persistent storage though, such as the installed python packages.

Exercise: Getting Started with Workbenches

Choose main workbench settings

Navigate to the project dashboard page and click the Create workbench button.
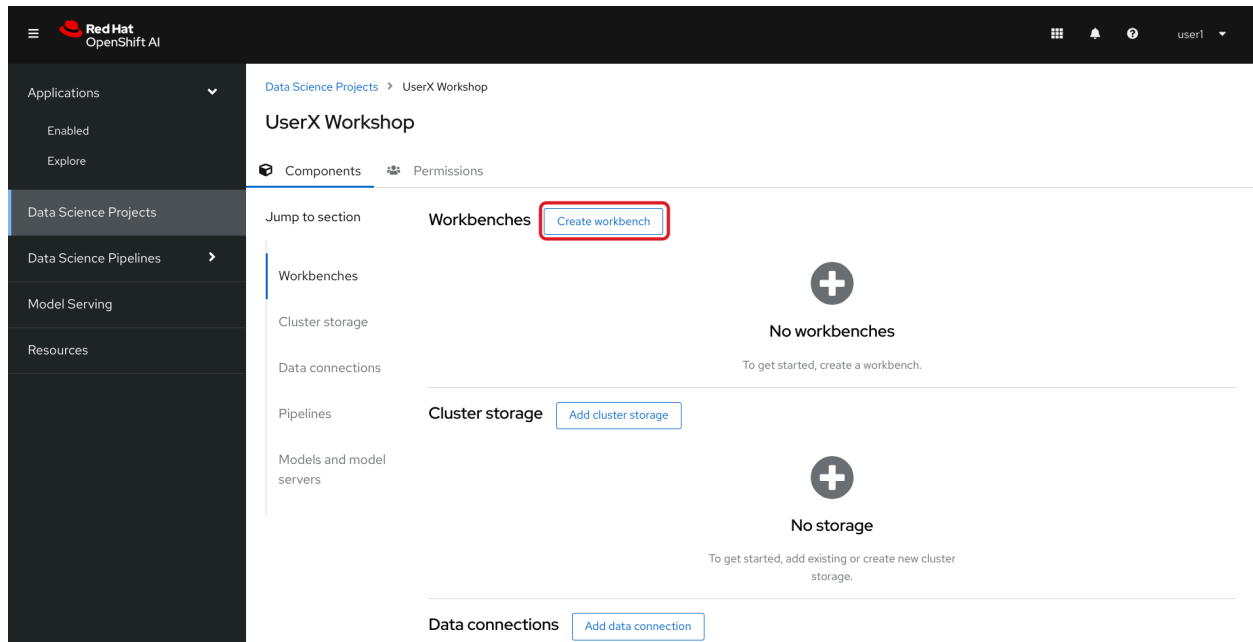
Fill out the name and description.

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 lab so we will pick that.

The requirements of your project dictate the image that best suits your needs. For example, if you plan to use TensorFlow, then you might want to select the TensorFlow image. You can view more details about the selected image by clicking View package information.

Notebook image

Image selection *

| TensorFlow | ▼ |

Version selection *

| 2023.2 (Recommended) | ▼ |

Hover an option to learn more information about the packages included.

❷ View package information

Choose Container size Custom Size - Small

Deployment size

Container size

| Custom Size - Small | ▼ |

There are usually many available sizes. We shortened the list to a single size for this lab

Accelerator

If you have a hardware accelerator enabled in your cluster and have set up so that RHOAI can see it, you will see an additional dropdown called "Accelerator". We won't need accelerated hardware for this workshop, so we will leave it at None.

Define Environment Variables

Environment variables are the recommended way to inject configuration into software applications. In this way, the application, or in this case the RHOAI workbench, is decoupled from the working environment.

Scroll down to the Environment variables section and click Add variable. Choose whether you want to create a ConfigMap or a Secret. You can define multiple key/value pairs or upload a file.

Environment variables

Config Map ▼ ⊖

Key / value ▼

Key *

DATA_FILE ⊖

Value *

data.csv

⊕ Add another key / value pair

⊕ Add more variables

Each ConfigMap or Secret can contain multiple key/value items. To add more items to a ConfigMap or a Secret, click Add another key / value pair. Alternatively, you can upload a file that includes multiple key/value pairs.

Add Storage

Workbenches require persistent storage to ensure that your progress is not lost when you stop or remove the workbench.
Scroll down to the Cluster storage section. Verify that RHOAI automatically selects the Create new persistent storage option. This is the default behavior. RHOAI creates a dedicated persistent storage for each workbench and mounts the storage in the root directory of the container.
Alternatively, you can select an already existing storage. The existing storage must not be associated with any other workbench.
You can adjust the storage size, which by default is 20 GiB.

Environment variables

➕ Add variable

Cluster storage

ℹ️ **Cluster storage will mount to /**

🔘 Create new persistent storage
This creates storage that is retained when logged out.

Name *

| Workshop |

Description

| |

Persistent storage size

| − | 20 | + | Gi ▼ |

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

> You can also create persistent storage items from the data science project page, and leave the storage unassigned. This is useful to create persistent storage that you plan to mount in workbench directories different from /.

> You cannot decrease the storage size of storage that is associated with a workbench.

> You cannot delete a storage item if it is assigned to a workbench. The storage must be unassigned if you want to delete it.

Data Connections

We will leave the data connection empty for now and come back to it later.

Data connections

☐ Use a data connection

Create the Workbench

Scroll down to the bottom of the page.
The Data connections section lets you inject S3 connection parameters into your workbench. Leave this section unchanged. Data connections are covered later in the course.
Click Create workbench. RHOAI triggers the creation of the workbench and redirects you to the data science project page. While RHOAI is provisioning the workbench, the workbench displays with the Starting… status.

Status

✓◯ Starting...
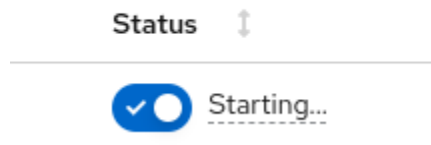
If the workbench creation is successful, then you should see the workbench running in the project dashboard. The dashboard should also display the persistent storage associated with the workbench.

Workbenches [Create workbench]

Name	Notebook image	Container size	Status		
› Workshop ⓘ My Workshop Workbench	TensorFlow	Small	✓◯ Running	Open ↗	⋮

Limits

If your memory and/or GPU requirements are high, then RHOAI might not be able to allocate the requested resources in the workbench. You might see a message similar to the following:

In this case, try to decrease your resource requirements by editing the workbench, or contact your RHOAI administrator.

Inspect the Workbench

In the data science project page, click the > icon to view more details about the workbench. The card that slides down displays the workbench details, including storage usage. You can use the Add storage button to add more storage mount points to the workbench.

Exercise: Use the Workbench

Open the Workbench

Click Open.
In the browser tab that opens, enter your credentials to authenticate to the workbench's JupyterLab. Next, allow the selected permissions to access JupyterLab. Click Allow selected permissions.

Verify that you see the JupyterLab application:
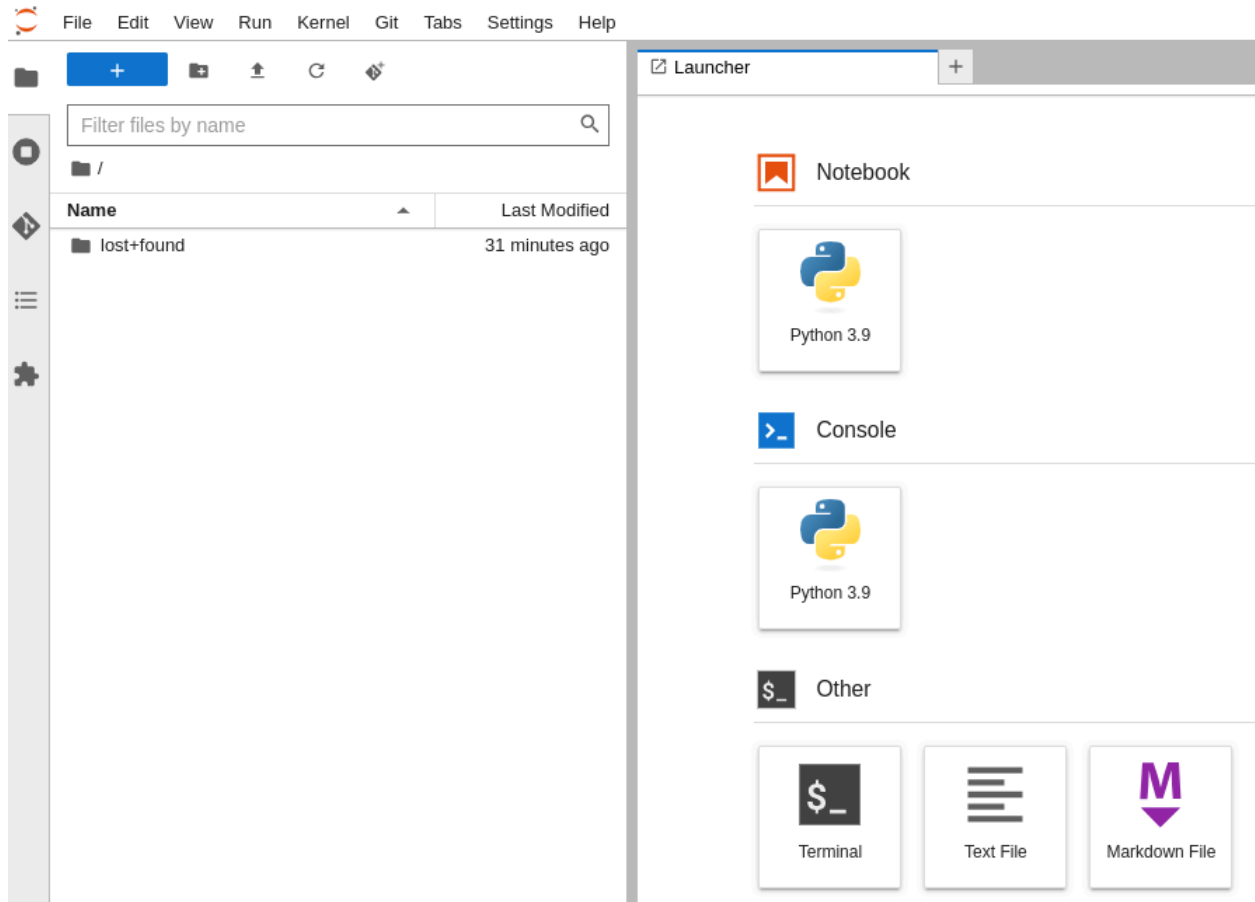
In the right pane, click the Terminal item to open a new terminal. In the terminal, verify that the `DATA_FILE` environment variable has been injected into the workbench:

```
(app-root) (app-root) echo $DATA_FILE
data.csv
```

Use Git

Version control systems, and in particular, Git, foster collaboration in the following ways:

- Multiple users can work on the same code base at the same time.
- Users can work in features or fixes by using branches, to reduce conflicts with the main development branch.
- Users can propose, review, and discuss changes by using pull requests.

The collaboration and collective experience of developers, companies, and open sources communities has resulted in sets of good practices and rules called *Git workflows*. These workflows define how teams should work together and coordinate

by using Git features, such as branches and pull requests. Some of these workflows are:

- GitHub Flow
- Git Flow
- Trunk-based Development

To bring the content for this lab into your workbench, follow these steps:

- On the toolbar, click the Git Clone icon:
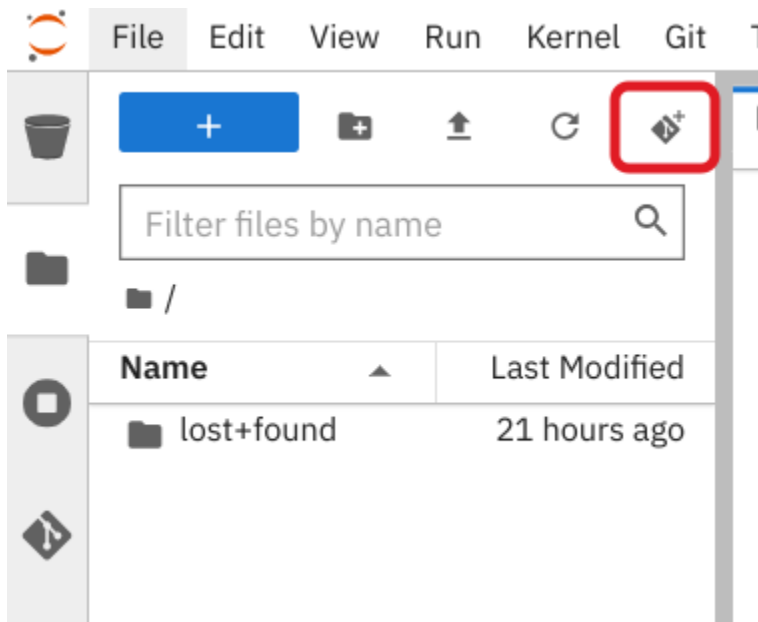
-
- Enter the following lab Git https URL:
- `https://github.com/rh-aiservices-bu/rhoai-training.git`
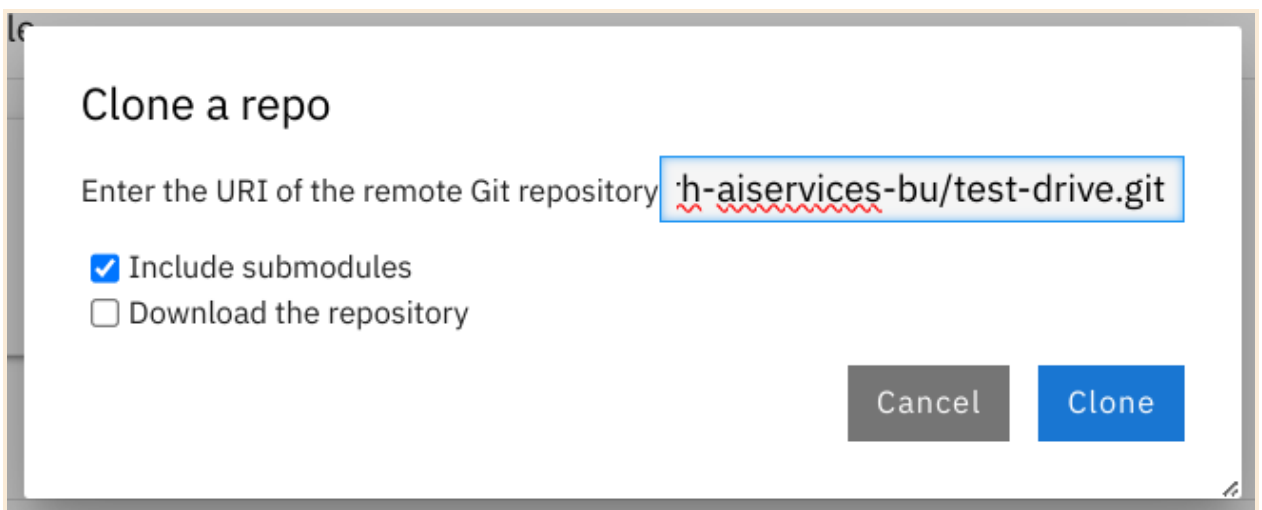
-
  a. Check the Include submodules option.
  b. Click CLONE.

Train a model

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

In your notebook environment, open the file `1_experiment_train.ipynb`, and follow the instructions directly in the notebook.

The instructions guide you through some simple data exploration, experimentation, and model training tasks.
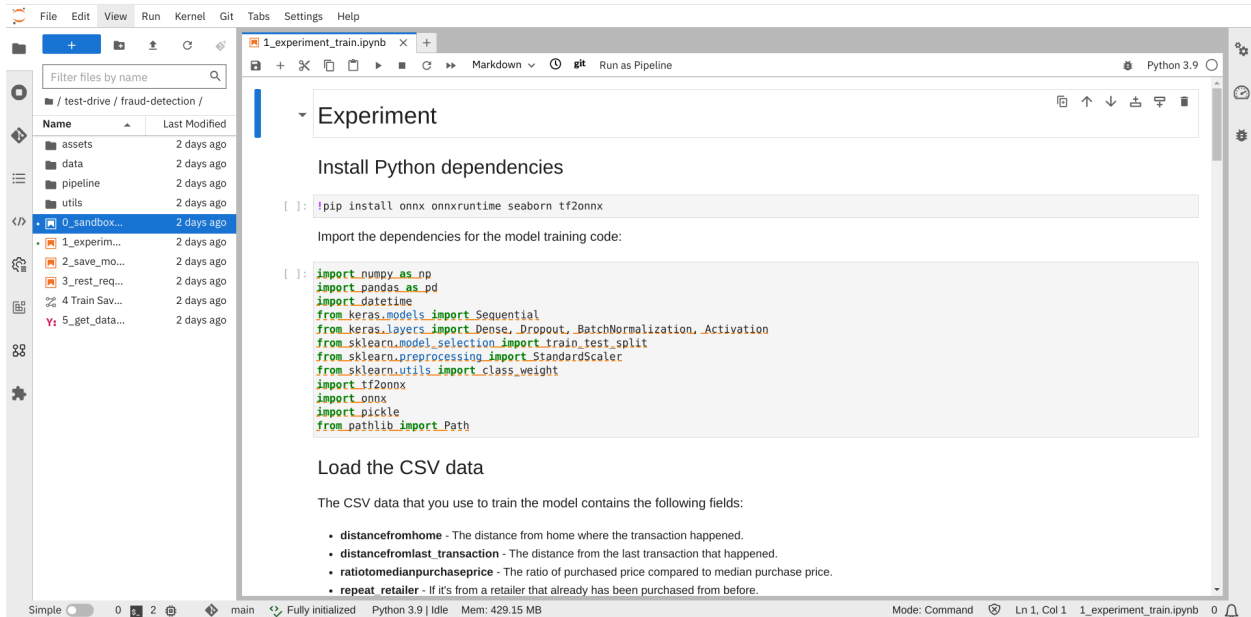
Data connections

Introduction

In data science, AI, and ML environments, the availability of data is a key factor for the success of a project. For simple projects, where a relatively small amount of data is used, you can load data by uploading files to the workbench via JupyterLab, or by making database queries from your notebooks.

More complex projects, however, involve larger datasets. Depending on the dataset size, uploading files or querying databases might not be a feasible solution. Storing the data directly in your code repository is not recommended

either, due to the overload that this can cause in your repository. In fact, many source control systems include file size limits.

A better approach is to store the data in a dedicated storage system, and download the data into your workbench when you need it, or run jobs over the data.

Similarly, a dedicated storage system is also required to store the artifacts that the training phase produces, i.e. the trained models. Trained models can result in large files when you export them, so the recommendation is to store them in a dedicated storage system.

Data Connections are Secrets

RHOAI introduces the concept of data connection to define a set of configuration values that you can use to connect your workbench to storage systems. You can use the external storage to store your datasets and your trained models.

> Do not confuse data connections with cluster storage.
> Cluster storage provides persistent storage for your workbench and is used to persist your progress in the workbench.
> Data connections are a set of configuration values that you can use to connect to storage systems.

RHOAI creates data connections in OpenShift as Secret objects. RHOAI stores these secrets in the OpenShift namespace that corresponds to your data science project. The secret associated with each data connection includes these key/value pairs:

- AWS_ACCESS_KEY_ID
- AWS_DEFAULT_REGION
- AWS_S3_BUCKET
- AWS_S3_ENDPOINT
- AWS_SECRET_ACCESS_KEY

You can set these values by using a web form in the RHOAI dashboard, as you will learn in the exercise.

Although the naming of these variables suggests the use of AWS S3, these values are only environment variables, so you decide how you use them in your notebooks. You can use them to connect to systems such as OpenShift Data Foundation, IBM Cloud Object Storage, and AWS S3.

Using the Data Connection Environment Variables

When you associate a data connection to a workbench, RHOAI injects the key/value pairs of the data connection as environment variables into the workbench container. This makes it so that you can reach them from within your workbench environment.

> RHOAI does not open a connection from the workbench to AWS S3. You are responsible for reading and using the environment variables of the data connection in the notebook.

Typically, you should read the data connection variables and then use them to configure the connection to the storage system. For S3 storage, this can be done with the boto3 library, as you will learn in the following exercise.
After you are connected to S3, then you can start downloading the files required for data exploration or model training.
Similarly, you should store your trained models in the S3 bucket that corresponds to the data connection. In fact, storing your models in the data connection enables you to use Model Serving. Model Serving downloads the model files from S3 by using a data connection.

Exercise: Getting Started with Data Connections

For convenience, we are providing a script that will:
- create a MinIO instance in your project
- create 2 buckets in that MinIO instance
- generate a random user id and password for your MinIO instance
- create 2 Data Connections in your data science project
  - one for each bucket
  - both using the same credentials

This script is based on a quick tutorial for installing MinIO (available [here](#)).
In addition it will install some necessary network policies for service mesh functionality.

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:
- ● Go to the OpenShift AI dashboard
- ● Select Data Science Projects
- ● Click the ? icon next to the project name
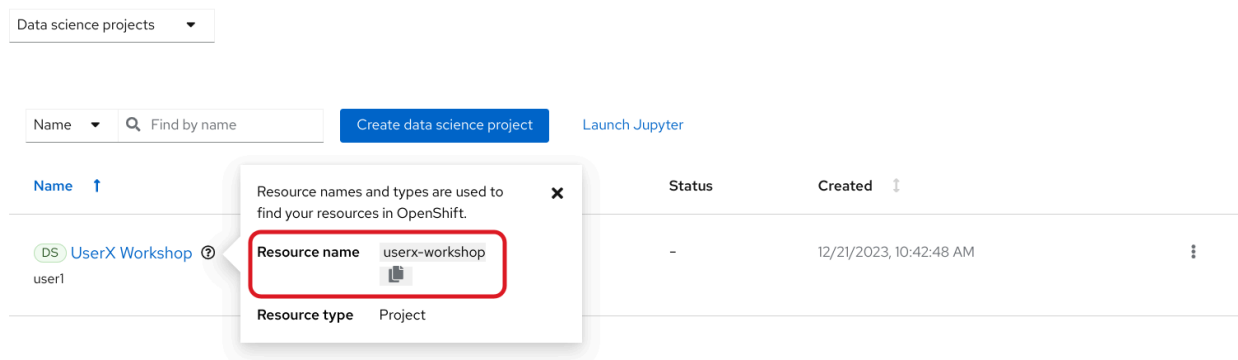
A text box appears with information about the project, including its resource name:

Make sure to note that resource name.

Installing MinIO and setting up Data Connections

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

2.

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

4.

5. Select your project from the list of projects.

6.

7. Verify that you selected the correct project. (In the next step you will import the YAML)
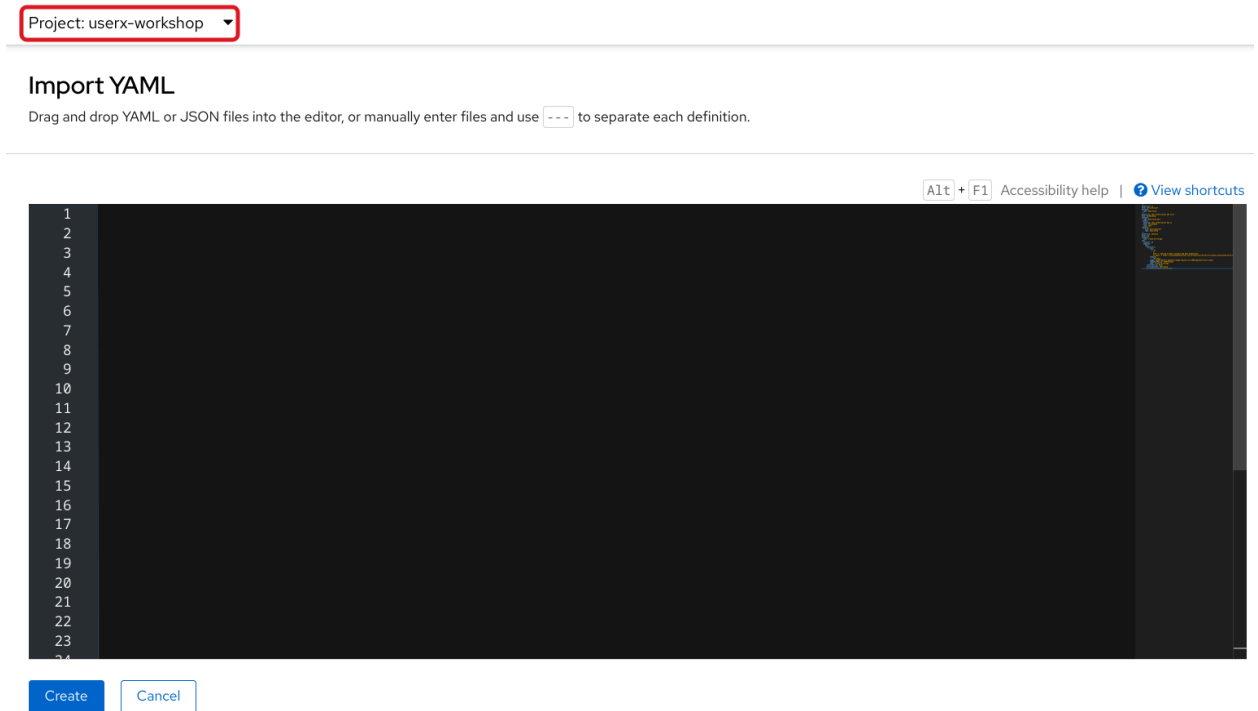
8.

9. Copy the following code and paste it into the Import YAML editor:

```
---
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://raw.githubusercontent.com/rh-aiservices-bu/test-drive/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
```

10.   Click Create.

Verification

You should see a "Resources successfully created" message and the following
resources listed:
- demo-setup
- demo-setup-edit
- create s3-storage

After a while you should see two Data Connections in your Data Science Project,
one called My Storage and one called Pipeline Artifacts.

Edit your workbench to include a Data Connection

From a data science project page, click the workbench ⋮ button, then click Edit workbench.
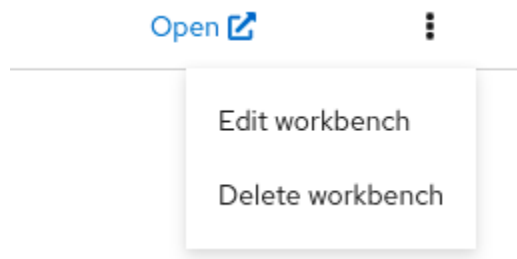
> Make sure that you save your work at JupyterLab before updating a workbench.
> When you edit a workbench, RHOAI restarts the workbench, so any unsaved work will be lost.

> You cannot edit a workbench while the workbench is starting or stopping.

Scroll down to the Data connections section and select Use existing data connection and choose My Storage.

Data connections

☑ Use a data connection

    ○ Create new data connection

    ◉ Use existing data connection

        Data connection *

        | My Storage   ⊗ ▾ |

Save the workbench. In the data science project page, notice the newly created data connection, which is associated with the workbench.

Data connections | Add data connection |

Name ↕	Type	Connected workbenches
My Storage ⑦	📦 Object storage	Workshop

> The association between data connections and workbenches follows a 1 to N schema:
> - The same data connection can be used in multiple workbenches.
> - A workbench can have only one data connection.

Edit a Data connection

Next, click the ⋮ button of the data connection, then click Edit data connection. Note that you can assign more workbenches to the same data connection.

Edit data connection ✕

Name *

my-first-data-connection

Access key *

1111111111111111111111111111111111

Secret key *

•••••••••••••••••••••••••••••••••••••   👁‍🗨

Endpoint *

https://s3.eu-de.cloud-object-storage.appdomain.cloud

Region

eu-de

Bucket

my-bucket

Connected workbench

| my-first-workbench ✕ | my-second-workbench ✕ | Select a workbench to connect | ⊗ ▾ |

Connect to workbenches that do not already have a data connection

> ℹ️ **Unsaved work will be lost**
>
> Running workbenches will restart upon updating. To avoid losing your work, save any recent data in the following running workbenches: my-second-workbench.

[Update data connection]   [Cancel]

You do not need to make any changes.

> You can use the data science project page to create new data connections and assign them to existing workspaces.
> You can also use this page to delete data connections. Deleting a data connection that is assigned to a workbench results in a workbench restart.

Using the Data Connection in a Workbench

1. Enter your Workbench and navigate to the **rhoai-training/lab-material/data-connections** folder.
2. Open the exercise.ipynb notebook.
3. Follow the instructions in the notebook. Click the first cell, then press Shift+Enter to execute the cell and move to the next one. Next, execute and review the rest of the cells. Keep pressing Shift+Enter until you reach the bottom.

Serve models

Introduction

Generally speaking, an organization decides to train a model, because the organization wishes to empower the applications portfolio with inferences made by such a model. In this way, the given known data points of a given event, can produce an inference, guess, or prediction of a topic of interest.

In this course, you will learn how to leverage Red Hat OpenShift Data Science to serve machine learning models.

How do we deliver a model to an inference engine, or server, so that, when the server receives a request from any of the applications in the organization's portfolio, the inference engine can reply with a prediction made based on the model that we have previously worked hard to train?

Machine learning models must be deployed in a production environment to process real-time data and handle the problem they were designed to solve. Deploying a model in a production environment means that the model that has been trained, and exported to a model file format, needs to be imported in a runtime engine, and exposed for applications to consume.

Consuming from a model, means that software applications will use a communication method, often REST/HTTP to send a prompt request to a server, such a server will fire a request to the model and provide a response. It is evident that the server processing the request, and providing a response based on the model needs to have access to such a model.

RHOAI Model Serving Runtimes

In Red Hat OpenShift AI you do not need to manually create serving runtimes. By default, Red Hat OpenShift AI includes multiple pre-configured model serving runtimes, which can load, execute, and expose models trained with TensorFlow and PyTorch.

In order to leverage the benefits of the model server, you must:
1. Create a Data Connection to the S3 containing the model
2. Export the model in a format compatible with one of the available RHOAI runtimes.
3. Upload the model to an S3
4. Create or use one of the available serving runtimes in a Model Server configuration that specifies the size and resources to use while setting up an inference engine.
5. Start a model server instance to publish your model for consumption

While publishing this model server instance, the configurations will allow you to define how applications securely connect to your model server to request for predictions, and the resources that it can provide.

Exercise: Preparing a model for deployment

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

To prepare it for deployment, you must copy the model from your workbench to your S3-compatible object storage.

You can use the data connection that you created in the 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.

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.
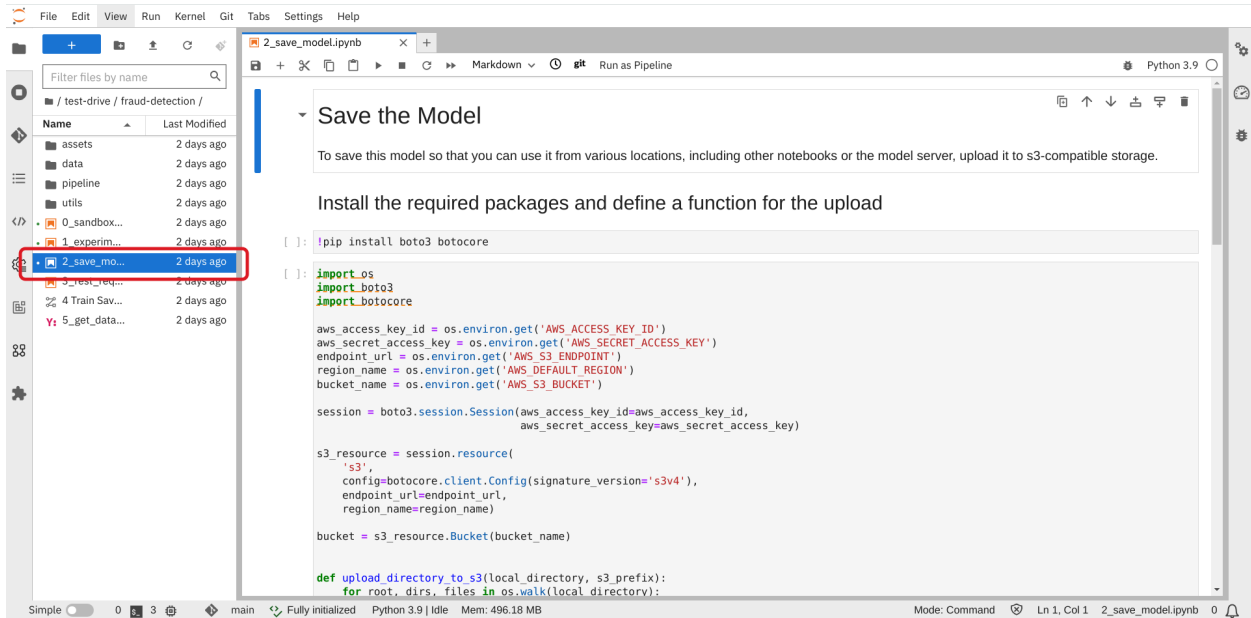
Verification

When you have completed the notebook instructions, the models/fraud/1/model.onnx file is in your object storage and it is ready for your model server to use.

Exercise: 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 now has two options for model serving:

- multi-model serving-platform
- single-model serving-platform

Review the descriptions available in the interface.

Procedure

1. In the OpenShift AI dashboard
2. Navigate to Models and model servers.
3. Under Single model serving platform, click Deploy model.
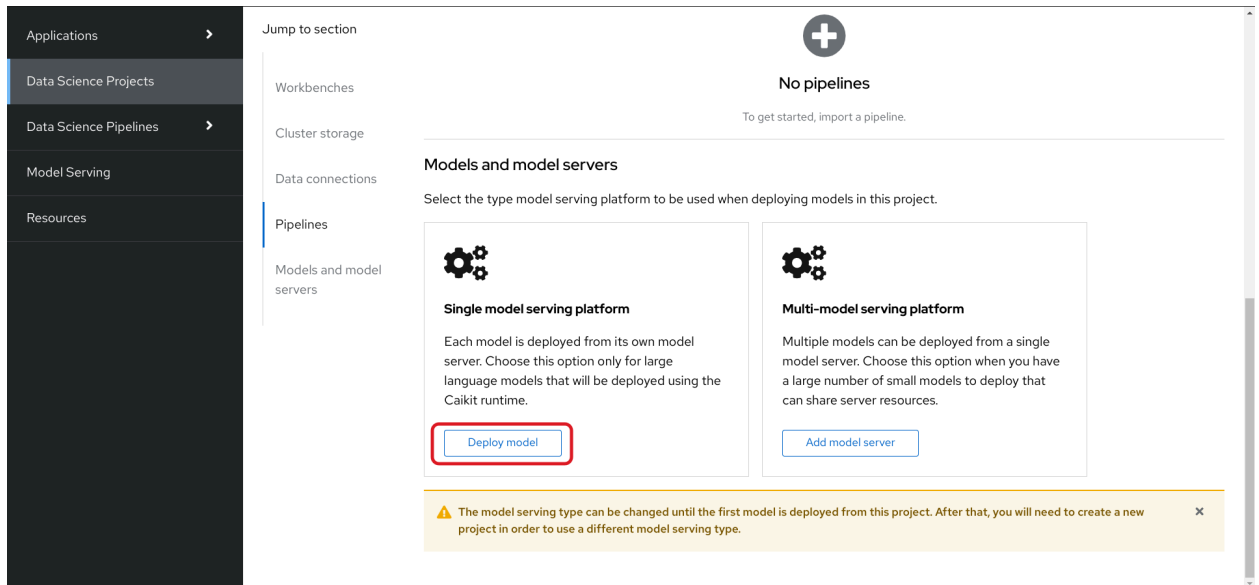
4.

5. In the form:
   - Fill out the Model Name with the value fraud.
   - Select the Serving runtime, OpenVINO Model Server.
   - Select the Model framework, onnx - 1.
   - Set the Model server replicas to 1.
   - Select the Model Server size, Lab Size - Small.
   - Select the Existing data connection: My Storage
   - Enter the path to your uploaded model: models/fraud.

Deploy model

Configure properties for deploying your model

Project

UserX Workshop

Model name *

fraud

Serving runtime *

OpenVINO Model Server ▼

Model framework (name - version) *

onnx - 1 ▼

Model server replicas

Number of model server replicas to deploy

− 1 +

Compute resources per replica

Model server size

Lab Custom Small ▼

Accelerator ⊙

None ▼

Model location

⦿ Existing data connection

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

> The path does not include 1/model.onnx. The OpenVINO model server expects the format to include the integer version as part of the subpath.

6. Click Deploy.
7. Wait for the model to deploy and for the Status to show a green checkmark.

Models and model servers	Deploy model			Single model serving enabled
Model name ↑	**Serving runtime**	**Inference endpoint**	**Status**	
› fraud ⓘ	OpenVINO Model Server	https://fraud-predictor-userx-workshop.app... 🗐	✅	⋮

8.
9. This might take a little while if the cluster is particularly busy, but should be less than 2 minutes.

Model Serving Request Body

To make HTTP requests to a deployed model you must use a specific request body format. The basic format of the input data is as follows:

```Python
{
 "inputs": [{
  "name" : "input",
  "shape" : [2,3],
  "datatype" : "INT64",
  "data" : [[34, 54, 65], [4, 12, 21]]
 }]
}
```

1. The name of the input tensor. The data scientist that creates the model must provide you with this value.
2. The shape of the input tensor.
3. The data type of the input tensor.
4. The tensor contents provided as a JSON array.

You will see two examples of how to do that below.

Exercise: Testing the model API

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

Procedure

1. In the Data Science dashboard
2. Navigate to the project details page
3. Scroll down to the Models and model servers section.
4. Take note of the model's Inference endpoint. You need this information when you test the model API.

Return to the Jupyter environment and try out your new endpoint in the terminal with this curl command. Remember to replace $ROUTE with the route you copied.

```
Unset
curl -k $ROUTE/v2/models/fraud/infer  -X POST \
--data '{"inputs" : [{"name" : "dense_input","shape" : [ 1, 5 ],"datatype" :
"FP32","data" : [0.3, 2, 1.0, 0.0, 0.0]}]}'
```

To make a request in Python you can use the request library, try REST API calls in **3_rest_requests.ipynb**

/ test-drive / 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
utils	2 hours ago
0_sandbox...	2 hours ago
1_experim...	2 hours ago
2_save_mo...	2 hours ago
3_rest_req...	a minute ago
4 Train Sav...	2 hours ago
5_get_data...	2 hours ago
Train and S...	6 minutes ago

Code ∨   Run as Pipeline   Python 3.9 ○

REST Inference

Setup

Change that following variable settings match your deployed model's *Inference endpoint*. for example:

```
deployed_model_name = "fraud"
infer_endpoint = "https://fraud-predictor-userx-workshop.apps.clusterx.sandboxx.opentlc.com"
```

[1]:
```
deployed_model_name = "fraud"
infer_endpoint = "https://fraud-predictor-userx-workshop.apps.cluster-6gt6q.sandbox2603.opentlc.com"
infer_url = f"{infer_endpoint}/v2/models/{deployed_model_name}/infer"
```

Request Function

Build and submit the REST request.

Note: You submit the data in the same format that you used for an ONNX inference.

[2]:
```
import requests

def rest_request(data):
    json_data = {
        "inputs": [
            {
                "name": "dense_input",
                "shape": [1, 5],
                "datatype": "FP32",
                "data": data
            }
        ]
```

Simple   0   4   main   Fully initialized   Python 3.9 | Idle   Mem: 695.48 MB   Mode: Command   Ln 1, Col 1   3_rest_requests.ipynb   0