



Red Hat OpenShift Data Science self-managed 2.4

Working on data science projects

Create and collaborate on notebooks in your OpenShift Data Science notebook environment

Red Hat OpenShift Data Science self-managed 2.4 Working on data science projects

Create and collaborate on notebooks in your OpenShift Data Science notebook environment

Legal Notice

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Learn how to create and collaborate on notebooks in your OpenShift Data Science notebook environment.

Table of Contents

| | |
|---|-----------|
| CHAPTER 1. CREATING AND IMPORTING NOTEBOOKS | 5 |
| 1.1. CREATING A NEW NOTEBOOK | 5 |
| 1.1.1. Notebook images for data scientists | 5 |
| 1.2. UPLOADING AN EXISTING NOTEBOOK FILE FROM LOCAL STORAGE | 6 |
| 1.3. UPLOADING AN EXISTING NOTEBOOK FILE FROM A GIT REPOSITORY USING JUPYTERLAB | 7 |
| 1.4. UPLOADING AN EXISTING NOTEBOOK FILE FROM A GIT REPOSITORY USING THE COMMAND LINE INTERFACE | 7 |
| 1.5. ADDITIONAL RESOURCES | 8 |
| CHAPTER 2. COLLABORATING ON NOTEBOOKS USING GIT | 9 |
| 2.1. UPLOADING AN EXISTING NOTEBOOK FILE FROM A GIT REPOSITORY USING JUPYTERLAB | 9 |
| 2.2. UPLOADING AN EXISTING NOTEBOOK FILE FROM A GIT REPOSITORY USING THE COMMAND LINE INTERFACE | 9 |
| 2.3. UPDATING YOUR PROJECT WITH CHANGES FROM A REMOTE GIT REPOSITORY | 10 |
| 2.4. PUSHING PROJECT CHANGES TO A GIT REPOSITORY | 11 |
| CHAPTER 3. WORKING ON DATA SCIENCE PROJECTS | 12 |
| 3.1. USING DATA SCIENCE PROJECTS | 12 |
| 3.1.1. Creating a data science project | 12 |
| 3.1.2. Updating a data science project | 13 |
| 3.1.3. Deleting a data science project | 13 |
| 3.2. USING PROJECT WORKBENCHES | 14 |
| 3.2.1. Creating a project workbench | 14 |
| 3.2.2. Starting a workbench | 16 |
| 3.2.3. Updating a project workbench | 16 |
| 3.2.4. Deleting a workbench from a data science project | 17 |
| 3.3. USING DATA CONNECTIONS | 18 |
| 3.3.1. Adding a data connection to your data science project | 18 |
| 3.3.2. Deleting a data connection | 19 |
| 3.3.3. Updating a connected data source | 19 |
| 3.4. CONFIGURING CLUSTER STORAGE | 20 |
| 3.4.1. Adding cluster storage to your data science project | 20 |
| 3.4.2. Updating cluster storage | 21 |
| 3.4.3. Deleting cluster storage from a data science project | 22 |
| 3.5. CONFIGURING DATA SCIENCE PIPELINES | 23 |
| 3.5.1. Configuring a pipeline server | 23 |
| 3.5.2. Defining a pipeline | 24 |
| 3.5.3. Importing a data science pipeline | 25 |
| 3.6. CONFIGURING MODEL SERVERS | 26 |
| 3.6.1. Adding a model server to your data science project | 26 |
| 3.6.2. Adding a custom model-serving runtime | 28 |
| 3.6.3. Updating a model server | 30 |
| 3.6.4. Deleting a model server | 32 |
| 3.7. CONFIGURING ACCESS TO DATA SCIENCE PROJECTS | 32 |
| 3.7.1. Configuring access to data science projects | 33 |
| 3.7.2. Sharing access to a data science project | 33 |
| 3.7.3. Updating access to a data science project | 34 |
| 3.7.4. Removing access to a data science project | 35 |
| 3.8. VIEWING PYTHON PACKAGES INSTALLED ON YOUR NOTEBOOK SERVER | 36 |
| 3.9. INSTALLING PYTHON PACKAGES ON YOUR NOTEBOOK SERVER | 37 |
| 3.10. UPDATING NOTEBOOK SERVER SETTINGS BY RESTARTING YOUR SERVER | 38 |

| | |
|---|-----------|
| CHAPTER 4. WORKING WITH DATA SCIENCE PIPELINES | 40 |
| 4.1. MANAGING DATA SCIENCE PIPELINES | 40 |
| 4.1.1. Configuring a pipeline server | 40 |
| 4.1.2. Defining a pipeline | 42 |
| 4.1.3. Importing a data science pipeline | 43 |
| 4.1.4. Downloading a data science pipeline | 43 |
| 4.1.5. Deleting a data science pipeline | 44 |
| 4.1.6. Deleting a pipeline server | 45 |
| 4.1.7. Viewing the details of a pipeline server | 45 |
| 4.1.8. Viewing existing pipelines | 46 |
| 4.2. MANAGING PIPELINE RUNS | 47 |
| 4.2.1. Overview of pipeline runs | 47 |
| 4.2.2. Scheduling a pipeline run | 47 |
| 4.2.3. Cloning a scheduled pipeline run | 48 |
| 4.2.4. Stopping a triggered pipeline run | 50 |
| 4.2.5. Deleting a scheduled pipeline run | 50 |
| 4.2.6. Deleting a triggered pipeline run | 51 |
| 4.2.7. Viewing scheduled pipeline runs | 52 |
| 4.2.8. Viewing triggered pipeline runs | 53 |
| 4.2.9. Viewing the details of a pipeline run | 53 |
| 4.3. WORKING WITH PIPELINES IN JUPYTERLAB | 54 |
| 4.3.1. Overview of pipelines in JupyterLab | 54 |
| 4.3.2. Accessing the pipeline editor | 55 |
| 4.3.3. Creating a runtime configuration | 56 |
| 4.3.4. Updating a runtime configuration | 58 |
| 4.3.5. Deleting a runtime configuration | 60 |
| 4.3.6. Duplicating a runtime configuration | 61 |
| 4.3.7. Running a pipeline in JupyterLab | 62 |
| 4.3.8. Exporting a pipeline in JupyterLab | 63 |
| 4.4. ADDITIONAL RESOURCES | 64 |
| CHAPTER 5. MODEL SERVING ON OPENSIFT DATA SCIENCE | 65 |
| 5.1. CONFIGURING MODEL SERVERS | 65 |
| 5.1.1. Adding a model server to your data science project | 65 |
| 5.1.2. Adding a custom model-serving runtime | 67 |
| 5.1.3. Updating a model server | 69 |
| 5.1.4. Duplicating a model server | 71 |
| 5.1.5. Deleting a model server | 71 |
| 5.2. WORKING WITH DEPLOYED MODELS | 72 |
| 5.2.1. Deploying a model in OpenShift Data Science | 72 |
| 5.2.2. Viewing a deployed model | 74 |
| 5.2.3. Updating the deployment properties of a deployed model | 74 |
| 5.2.4. Deleting a deployed model | 75 |
| CHAPTER 6. WORKING WITH ACCELERATORS | 77 |
| 6.1. OVERVIEW OF ACCELERATORS | 77 |
| 6.2. WORKING WITH ACCELERATOR PROFILES | 77 |
| 6.2.1. Creating an accelerator profile | 78 |
| 6.2.2. Updating an accelerator profile | 79 |
| 6.2.3. Deleting an accelerator profile | 80 |
| 6.2.4. Configuring a recommended accelerator for notebook images | 81 |
| 6.2.5. Configuring a recommended accelerator for serving runtimes | 82 |
| 6.3. HABANA GAUDI INTEGRATION | 83 |

| | |
|---|-----------|
| 6.3.1. Enabling Habana Gaudi devices | 84 |
| CHAPTER 7. WORKING WITH DISTRIBUTED WORKLOADS | 86 |
| 7.1. OVERVIEW OF DISTRIBUTED WORKLOADS | 86 |
| 7.2. CONFIGURING DISTRIBUTED WORKLOADS | 86 |
| 7.3. RUNNING DISTRIBUTED DATA SCIENCE WORKLOADS FROM NOTEBOOKS | 88 |
| CHAPTER 8. SERVING LARGE LANGUAGE MODELS | 90 |
| 8.1. OVERVIEW OF REQUIRED OPERATORS FOR THE CAIKIT-TGIS-KSERVE STACK | 90 |
| 8.2. INSTALLING THE CAIKIT-TGIS-KSERVE STACK | 91 |
| 8.3. ADDITIONAL RESOURCES | 91 |
| CHAPTER 9. TROUBLESHOOTING COMMON PROBLEMS IN JUPYTER FOR ADMINISTRATORS | 92 |
| 9.1. A USER RECEIVES A 404: PAGE NOT FOUND ERROR WHEN LOGGING IN TO JUPYTER | 92 |
| 9.2. A USER'S NOTEBOOK SERVER DOES NOT START | 92 |
| 9.3. THE USER RECEIVES A DATABASE OR DISK IS FULL ERROR OR A NO SPACE LEFT ON DEVICE ERROR WHEN THEY RUN NOTEBOOK CELLS | 93 |
| CHAPTER 10. TROUBLESHOOTING COMMON PROBLEMS IN JUPYTER FOR USERS | 95 |
| 10.1. I SEE A 403: FORBIDDEN ERROR WHEN I LOG IN TO JUPYTER | 95 |
| 10.2. MY NOTEBOOK SERVER DOES NOT START | 95 |
| 10.3. I SEE A DATABASE OR DISK IS FULL ERROR OR A NO SPACE LEFT ON DEVICE ERROR WHEN I RUN MY NOTEBOOK CELLS | 95 |

CHAPTER 1. CREATING AND IMPORTING NOTEBOOKS

You can create a blank notebook or import a notebook from a number of different sources.

1.1. CREATING A NEW NOTEBOOK

You can create a new Jupyter notebook from an existing notebook container image to access its resources and properties. The **Notebook server control panel** contains a list of available container images that you can run as a single-user notebook server.

Prerequisites

- Ensure that you have logged in to Red Hat OpenShift Data Science.
- Ensure that you have launched your notebook server and logged in to Jupyter.
- The notebook image exists in a registry, image stream, and is accessible.

Procedure

1. Click **File** → **New** → **Notebook**.
2. If prompted, select a kernel for your notebook from the list.
If you want to use a kernel, click **Select**. If you do not want to use a kernel, click **No Kernel**.

Verification

- Check that the notebook file is visible in the JupyterLab interface.

1.1.1. Notebook images for data scientists

Red Hat OpenShift Data Science contains Jupyter notebook images optimized with industry-leading tools and libraries required for your data science work. To provide a consistent, stable platform for your model development, all notebook images contain the same version of Python. Notebook images available on Red Hat OpenShift Data Science are pre-built and ready for you to use immediately after OpenShift Data Science is installed or upgraded.

Notebook images are supported for a minimum of one year. Major updates to pre-configured notebook images occur about every six months. Therefore, two supported notebook image versions are typically available at any given time. You can use this support period to update your code to use components from the latest available notebook image. Legacy notebook image versions, that is, not the two most recent versions, might still be available for selection. Legacy image versions include a label that indicates the image is out-of-date. To use the latest package versions, Red Hat recommends that you use the most recently added notebook image. If necessary, you can still access older notebook images from the registry, even if they are no longer supported. You can then add the older notebook images as custom notebook images to cater for your project's specific requirements.

See the table in [Options for notebook server environments](#) for a complete list of packages and versions included in these images.

Red Hat OpenShift Data Science contains the following notebook images that are available by default:

Table 1.1. Default notebook images

| Image name | Description |
|-----------------------|--|
| CUDA | If you are working with compute-intensive data science models that require GPU support, use the Compute Unified Device Architecture (CUDA) notebook image to gain access to the NVIDIA CUDA Toolkit. Using this toolkit, you can optimize your work using GPU-accelerated libraries and optimization tools. |
| Standard Data Science | Use the Standard Data Science notebook image for models that do not require TensorFlow or PyTorch. This image contains commonly used libraries to assist you in developing your machine learning models. |
| TensorFlow | TensorFlow is an open source platform for machine learning. With TensorFlow, you can build, train and deploy your machine learning models. TensorFlow contains advanced data visualization features, such as computational graph visualizations. It also allows you to easily monitor and track the progress of your models. |
| PyTorch | PyTorch is an open source machine learning library optimized for deep learning. If you are working with computer vision or natural language processing models, use the Pytorch notebook image. |
| Minimal Python | If you do not require advanced machine learning features, or additional resources for compute-intensive data science work, you can use the Minimal Python image to develop your models. |
| TrustyAI | Use the TrustyAI notebook image to leverage your data science work with model explainability, tracing, and accountability, and runtime monitoring. |
| HabanaAI | The HabanaAI notebook image optimizes high-performance deep learning (DL) with Habana Gaudi devices. Habana Gaudi devices accelerate DL training workloads and maximize training throughput and efficiency. |

Additional resources

- [Installing Python packages on your notebook server](#)
- [Options for notebook server environments](#)


1.2. UPLOADING AN EXISTING NOTEBOOK FILE FROM LOCAL STORAGE

You can load an existing notebook from local storage into JupyterLab to continue work, or adapt a project for a new use case.

Prerequisites

- Credentials for logging in to Jupyter.
- A launched and running notebook server.
- A notebook file exists in your local storage.

Procedure

1. In the **File Browser** in the left sidebar of the JupyterLab interface, click **Upload Files** ().
2. Locate and select the notebook file and click **Open**.
The file is displayed in the **File Browser**.

Verification

- The notebook file displays in the **File Browser** in the left sidebar of the JupyterLab interface.
- You can open the notebook file in JupyterLab.

1.3. UPLOADING AN EXISTING NOTEBOOK FILE FROM A GIT REPOSITORY USING JUPYTERLAB

You can use the JupyterLab user interface to clone a Git repository into your workspace to continue your work or integrate files from an external project.


Prerequisites

- A launched and running Jupyter server.
- Read access for the Git repository you want to clone.

Procedure

1. Copy the HTTPS URL for the Git repository.
 - On GitHub, click **Code** → **HTTPS** and click the Clipboard button.
 - On GitLab, click **Clone** and click the Clipboard button under **Clone with HTTPS**.

2. In the JupyterLab interface, click the **Git Clone** button ().

You can also click **Git** → **Clone a repository** in the menu, or click the Git icon () and click the **Clone a repository** button.

The *Clone a repo* dialog appears.

3. Enter the HTTPS URL of the repository that contains your notebook.
4. Click **CLONE**.
5. If prompted, enter your username and password for the Git repository.

Verification

- Check that the contents of the repository are visible in the file browser in JupyterLab, or run the **ls** command in the terminal to verify that the repository is shown as a directory.

1.4. UPLOADING AN EXISTING NOTEBOOK FILE FROM A GIT REPOSITORY USING THE COMMAND LINE INTERFACE

You can use the command line interface to clone a Git repository into your workspace to continue your work or integrate files from an external project.

Prerequisites

- A launched and running Jupyter server.

Procedure

1. Copy the HTTPS URL for the Git repository.
 - On GitHub, click **Code** → **HTTPS** and click the Clipboard button.
 - On GitLab, click **Clone** and click the Clipboard button under **Clone with HTTPS**.
2. In JupyterLab, click **File** → **New** → **Terminal** to open a terminal window.
3. Enter the **git clone** command.

```
git clone <git-clone-URL>
```

Replace `<git-clone-URL>` with the HTTPS URL, for example:

```
[1234567890@jupyter-nb-jdoe ~]$ git clone https://github.com/example/myrepo.git
Cloning into myrepo...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 2821 (delta 1), reused 5 (delta 1), pack-reused 2810
Receiving objects: 100% (2821/2821), 39.17 MiB | 23.89 MiB/s, done.
Resolving deltas: 100% (1416/1416), done.
```

Verification

- Check that the contents of the repository are visible in the file browser in JupyterLab, or run the `ls` command in the terminal to verify that the repository is shown as a directory.

1.5. ADDITIONAL RESOURCES

- [Collaborating on notebooks using Git](#)

CHAPTER 2. COLLABORATING ON NOTEBOOKS USING GIT

If your notebooks or other files are stored in Git version control, you can import them from a Git repository onto your notebook server to work with them in JupyterLab. When you are ready, you can push your changes back to the Git repository so that others can review or use your models.

2.1. UPLOADING AN EXISTING NOTEBOOK FILE FROM A GIT REPOSITORY USING JUPYTERLAB

You can use the JupyterLab user interface to clone a Git repository into your workspace to continue your work or integrate files from an external project.


Prerequisites

- A launched and running Jupyter server.
- Read access for the Git repository you want to clone.

Procedure

1. Copy the HTTPS URL for the Git repository.
 - On GitHub, click **Code** → **HTTPS** and click the Clipboard button.
 - On GitLab, click **Clone** and click the Clipboard button under **Clone with HTTPS**.

2. In the JupyterLab interface, click the **Git Clone** button ().

You can also click **Git** → **Clone a repository** in the menu, or click the Git icon () and click the **Clone a repository** button.

The *Clone a repo* dialog appears.

3. Enter the HTTPS URL of the repository that contains your notebook.
4. Click **CLONE**.
5. If prompted, enter your username and password for the Git repository.

Verification

- Check that the contents of the repository are visible in the file browser in JupyterLab, or run the `ls` command in the terminal to verify that the repository is shown as a directory.

2.2. UPLOADING AN EXISTING NOTEBOOK FILE FROM A GIT REPOSITORY USING THE COMMAND LINE INTERFACE

You can use the command line interface to clone a Git repository into your workspace to continue your work or integrate files from an external project.

Prerequisites

- A launched and running Jupyter server.

Procedure

1. Copy the HTTPS URL for the Git repository.
 - On GitHub, click **Code** → **HTTPS** and click the Clipboard button.
 - On GitLab, click **Clone** and click the Clipboard button under **Clone with HTTPS**.
2. In JupyterLab, click **File** → **New** → **Terminal** to open a terminal window.
3. Enter the **git clone** command.

```
git clone <git-clone-URL>
```

Replace `*git-clone-URL*` with the HTTPS URL, for example:

```
[1234567890@jupyter-nb-jdoe ~]$ git clone https://github.com/example/myrepo.git
Cloning into myrepo...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 2821 (delta 1), reused 5 (delta 1), pack-reused 2810
Receiving objects: 100% (2821/2821), 39.17 MiB | 23.89 MiB/s, done.
Resolving deltas: 100% (1416/1416), done.
```

Verification

- Check that the contents of the repository are visible in the file browser in JupyterLab, or run the **ls** command in the terminal to verify that the repository is shown as a directory.



2.3. UPDATING YOUR PROJECT WITH CHANGES FROM A REMOTE GIT REPOSITORY

You can pull changes made by other users into your data science project from a remote Git repository.

Prerequisites

- You have configured the remote Git repository.
- You have already imported the Git repository into JupyterLab, and the contents of the repository are visible in the file browser in JupyterLab.
- You have permissions to pull files from the remote Git repository to your local repository.
- You have credentials for logging in to Jupyter.
- You have a launched and running Jupyter server.

Procedure

1. In the JupyterLab interface, click the **Git** button ().
2. Click the **Pull latest changes** button ().

Verification

- You can view the changes pulled from the remote repository in the **History** tab of the Git pane.


2.4. PUSHING PROJECT CHANGES TO A GIT REPOSITORY

To build and deploy your application in a production environment, upload your work to a remote Git repository.

Prerequisites

- You have opened a notebook in the JupyterLab interface.
- You have already added the relevant Git repository to your notebook server.
- You have permission to push changes to the relevant Git repository.
- You have installed the Git version control extension.

Procedure

1. Click **File** → **Save All** to save any unsaved changes.
2. Click the Git icon () to open the Git pane in the JupyterLab interface.
3. Confirm that your changed files appear under **Changed**.
If your changed files appear under **Untracked**, click **Git** → **Simple Staging** to enable a simplified Git process.
4. Commit your changes.
 - a. Ensure that all files under **Changed** have a blue checkmark beside them.
 - b. In the **Summary** field, enter a brief description of the changes you made.
 - c. Click **Commit**.
5. Click **Git** → **Push to Remote** to push your changes to the remote repository.
6. When prompted, enter your Git credentials and click **OK**.

Verification

- Your most recently pushed changes are visible in the remote Git repository.

CHAPTER 3. WORKING ON DATA SCIENCE PROJECTS

As a data scientist, you can organize your data science work into a single project. A data science project in OpenShift Data Science can consist of the following components:

Workbenches

Creating a workbench allows you to add a Jupyter notebook to your project.

Cluster storage

For data science projects that require data to be retained, you can add cluster storage to the project.

Data connections

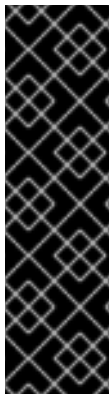
Adding a data connection to your project allows you to connect data inputs to your workbenches.

Pipelines

Standardize and automate machine learning workflows to enable you to further enhance and deploy your data science models.

Models and model servers

Deploy a trained data science model to serve intelligent applications. Your model is deployed with an endpoint that allows applications to send requests to the model.



IMPORTANT

If you create an OpenShift project outside of the OpenShift Data Science user interface, the project is not shown on the **Data science projects** page. In addition, you cannot use features exclusive to OpenShift Data Science, such as workbenches and model serving, with a standard OpenShift project.

To classify your OpenShift project as a data science project, and to make available features exclusive to OpenShift Data Science, you must add the label **opendatahub.io/dashboard: 'true'** to the project namespace. After you add this label, your project is subsequently shown on the **Data science projects** page.

3.1. USING DATA SCIENCE PROJECTS

3.1.1. Creating a data science project

To start your data science work, create a data science project. Creating a project helps you organize your work in one place. You can also enhance the capabilities of your data science project by adding workbenches, adding storage to your project's cluster, adding data connections, and adding model servers.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.

2. Click **Create data science project**
The **Create a data science project** dialog opens.
3. Enter a **name** for your data science project.
4. Optional: Edit the **resource name** for your data science project. The resource name must consist of lowercase alphanumeric characters, -, and must start and end with an alphanumeric character.
5. Enter a **description** for your data science project.
6. Click **Create**.
A project details page opens. From here, you can create workbenches, add cluster storage, and add data connections to your project.

Verification

- The data science project that you created is displayed on the **Data science projects** page.


3.1.2. Updating a data science project

You can update your data science project's details by changing your project's name and description text.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have created a data science project.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the action menu () beside the project whose details you want to update and click **Edit project**.
The **Edit data science project** dialog opens.
3. Optional: Update the **name** for your data science project.
4. Optional: Update the **description** for your data science project.
5. Click **Update**.

Verification

- The data science project that you updated is displayed on the **Data science projects** page.


3.1.3. Deleting a data science project

You can delete data science projects so that they do not appear on the OpenShift Data Science **Data science projects** page when you no longer want to use them.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users**) in OpenShift.
- You have created a data science project.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the action menu () beside the project that you want to delete and click **Delete project**.
The **Delete project** dialog opens.
3. Enter the project name in the text field to confirm that you intend to delete it.
4. Click **Delete project**.

Verification

- The data science project that you deleted is no longer displayed on the **Data science projects** page.
- Deleting a data science project deletes any associated workbenches, cluster storage, and data connections. This data is permanently deleted and is not recoverable.

3.2. USING PROJECT WORKBENCHES

3.2.1. Creating a project workbench

To examine and work with data models in an isolated area, you can create a workbench. You can use this workbench to create a Jupyter notebook from an existing notebook container image to access its resources and properties. For data science projects that require data retention, you can add container storage to the workbench you are creating. If you require extra power for use with large datasets, you can assign accelerators to your workbench to optimize performance.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you use specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that you can add a workbench to.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to add the workbench to.
The **Details** page for the project opens.

3. In the **Workbenches** section, click **Create workbench**.
The **Create workbench** page opens.
4. Configure the properties of the workbench you are creating.
 - a. In the **Name** field, enter a name for your workbench.
 - b. Optional: In the **Description** field, enter a description to define your workbench.
 - c. In the **Notebook image** section, complete the fields to specify the notebook image to use with your workbench.
 - i. From the **Image selection** list, select a notebook image.
 - d. In the **Deployment size** section, specify the size of your deployment instance.
 - i. From the **Container size** list, select a container size for your server.
 - ii. Optional: From the **Accelerator** list, select an accelerator.
 - iii. If you selected an accelerator in the preceding step, specify the number of accelerators to use.
 - e. Optional: Select and specify values for any new **environment variables**.

**NOTE**

To enable data science pipelines in JupyterLab in self-managed deployments, create the following environment variable:

PIPELINES_SSL_SA_CERTS=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt


- f. Configure the storage for your OpenShift Data Science cluster.
 - i. Select **Create new persistent storage** to create storage that is retained after you log out of OpenShift Data Science. Complete the relevant fields to define the storage.
 - ii. Select **Use existing persistent storage** to reuse existing storage and select the storage from the **Persistent storage** list.
- g. To use a data connection, in the **Data connections** section, select the **Use a data connection** checkbox.
 - Create a new data connection as follows:
 - i. Select **Create new data connection**
 - ii. In the **Name** field, enter a unique name for the data connection.
 - iii. In the **Access key** field, enter your access key ID for Amazon Web Services (AWS).
 - iv. In the **Secret key** field, enter your secret access key for the AWS account that you specified.
 - v. In the **Endpoint** field, enter the endpoint of your AWS S3 storage.
 - vi. In the **Region** field, enter the default region of your AWS account.

If your data science work requires you to change your workbench's notebook image, container size, or identifying information, you can update the properties of your project's workbench. If you require extra power for use with large datasets, you can assign accelerators to your workbench to optimize performance.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you use specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have created a data science project that has a workbench.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project whose workbench you want to update.
The **Details** page for the project opens.
3. Click the action menu () beside the workbench that you want to update in the **Workbenches** section and click **Edit workbench**.
The **Edit workbench** page opens.
4. Update any of the workbench properties and then click **Update workbench**.

Verification

- The workbench that you updated appears on the **Details** page for the project.

3.2.4. Deleting a workbench from a data science project


You can delete workbenches from your data science projects to help you remove Jupyter notebooks that are no longer relevant to your work.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project with a workbench.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to delete the workbench from.
The **Details** page for the project opens.

3. Click the action menu () beside the workbench that you want to delete in the **Workbenches** section and click **Delete workbench**.
The **Delete workbench** dialog opens.
4. Enter the name of the workbench in the text field to confirm that you intend to delete it.
5. Click **Delete workbench**.

Verification

- The workbench that you deleted is no longer displayed in the **Workbenches** section on the project **Details** page.
- The custom resource (CR) associated with the workbench's Jupyter notebook is deleted.

3.3. USING DATA CONNECTIONS

3.3.1. Adding a data connection to your data science project

You can enhance your data science project by adding a connection to a data source. When you want to work with a very large data sets, you can store your data in an S3-compatible object storage bucket, so that you do not fill up your local storage. You also have the option of associating the data connection with an existing workbench that does not already have a connection.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that you can add a data connection to.
- You have access to S3-compatible object storage.
- If you intend to add the data connection to an existing workbench, you have saved any data in the workbench to avoid losing work.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to add a data connection to.
A project details page opens.
3. In the **Data connections** section of the page, click **Add data connection**.
The **Add data connection** dialog opens.
4. Enter a **name** for the data connection.
5. In the **Access key** field, enter the access key ID for your S3-compatible object storage provider.
6. In the **Secret key** field, enter the secret access key for the S3-compatible object storage account you specified.

7. In the **Endpoint** field, enter the endpoint of your S3-compatible object storage bucket.
8. In the **Region** field, enter the default region of your S3-compatible object storage account.
9. In the **Bucket** field, enter the name of your S3-compatible object storage bucket.
10. Click **Add data connection**

Verification

- The data connection that you added appears in the **Data connections** section on the **Details** page for the project.
- If you selected a workbench, the data connection is visible in the **Workbenches** section on your data science project page.


3.3.2. Deleting a data connection

You can delete data connections from your data science projects to help you remove connections that are no longer relevant to your work.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project with a data connection.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to delete the data connection from.
The **Details** page for the project opens.
3. Click the action menu () beside the data connection that you want to delete in the **Data connections** section and click **Delete data connection**.
The **Delete data connection** dialog opens.
4. Enter the name of the data connection in the text field to confirm that you intend to delete it.
5. Click **Delete data connection**

Verification

- The data connection that you deleted is no longer displayed in the **Data connections** section on the project **Details** page.

3.3.3. Updating a connected data source

To use an existing data source with a different workbench, you can change the data source that is connected to your project's workbench.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have created a data science project, created a workbench, and you have defined a data connection.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project whose data source you want to change.
A project details page opens.
3. Click the action menu (**⋮**) beside the data source that you want to change in the **Data connections** section and click **Change connected workbenches**.
The **Update connected workbenches** dialog opens.
4. Select an existing **workbench** to connect the data source to from the list.
5. Click **Update connected workbenches**.

Verification

- The data connection that you changed is displayed in the **Data connections** section on the project **Details** page.
- You can access your S3 data source using environment variables in the connected workbench.

3.4. CONFIGURING CLUSTER STORAGE

3.4.1. Adding cluster storage to your data science project

For data science projects that require data to be retained, you can add cluster storage to the project. Additionally, you can also connect cluster storage to a specific project's workbench.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that you can add cluster storage to.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to add the cluster storage to.

- A project details page opens.
3. In the **Cluster storage** section of the page, click **Add cluster storage**. The **Add storage** dialog opens.
 4. Enter a **name** for the cluster storage.
 5. Enter a **description** for the cluster storage.
 6. Under **Persistent storage size**, enter a new size in gibibytes. The minimum size is 1 GiB, and the maximum size is 16384 GiB.
 7. Optional: Select a **workbench** from the list to connect the cluster storage to an existing workbench.
 8. If you selected a workbench to connect the storage to, enter the storage directory in the **Mount folder** field.
 9. Click **Add storage**.

Verification

- The cluster storage that you added appears in the **Cluster storage** section on the **Details** page for the project.
- A new persistent volume claim (PVC) is created with the storage size that you defined.
- The persistent volume claim (PVC) is visible as an attached storage in the **Workbenches** section on the **Details** page for the project.

3.4.2. Updating cluster storage

If your data science work requires you to change the identifying information of a project's cluster storage or the workbench that the storage is connected to, you can update your project's cluster storage to change these properties.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have created a data science project that contains cluster storage.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**. The **Data science projects** page opens.
2. Click the name of the project whose storage you want to update. The **Details** page for the project opens.
3. Click the action menu (**⋮**) beside the storage that you want to update in the **Cluster storage** section and click **Edit storage**. The **Edit storage** page opens.

4. Update the storage's properties.
 - a. Update the **name** for the storage, if applicable.
 - b. Update the **description** for the storage, if applicable.
 - c. Increase the **Persistent storage size** for the storage, if applicable.
Note that you can only increase the storage size. Updating the storage size restarts the workbench and makes it unavailable for a period of time that is usually proportional to the size change.
 - d. Update the **workbench** that the storage is connected to, if applicable.
 - e. If you selected a new workbench to connect the storage to, enter the storage directory in the **Mount folder** field.
5. Click **Update storage**.

If you increased the storage size, the workbench restarts and is unavailable for a period of time that is usually proportional to the size change.

Verification

- The storage that you updated appears in the **Cluster storage** section on the **Details** page for the project.

3.4.3. Deleting cluster storage from a data science project

You can delete cluster storage from your data science projects to help you free up resources and delete unwanted storage space.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project with cluster storage.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to delete the storage from.
A project details page opens.
3. In the **Cluster storage** section, click the action menu (**⋮**) beside the storage that you want to delete and then click **Delete storage**.
The **Delete storage** dialog opens.
4. Enter the name of the storage in the text field to confirm that you intend to delete it.
5. Click **Delete storage**.

Verification

- The storage that you deleted is no longer displayed in the **Cluster storage** section on the project **Details** page.
- The persistent volume (PV) and persistent volume claim (PVC) associated with the cluster storage are both permanently deleted. This data is not recoverable.

3.5. CONFIGURING DATA SCIENCE PIPELINES

3.5.1. Configuring a pipeline server

Before you can successfully create a pipeline in OpenShift Data Science, you must configure a pipeline server. This includes configuring where your pipeline artifacts and data are stored.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that you can add a pipeline server to.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to configure a pipeline server for.
A project details page opens.
3. In the **Pipelines** section, click **Create a pipeline server**.
The **Configure pipeline server** dialog appears.
4. In the **Object storage connection** section, to specify the S3-compatible data connection to store your pipeline artifacts, select one of the following sets of actions:



NOTE

After the pipeline server is created, the **/metadata** and **/artifacts** folders are automatically created in the default **root** folder. Therefore, you are not required to specify any storage directories when configuring a data connection for your pipeline server.

- Select **Existing data connection** to use a data connection that you previously defined. If you selected this option, from the **Name** list, select the name of the relevant data connection and skip to step 6.
- Select **Create new data connection** to add a new data connection that your pipeline server can access.

5. If you selected **Create new data connection**, perform the following steps:
 - a. In the **Name** field, enter a name for the data connection.
 - b. In the **Access key** field, enter your access key ID for Amazon Web Services.
 - c. In the **Secret key** field, enter your secret access key for the account you specified.
 - d. Optional: In the **Endpoint** field, enter the endpoint of your AWS S3 storage.
 - e. Optional: In the **Region** field, enter the default region of your AWS account.
 - f. In the **Bucket** field, enter the name of the AWS S3 bucket.



IMPORTANT

If you are creating a new data connection, in addition to the other designated mandatory fields, the **AWS_S3_BUCKET** field is mandatory. If you specify incorrect data connection settings, you cannot update these settings on the same pipeline server. Therefore, you must delete the pipeline server and configure another one.

6. In the **Database** section, click **Show advanced database options** to specify the database to store your pipeline data and select one of the following sets of actions:
 - Select **Use default database stored on your cluster** to deploy a MariaDB database in your project.
 - Select **Connect to external MySQL database** to add a new connection to an external database that your pipeline server can access.
 - i. In the **Host** field, enter the database's host name.
 - ii. In the **Port** field, enter the database's port.
 - iii. In the **Username** field, enter the default user name that is connected to the database.
 - iv. In the **Password** field, enter the password for the default user account.
 - v. In the **Database** field, enter the database name.
7. Click **Configure**.

Verification

- The pipeline server that you configured is displayed in the **Pipelines** section on the project details page.
- The **Import pipeline** button is available in the **Pipelines** section on the project details page.

3.5.2. Defining a pipeline

The Kubeflow Pipelines SDK enables you to define end-to-end machine learning and data pipelines. Use the Kubeflow Pipelines SDK to build your data science pipeline in Python code. After you have built your pipeline, compile it into Tekton-formatted YAML code using kfp-tekton SDK (version 1.5.x only). After

defining the pipeline, you can import the YAML file to the OpenShift Data Science dashboard to enable you to configure its execution settings. For more information about installing and using Kubeflow Pipelines SDK for Tekton, see [Kubeflow Pipelines SDK for Tekton](#).

You can also use the Elyra JupyterLab extension to create and run data science pipelines within JupyterLab. For more information on creating pipelines in JupyterLab, see [Working with pipelines in JupyterLab](#). For more information on the Elyra JupyterLab extension, see [Elyra Documentation](#).

Additional resources

- [Kubeflow Pipelines SDK for Tekton](#)
- [KFP Tekton samples and compiler samples](#)
- [Kubeflow Pipelines v1 Documentation](#)
- [Elyra Documentation](#)

3.5.3. Importing a data science pipeline

To help you begin working with data science pipelines in OpenShift Data Science, you can import a YAML file containing your pipeline's code to an active pipeline server. This file contains a Kubeflow pipeline compiled with the Tekton compiler. After you have imported the pipeline to a pipeline server, you can execute the pipeline by creating a pipeline run.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have previously created a data science project that is available and contains a configured pipeline server.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines** → **Pipelines**. The **Pipelines** page opens.
2. From the **Project** list, select the project that you want to import a pipeline to.
3. Click **Import pipeline**. The **Import pipeline** dialog opens.
4. Enter the details for the pipeline that you are importing.
 - a. In the **Pipeline name** field, enter a name for the pipeline that you are importing.
 - b. In the **Pipeline description** field, enter a description for the pipeline that you are importing.
 - c. Click **Upload**. Alternatively, drag the file from your local machine's file system and drop it in the designated area in the **Import pipeline** dialog. A file browser opens.

- d. Navigate to the file containing the pipeline code and click **Select**.
- e. Click **Import pipeline**.

Verification

- The pipeline that you imported is displayed on the **Pipelines** page.

For more information about using pipelines in OpenShift Data Science, see *Working with data science pipelines*.

3.6. CONFIGURING MODEL SERVERS

3.6.1. Adding a model server to your data science project

Before you can successfully deploy a data science model on OpenShift Data Science, you must configure a model server. This includes configuring the number of replicas to deploy, the server size, the token authorization, and how to access the project. If you require extra power for use with large datasets, you can assign accelerators to your model server to optimize performance.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you use specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that you can add a model server to.
- If you want to use a custom model-serving runtime for your model server, you have added and enabled the runtime. See [Adding a custom model-serving runtime](#).
- If you want to use graphics processing units (GPUs) with your model server, you have enabled GPU support in OpenShift Data Science. See [Enabling GPU support in OpenShift Data Science](#).

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to configure a model server for.
A project details page opens.
3. In the **Models and model servers** section, click **Add server**.
The **Add model server** dialog opens.
4. In the **Model server name** field, enter a unique name for the model server.
5. From the **Serving runtime** list, select a model-serving runtime that is installed and enabled in your OpenShift Data Science deployment.
6. In the **Number of model replicas to deploy** field, specify a value.
7. From the **Model server size** list, select one of the following server sizes:

- Small
 - Medium
 - Large
 - Custom
8. Optional: If you selected **Custom** in the preceding step, configure the following settings in the **Model server size** section to customize your model server:
- a. In the **CPUs requested** field, specify a number of CPUs to use with your model server. Use the list beside this field to specify the value in cores or millicores.
 - b. In the **CPU limit** field, specify the maximum number of CPUs to use with your model server. Use the list beside this field to specify the value in cores or millicores.
 - c. In the **Memory requested** field, specify the requested memory for the model server in gibibytes (Gi).
 - d. In the **Memory limit** field, specify the maximum memory limit for the model server in gibibytes (Gi).



IMPORTANT

OpenShift Data Science includes two versions of the OpenVINO Model Server (OVMS) runtime by default; a version that supports GPUs and one that does not. To use GPUs, from the **Serving runtime** list, you must select the version whose display name includes **Supports GPUs**.

If you are using a *custom* model-serving runtime with your model server, you must ensure that your custom runtime supports GPUs and is appropriately configured to use them.

9. Optional: From the **Accelerator** list, select an accelerator.
- a. If you selected an accelerator in the preceding step, specify the number of accelerators to use.
10. Optional: In the **Model route** section, select the **Make deployed models available through an external route** check box to make your deployed models available to external clients.
11. Optional: In the **Token authorization** section, select the **Require token authentication** check box to require token authentication for your model server. To finish configuring token authentication, perform the following actions:
- a. In the **Service account name** field, enter a service account name for which the token will be generated. The generated token is created and displayed in the **Token secret** field when the model server is configured.
 - b. To add an additional service account, click **Add a service account** and enter another service account name.
12. Click **Add**.

Verification

- The model server that you configured appears in the **Models and model servers** section of the project details page.

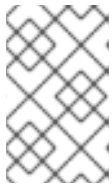
3.6.2. Adding a custom model-serving runtime

A model-serving runtime provides integration with a specified model server and the model frameworks that it supports. By default, Red Hat OpenShift Data Science includes the OpenVINO Model Server runtime. However, if this runtime doesn't meet your needs (it doesn't support a particular model framework, for example), you might want to add your own, custom runtimes.

As an administrator, you can use the OpenShift Data Science interface to add and enable custom model-serving runtimes. You can then choose from your enabled runtimes when you create a new model server.

Prerequisites

- You have logged in to OpenShift Data Science as an administrator.
- You are familiar with how to [add a model server to your project](#). When you have added a custom model-serving runtime, you must configure a new model server to use the runtime.
- You have reviewed the example runtimes in the [kserve/modelmesh-serving](#) repository. You can use these examples as *starting points*. However, each runtime requires some further modification before you can deploy it in OpenShift Data Science. The required modifications are described in the following procedure.



NOTE

OpenShift Data Science includes the OpenVINO Model Server model-serving runtime by default. You do not need to add this runtime to OpenShift Data Science.

Procedure

1. From the OpenShift Data Science dashboard, click **Settings > Serving runtimes**.
The **Serving runtimes** page opens and shows the model-serving runtimes that are already installed and enabled in your OpenShift Data Science deployment. By default, the OpenVINO Model Server runtime is pre-installed and enabled in OpenShift Data Science.
2. To add a new, custom runtime, click **Add serving runtime**.
The **Add serving runtime** page opens.
3. To start adding a new runtime, perform one of the following sets of actions:
 - **To upload a YAML file**
 - a. Click **Upload files**.
A file browser opens.
 - b. In the file browser, select a YAML file on your computer. This file might be the one of the example runtimes that you downloaded from the [kserve/modelmesh-serving](#) repository.
The embedded YAML editor opens and shows the contents of the file that you uploaded.
 - **To enter YAML code directly in the editor**

- a. Click **Start from scratch**
The embedded YAML editor opens with no content.
 - b. Enter or paste YAML code directly in the embedded editor. The YAML that you paste might be copied from one of the example runtimes in the [kserve/modelmesh-serving](#) repository.
4. Optional: If you are adding one of the example runtimes in the [kserve/modelmesh-serving](#) repository, perform the following modifications:
- a. In the YAML editor, locate the **kind** field for your runtime. Update the value of this field to **ServingRuntime**.
 - b. In the YAML editor, locate the **containers.image** field for your runtime. Based on the runtime that you are adding, update the field to one of the following:

Nvidia Triton Inference Server

image: nvcr.io/nvidia/tritonserver:21.06.1-py3

Seldon Python MLServer

image: seldonio/mlserver:0.5.2

TorchServe

image: pytorch/torchserve:0.6.0-cpu

5. In the **metadata.name** field, ensure that the value of the runtime you are adding is unique (that is, the value isn't the same as for a runtime you have already added).
6. Optional: To configure a custom display name for the runtime that you are adding, add a **metadata.annotations.openshift.io/display-name** field and specify a value, as shown in the following example:

```
apiVersion: serving.kserve.io/v1alpha1
kind: ServingRuntime
metadata:
  name: mlserver-0.x
  annotations:
    openshift.io/display-name: MLServer
```



NOTE

If you do not configure a custom display name for your runtime, OpenShift Data Science shows the value of the **metadata.name** field.

7. Click **Add**.
The **Serving runtimes** page opens and shows the updated list of runtimes that are installed. Observe that the runtime you added is automatically enabled.
8. Optional: To edit your custom runtime, click the action menu (**:**) and select **Edit**.

**NOTE**

You cannot directly edit the OpenVINO Model Server runtime that is included in OpenShift Data Science by default. However, you can *clone* this runtime and edit the cloned version. You can then add the edited clone as a new, custom runtime. To do this, click the action menu beside the OpenVINO Model Server and select **Clone**.

Verification

- The model-serving runtime you added is shown in an enabled state on the **Serving runtimes** page.

Additional resources

- To learn how to configure a model server that uses a custom model-serving runtime that you have added, see [Adding a model server to your data science project](#) .


3.6.3. Updating a model server

You can update your data science project's model server by changing details, such as the number of deployed replicas, the server size, the token authorization, and how to access the project. If you require extra power for use with large datasets, you can update you can assign accelerators to your model server to optimize performance.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you use specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have created a data science project that has a model server assigned.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project whose model server details you want to update.
A project details page opens.
3. In the **Models and model servers** section, locate the model server you want to update. Click the action menu () and select **Edit model server**.
The **Configure model server** dialog opens.
4. Update the model server properties, as follows:

**NOTE**

You cannot change the **Serving runtime** selection for a model server that is already configured. This protects against changing to a runtime that does not support already-deployed models.

- a. In the **Model server name** field, enter a new, unique name for the model server.
- b. In the **Number of model replicas to deploy** field, specify a value.
- c. From the **Model server size** list, select one of the following server sizes:
 - Small
 - Medium
 - Large
 - Custom
- d. Optional: If you selected **Custom** in the preceding step, configure the following settings in the **Model server size** section to customize your model server:
 - i. In the **CPUs requested** field, specify a number of CPUs to use with your model server. Use the list beside this field to specify the value in cores or millicores.
 - ii. In the **CPU limit** field, specify the maximum number of CPUs to use with your model server. Use the list beside this field to specify the value in cores or millicores.
 - iii. In the **Memory requested** field, specify the requested memory for the model server in gibibytes (Gi).
 - iv. In the **Memory limit** field, specify the maximum memory limit for the model server in gibibytes (Gi).



IMPORTANT

OpenShift Data Science includes two versions of the OpenVINO Model Server (OVMS) runtime by default; a version that supports GPUs and one that does not. To use GPUs, from the **Serving runtime** list, you must select the version whose display name includes **Supports GPUs**.

If you use a *custom* model-serving runtime with your model server, you must ensure that your custom runtime supports GPUs and is appropriately configured to use them.

- e. Optional: From the **Accelerator** list, select an accelerator.
- f. If you selected an accelerator in the preceding step, specify the number of accelerators to use.
- g. Optional: In the **Model route** section, select the **Make deployed models available through an external route** check box to make your deployed models available to external clients.
- h. Optional: In the **Token authorization** section, select the **Require token authentication** check box to require token authentication for your model server. To finish configuring token authentication, perform the following actions:
 - i. In the **Service account name** field, enter a service account name for which the token will be generated. The generated token is created and displayed in the **Token secret** field when the model server is configured.

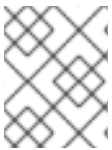
- ii. To add an additional service account, click **Add a service account** and enter another service account name.
5. Click **Configure**.

Verification

- The model server that you updated appears in the **Models and model servers** section on the project details page.

3.6.4. Deleting a model server

When you no longer need a model server to host models, you can remove it from your data science project.




NOTE

When you remove a model server, you also remove the models that are hosted on that model server. As a result, the models are no longer available to applications.

Prerequisites

- You have created a data science project and an associated model server.
- You have notified the users of the applications that access the models that the models will no longer be available.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project from which you want to delete the model server.
A project details page opens.
3. Click the action menu () beside the project whose model server you want to delete in the **Models and model servers** section and then click **Delete model server**.
The **Delete model server** dialog opens.
4. Enter the name of the model server in the text field to confirm that you intend to delete it.
5. Click **Delete model server**.

Verification

- The model server that you deleted is no longer displayed in the **Models and model servers** section on the project details page.

For more information about model serving, see *Model serving on OpenShift Data Science*.

3.7. CONFIGURING ACCESS TO DATA SCIENCE PROJECTS

3.7.1. Configuring access to data science projects

To enable you to work collaboratively on your data science projects with other users, you can share access to your project. After creating your project, you can then set the appropriate access permissions from the OpenShift Data Science user interface.

You can assign the following access permission levels to your data science projects:

- **Admin** - Users can modify all areas of a project, including its details (project name and description), components, and access permissions.
- **Edit** - Users can modify a project's components, such as its workbench, but they cannot edit a project's access permissions or its details (project name and description).


3.7.2. Sharing access to a data science project

To enable your organization to work collaboratively, you can share access to your data science project with other users and groups.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have created a data science project.

Procedure


1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. From the list of data science projects, click the name of the data science project that you want to share access to.
A project details page opens.
3. Click the **Permissions** tab.
The **Permissions** page for the project opens.
4. Provide one or more users with access to the project.
 - a. In the **Users** section, click **Add user**.
 - b. In the **Name** field, enter the user name of the user whom you want to provide access to the project.
 - c. From the **Permissions** list, select one of the following access permission levels:
 - **Admin**: Users with this access level can edit project details and manage access to the project.
 - **Edit**: Users with this access level can view and edit project components, such as its workbenches, data connections, and storage.
 - d. To confirm your entry, click **Confirm** ().

- e. Optional: To add an additional user, click **Add user** and repeat the process.
5. Provide one or more OpenShift groups with access to the project.
 - a. In the **Groups** section, click **Add group**.
 - b. From the **Name** list, select a group to provide access to the project.



NOTE

If you do not have **cluster-admin** permissions, the **Name** list is not visible. Instead, an input field is displayed enabling you to configure group permissions.

- c. From the **Permissions** list, select one of the following access permission levels:
 - **Admin**: Groups with this access permission level can edit project details and manage access to the project.
 - **Edit**: Groups with this access permission level can view and edit project components, such as its workbenches, data connections, and storage.
- d. To confirm your entry, click **Confirm** ().
- e. Optional: To add an additional group, click **Add group** and repeat the process.

Verification

- Users to whom you provided access to the project can perform only the actions permitted by their access permission level.
- The **Users** and **Groups** sections on the **Permissions** tab show the respective users and groups that you provided with access to the project.

3.7.3. Updating access to a data science project


To change the level of collaboration on your data science project, you can update the access permissions of users and groups who have access to your project.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have created a data science project.
- You have previously shared access to your project with other users or groups.
- You have administrator permissions or you are the project owner.

Procedure


1. From the OpenShift Data Science dashboard, click **Data Science Projects**.

- The **Data science projects** page opens.
2. Click the name of the project that you want to change the access permissions of.
A project details page opens.
 3. Click the **Permissions** tab.
The **Permissions** page for the project opens.
 4. Update the user access permissions to the project.
 - a. In the **Name** field, update the user name of the user whom you want to provide access to the project.
 - b. From the **Permissions** list, update the user access permissions by selecting one of the following:
 - Admin: Users with this access level can edit project details and manage access to the project.
 - Edit: Users with this access level can view and edit project components, such as its workbenches, data connections, and storage.
 - c. To confirm the update to the entry, click **Confirm** ().
 5. Update the OpenShift groups access permissions to the project.
 - a. From the **Name** list, update the group that has access to the project by selecting another group from the list.



NOTE

If you do not have **cluster-admin** permissions, the **Name** list is not visible. Instead, you can configure group permissions in the input field that appears.

- b. From the **Permissions** list, update the group access permissions by selecting one of the following:
 - Admin: Groups with this access permission level can edit project details and manage access to the project.
 - Edit: Groups with this access permission level can view and edit project components, such as its workbenches, data connections, and storage.
- c. To confirm the update to the entry, click **Confirm** ().

Verification

- The **Users** and **Groups** sections on the **Permissions** tab show the respective users and groups whose project access permissions you changed.


3.7.4. Removing access to a data science project

If you no longer want to work collaboratively on your data science project, you can restrict access to your project by removing users and groups that you previously provided access to your project.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project.
- You have previously shared access to your project with other users or groups.
- You have administrator permissions or you are the project owner.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to change the access permissions of.
A project details page opens.
3. Click the **Permissions** tab.
The **Permissions** page for the project opens.
4. Click the action menu () beside the user or group whose access permissions you want to revoke and click **Delete**.

Verification

- Users whose access you have revoked can no longer perform the actions that were permitted by their access permission level.

3.8. VIEWING PYTHON PACKAGES INSTALLED ON YOUR NOTEBOOK SERVER

You can check which Python packages are installed on your notebook server and which version of the package you have by running the **pip** tool in a notebook cell.

Prerequisites

- Log in to Jupyter and open a notebook.

Procedure

1. Enter the following in a new cell in your notebook:

```
!pip list
```

2. Run the cell.

Verification

- The output shows an alphabetical list of all installed Python packages and their versions. For example, if you use this command immediately after creating a notebook server that uses the **Minimal** image, the first packages shown are similar to the following:

| Package | Version |
|-----------------|---------|
| ----- | |
| aiohttp | 3.7.3 |
| alembic | 1.5.2 |
| appdirs | 1.4.4 |
| argo-workflows | 3.6.1 |
| argon2-cffi | 20.1.0 |
| async-generator | 1.10 |
| async-timeout | 3.0.1 |
| attrdict | 2.0.1 |
| attrs | 20.3.0 |
| backcall | 0.2.0 |

Additional resources

- [Installing Python packages on your notebook server](#)

3.9. INSTALLING PYTHON PACKAGES ON YOUR NOTEBOOK SERVER

You can install Python packages that are not part of the default notebook server image by adding the package and the version to a **requirements.txt** file and then running the **pip install** command in a notebook cell.



NOTE

You can also install packages directly, but Red Hat recommends using a **requirements.txt** file so that the packages stated in the file can be easily re-used across different notebooks. In addition, using a **requirements.txt** file is also useful when using a S2I build to deploy a model.

Prerequisites

- Log in to Jupyter and open a notebook.

Procedure

1. Create a new text file using one of the following methods:
 - Click **+** to open a new launcher and click **Text file**.
 - Click **File** → **New** → **Text File**.
2. Rename the text file to **requirements.txt**.
 - a. Right-click on the name of the file and click **Rename Text**. The **Rename File** dialog opens.
 - b. Enter **requirements.txt** in the **New Name** field and click **Rename**.
3. Add the packages to install to the **requirements.txt** file.

altair

-

You can specify the exact version to install by using the `==` (equal to) operator, for example:

```
altair==4.1.0
```



NOTE

Red Hat recommends specifying exact package versions to enhance the stability of your notebook server over time. New package versions can introduce undesirable or unexpected changes in your environment's behavior.

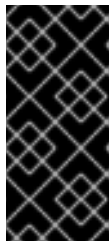
To install multiple packages at the same time, place each package on a separate line.

4. Install the packages in **requirements.txt** to your server using a notebook cell.

a. Create a new cell in your notebook and enter the following command:

```
!pip install -r requirements.txt
```

b. Run the cell by pressing Shift and Enter.



IMPORTANT

This command installs the package on your notebook server, but you must still run the **import** directive in a code cell to use the package in your code.

```
import altair
```

Verification

- Confirm that the packages in **requirements.txt** appear in the list of packages installed on the notebook server. See [Viewing Python packages installed on your notebook server](#) for details.

3.10. UPDATING NOTEBOOK SERVER SETTINGS BY RESTARTING YOUR SERVER

You can update the settings on your notebook server by stopping and relaunching the notebook server. For example, if your server runs out of memory, you can restart the server to make the container size larger.

Prerequisites

- A running notebook server.
- Log in to Jupyter.

Procedure

1. Click **File** → **Hub Control Panel**
The **Notebook server control panel** opens.
2. Click the **Stop notebook server** button.

The **Stop server** dialog opens.

3. Click **Stop server** to confirm your decision.
The **Start a notebook server** page opens.
4. Update the relevant notebook server settings and click **Start server**.

Verification

- The notebook server starts and contains your updated settings.

Additional resources

- [Launching Jupyter and starting a notebook server](#)

CHAPTER 4. WORKING WITH DATA SCIENCE PIPELINES

As a data scientist, you can enhance your data science projects on OpenShift Data Science by building portable machine learning (ML) workflows with data science pipelines, using Docker containers. This enables you to standardize and automate machine learning workflows to enable you to develop and deploy your data science models.

For example, the steps in a machine learning workflow might include items such as data extraction, data processing, feature extraction, model training, model validation, and model serving. Automating these activities enables your organization to develop a continuous process of retraining and updating a model based on newly received data. This can help address challenges related to building an integrated machine learning deployment and continuously operating it in production.

You can also use the Elyra JupyterLab extension to create and run data science pipelines within JupyterLab. For more information, see [Working with pipelines in JupyterLab](#).

A data science pipeline in OpenShift Data Science consists of the following components:

- Pipeline server: A server that is attached to your data science project and hosts your data science pipeline.
- Pipeline: A pipeline defines the configuration of your machine learning workflow and the relationship between each component in the workflow.
 - Pipeline code: A definition of your pipeline in a Tekton-formatted YAML file.
 - Pipeline graph: A graphical illustration of the steps executed in a pipeline run and the relationship between them.
- Pipeline run: An execution of your pipeline.
 - Triggered run: A previously executed pipeline run.
 - Scheduled run: A pipeline run scheduled to execute at least once.

This feature is based on Kubeflow Pipelines v1. Use the Kubeflow Pipelines SDK to build your data science pipeline in Python code. After you have built your pipeline, compile it into Tekton-formatted YAML code using kfp-tekton SDK (version 1.5.x only). The OpenShift Data Science user interface enables you to track and manage pipelines and pipeline runs.

Before you can use data science pipelines, you must install the OpenShift Pipelines operator. For more information about installing a compatible version of the OpenShift Pipelines operator, see [Red Hat OpenShift Pipelines release notes](#) and [Red Hat OpenShift Data Science: Supported Configurations](#).

You can store your pipeline artifacts in an Amazon Web Services (AWS) Simple Storage Service (S3) bucket so that you do not consume local storage. To do this, you must first configure write access to your S3 bucket on your AWS account.

4.1. MANAGING DATA SCIENCE PIPELINES

4.1.1. Configuring a pipeline server

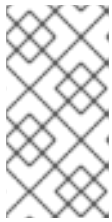
Before you can successfully create a pipeline in OpenShift Data Science, you must configure a pipeline server. This includes configuring where your pipeline artifacts and data are stored.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that you can add a pipeline server to.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to configure a pipeline server for.
A project details page opens.
3. In the **Pipelines** section, click **Create a pipeline server**.
The **Configure pipeline server** dialog appears.
4. In the **Object storage connection** section, to specify the S3-compatible data connection to store your pipeline artifacts, select one of the following sets of actions:



NOTE

After the pipeline server is created, the **/metadata** and **/artifacts** folders are automatically created in the default **root** folder. Therefore, you are not required to specify any storage directories when configuring a data connection for your pipeline server.

- Select **Existing data connection** to use a data connection that you previously defined. If you selected this option, from the **Name** list, select the name of the relevant data connection and skip to step 6.
 - Select **Create new data connection** to add a new data connection that your pipeline server can access.
5. If you selected **Create new data connection**, perform the following steps:
 - a. In the **Name** field, enter a name for the data connection.
 - b. In the **Access key** field, enter your access key ID for Amazon Web Services.
 - c. In the **Secret key** field, enter your secret access key for the account you specified.
 - d. Optional: In the **Endpoint** field, enter the endpoint of your AWS S3 storage.
 - e. Optional: In the **Region** field, enter the default region of your AWS account.
 - f. In the **Bucket** field, enter the name of the AWS S3 bucket.



IMPORTANT

If you are creating a new data connection, in addition to the other designated mandatory fields, the **AWS_S3_BUCKET** field is mandatory. If you specify incorrect data connection settings, you cannot update these settings on the same pipeline server. Therefore, you must delete the pipeline server and configure another one.

6. In the **Database** section, click **Show advanced database options** to specify the database to store your pipeline data and select one of the following sets of actions:
 - Select **Use default database stored on your cluster** to deploy a MariaDB database in your project.
 - Select **Connect to external MySQL database** to add a new connection to an external database that your pipeline server can access.
 - i. In the **Host** field, enter the database's host name.
 - ii. In the **Port** field, enter the database's port.
 - iii. In the **Username** field, enter the default user name that is connected to the database.
 - iv. In the **Password** field, enter the password for the default user account.
 - v. In the **Database** field, enter the database name.
7. Click **Configure**.

Verification

- The pipeline server that you configured is displayed in the **Pipelines** section on the project details page.
- The **Import pipeline** button is available in the **Pipelines** section on the project details page.

4.1.2. Defining a pipeline

The Kubeflow Pipelines SDK enables you to define end-to-end machine learning and data pipelines. Use the Kubeflow Pipelines SDK to build your data science pipeline in Python code. After you have built your pipeline, compile it into Tekton-formatted YAML code using kfp-tekton SDK (version 1.5.x only). After defining the pipeline, you can import the YAML file to the OpenShift Data Science dashboard to enable you to configure its execution settings. For more information about installing and using Kubeflow Pipelines SDK for Tekton, see [Kubeflow Pipelines SDK for Tekton](#).

You can also use the Elyra JupyterLab extension to create and run data science pipelines within JupyterLab. For more information on creating pipelines in JupyterLab, see [Working with pipelines in JupyterLab](#). For more information on the Elyra JupyterLab extension, see [Elyra Documentation](#).

Additional resources

- [Kubeflow Pipelines SDK for Tekton](#)
- [KFP Tekton samples and compiler samples](#)
- [Kubeflow Pipelines v1 Documentation](#)

- [Elyra Documentation](#)

4.1.3. Importing a data science pipeline

To help you begin working with data science pipelines in OpenShift Data Science, you can import a YAML file containing your pipeline's code to an active pipeline server. This file contains a KubeFlow pipeline compiled with the Tekton compiler. After you have imported the pipeline to a pipeline server, you can execute the pipeline by creating a pipeline run.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have previously created a data science project that is available and contains a configured pipeline server.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Pipelines**. The **Pipelines** page opens.
2. From the **Project** list, select the project that you want to import a pipeline to.
3. Click **Import pipeline**. The **Import pipeline** dialog opens.
4. Enter the details for the pipeline that you are importing.
 - a. In the **Pipeline name** field, enter a name for the pipeline that you are importing.
 - b. In the **Pipeline description** field, enter a description for the pipeline that you are importing.
 - c. Click **Upload**. Alternatively, drag the file from your local machine's file system and drop it in the designated area in the **Import pipeline** dialog. A file browser opens.
 - d. Navigate to the file containing the pipeline code and click **Select**.
 - e. Click **Import pipeline**.

Verification

- The pipeline that you imported is displayed on the **Pipelines** page.


4.1.4. Downloading a data science pipeline

To make further changes to a data science pipeline that you previously uploaded to OpenShift Data Science, you can download the pipeline's code from the user interface.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have previously created a data science project that is available and contains a configured pipeline server.
- You have created and imported a pipeline to an active pipeline server that is available to download.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Pipelines**.
The **Pipelines** page opens.
2. From the **Project** list, select the project whose pipeline that you want to download.
3. In the **Pipeline name** column, click the name of the pipeline that you want to download.
The **Pipeline details** page opens displaying the **Graph** tab.
4. Click the **YAML** tab.
The page reloads to display an embedded YAML editor showing the pipeline code.
5. Click the **Download** button () to download the YAML file containing your pipeline's code to your local machine.

Verification

- The pipeline code is downloaded to your browser's default directory for downloaded files.

4.1.5. Deleting a data science pipeline


You can delete data science pipelines so that they do not appear on the OpenShift Data Science **Pipelines** page.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- There are active pipelines available on the **Pipelines** page.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Pipelines**.
The **Pipelines** page opens.

2. From the **Project** list, select the project that contains the pipeline that you want to delete.
3. Click the action menu () beside the pipeline that you want to delete and click **Delete pipeline**. The **Delete pipeline** dialog opens.
4. Enter the pipeline name in the text field to confirm that you intend to delete it.
5. Click **Delete pipeline**.

Verification

- The data science pipeline that you deleted is no longer displayed on the **Pipelines** page.

4.1.6. Deleting a pipeline server

After you have finished running your data science pipelines, you can delete the pipeline server. Deleting a pipeline server automatically deletes all of its associated pipelines and runs. If your pipeline data is stored in a database, the database is also deleted along with its meta-data. In addition, after deleting a pipeline server, you cannot create new pipelines or pipeline runs until you create another pipeline server.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have previously created a data science project that is available and contains a pipeline server.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Pipelines**. The **Pipelines** page opens.
2. From the **Project** list, select the project whose pipeline server you want to delete.
3. From the **Pipeline server actions** list, select **Delete pipeline server**. The **Delete pipeline server** dialog opens.
4. Enter the pipeline server's name in the text field to confirm that you intend to delete it.
5. Click **Delete**.

Verification

- Pipelines previously assigned to the deleted pipeline server are no longer displayed on the **Pipelines** page for the relevant data science project.
- Pipeline runs previously assigned to the deleted pipeline server are no longer displayed on the **Runs** page for the relevant data science project.

4.1.7. Viewing the details of a pipeline server

You can view the details of pipeline servers configured in OpenShift Data Science, such as the pipeline's data connection details and where its data is stored.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- You have previously created a data science project that contains an active and available pipeline server.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Pipelines**. The **Pipelines** page opens.
2. From the **Project** list, select the project whose pipeline server you want to view.
3. From the **Pipeline server actions** list, select **View pipeline server configuration**.
4. When you have finished inspecting the pipeline server's details, click **Done**.

Verification

- You can view the relevant pipeline server's details in the **View pipeline server** dialog.

4.1.8. Viewing existing pipelines


You can view the details of pipelines that you have imported to Red Hat OpenShift Data Science, such as the pipeline's last run, when it was created, and the pipeline's executed runs.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have previously created a data science project that is available and contains a pipeline server.
- You have imported a pipeline to an active and available pipeline server.
- The pipeline you imported is available, or there are other previously imported pipelines available to view.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Pipelines**. The **Pipelines** page opens.

2. From the **Project** list, select the relevant project whose pipelines you want to view.
3. Study the pipelines on the list.
4. Optional: Click **Expand** () on the relevant row to view the pipeline's executed runs. If the pipeline does not contain any runs, click **Create run** to create one.

Verification

- A list of previously created data science pipelines is displayed on the **Pipelines** page.

4.2. MANAGING PIPELINE RUNS

4.2.1. Overview of pipeline runs

A pipeline run is a single execution of a data science pipeline. As data scientist, you can use OpenShift Data Science to define, manage, and track executions of a data science pipeline. You can view a record of your data science project's previously executed and scheduled runs from the **Runs** page in the OpenShift Data Science user interface.

Runs are intended for portability. Therefore, you can clone your pipeline runs to reproduce and scale them accordingly, or delete them when you no longer require them. You can configure a run to execute only once immediately after creation or on a recurring basis. Recurring runs consist of a copy of a pipeline with all of its parameter values and a run trigger. A run trigger indicates when a recurring run executes. You can define the following run triggers:

- **Periodic**: used for scheduling runs to execute in intervals.
- **Cron**: used for scheduling runs as a cron job.

When executed, you can track the run's progress from the run's **Details** page on the OpenShift Data Science user interface. From here, you can view the run's graph, and output artifacts.

A pipeline run can be classified as the following:

- **Scheduled run**: A pipeline run scheduled to execute at least once
- **Triggered run**: A previously executed pipeline run.

4.2.2. Scheduling a pipeline run



You can instantiate a single execution of a pipeline by scheduling a pipeline run. In OpenShift Data Science, you can schedule runs to occur at specific times or execute them immediately after creation.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have previously created a data science project that is available and contains a configured pipeline server.

- You have imported a pipeline to an active pipeline server.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Pipelines**.
The **Pipelines** page opens.
2. Click the action menu () beside the relevant pipeline and click **Create run**.
The **Create run** page opens.
3. From the **Project** list, select the project that contains the pipeline you want to create a run for.
4. In the **Name** field, enter a name for the run.
5. In the **Description** field, enter a description for the run.
6. From the **Pipeline** list, select the pipeline to create a run for. Alternatively, to upload a new pipeline, click **Upload new pipeline** and fill in the relevant fields in the **Import pipeline** dialog.
7. Configure the run type by performing one of the following sets of actions:
 - Select **Run once immediately after creation** to specify the run executes once, and immediately after its creation.
 - Select **Schedule recurring run** to schedule the run to recur.
 - i. Configure the run's trigger type.
 - A. Select **Periodic** and select the execution frequency from the list.
 - B. Select **Cron** to specify the execution schedule in **cron** format. This creates a cron job to execute the run. Click the **Copy** button () to copy the cron job schedule to the clipboard. The field furthest to the left represents seconds. For more information about scheduling tasks using the supported **cron** format, see [Cron Expression Format](#).
 - ii. Configure the run's duration.
 - A. Select the **Start date** check box to specify a start date for the run. Select the run's start date using the **Calendar** and the start time from the list of times.
 - B. Select the **End date** check box to specify an end date for the run. Select the run's end date using the **Calendar** and the end time from the list of times.
8. Configure the input parameters for the run by selecting the parameters from the list.
9. Click **Create**.

Verification

- The pipeline run that you created is shown in the **Scheduled** tab on the **Runs** page.



4.2.3. Cloning a scheduled pipeline run

To make it easier to schedule runs to execute as part of your pipeline configuration, you can duplicate existing scheduled runs by cloning them.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have previously created a data science project that is available and contains a configured pipeline server.
- You have imported a pipeline to an active pipeline server.
- You have previously scheduled a run that is available to clone.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Runs**.
The **Runs** page opens.
2. Click the action menu () beside the relevant run and click **Clone**.
The **Clone** page opens.
3. From the **Project** list, select the project that contains the pipeline whose run that you want to clone.
4. In the **Name** field, enter a name for the run that you want to clone.
5. In the **Description** field, enter a description for the run that you want to clone.
6. From the **Pipeline** list, select the pipeline containing the run that you want to clone.
7. To configure the run type for the run that you are cloning, in the **Run type** section, perform one of the following sets of actions:
 - Select **Run once immediately after create** to specify the run that you are cloning executes once, and immediately after its creation. If you selected this option, skip to step 10.
 - Select **Schedule recurring run** to schedule the run that you are cloning to recur.
8. If you selected **Schedule recurring run** in the previous step, to configure the trigger type for the run, perform one of the following actions:
 - Select **Periodic** and select the execution frequency from the **Run every** list.
 - Select **Cron** to specify the execution schedule in **cron** format. This creates a cron job to execute the run. Click the **Copy** button () to copy the cron job schedule to the clipboard. The field furthest to the left represents seconds. For more information about scheduling tasks using the supported **cron** format, see [Cron Expression Format](#).
9. If you selected **Schedule recurring run** in step 7, configure the duration for the run that you are cloning.
 - a. Select the **Start date** check box to specify a start date for the run. Select the start date using the calendar tool and the start time from the list of times.

- b. Select the **End date** check box to specify an end date for the run. Select the end date using the calendar tool and the end time from the list of times.
10. In the **Parameters** section, configure the input parameters for the run that you are cloning by selecting the appropriate parameters from the list.
11. Click **Create**.

Verification

- The pipeline run that you cloned is shown in the **Scheduled** tab on the **Runs** page.

4.2.4. Stopping a triggered pipeline run

If you no longer require a triggered pipeline run to continue executing, you can stop the run before its defined end date.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- There is a previously created data science project available that contains a pipeline server.
- You have imported a pipeline to an active and available pipeline server.
- You have previously triggered a pipeline run.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Runs**.
The **Runs** page opens.
2. From the **Project** list, select the project that contains the pipeline whose triggered run you want to stop.
The page refreshes to show the pipeline's triggered runs on the **Triggered** tab.
3. Click the action menu (**⋮**) beside the triggered run that you want to delete and click **Stop**.
There might be a short delay while the run stops.

Verification

- A list of previously triggered runs are displayed in the **Triggered** tab on the **Runs** page.

4.2.5. Deleting a scheduled pipeline run

To discard pipeline runs that you previously scheduled, but no longer require, you can delete them so that they do not appear on the **Runs** page.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have previously created a data science project that is available and contains a configured pipeline server.
- You have imported a pipeline to an active pipeline server.
- You have previously scheduled a run that is available to delete.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines** → **Runs**.
The **Runs** page opens.
2. From the **Project** list, select the project that contains the pipeline whose scheduled run you want to delete.
The page refreshes to show the pipeline's scheduled runs on the **Scheduled** tab.
3. Click the action menu (**⋮**) beside the scheduled run that you want to delete and click **Delete**.
The **Delete scheduled run** dialog opens.
4. Enter the run's name in the text field to confirm that you intend to delete it.
5. Click **Delete scheduled run**.

Verification

- The run that you deleted is no longer displayed on the **Scheduled** tab.

4.2.6. Deleting a triggered pipeline run

To discard pipeline runs that you previously executed, but no longer require a record of, you can delete them so that they do not appear on the **Triggered** tab on the **Runs** page.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have previously created a data science project that is available and contains a configured pipeline server.
- You have imported a pipeline to an active pipeline server.
- You have previously executed a run that is available to delete.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Runs**.
The **Runs** page opens.
2. From the **Project** list, select the project that contains the pipeline whose triggered run you want to delete.
The page refreshes to show the pipeline's triggered runs on the **Triggered** tab.
3. Click the action menu (**⋮**) beside the triggered run that you want to delete and click **Delete**.
The **Delete triggered run** dialog opens.
4. Enter the run's name in the text field to confirm that you intend to delete it.
5. Click **Delete triggered run**.

Verification

- The run that you deleted is no longer displayed on the **Triggered** tab.

4.2.7. Viewing scheduled pipeline runs

You can view a list of pipeline runs that are scheduled for execution in OpenShift Data Science. From this list, you can view details relating to your pipeline's runs, such as the pipeline that the run belongs to. You can also view the run's status, execution frequency, and schedule.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- You have installed the OpenShift Pipelines operator.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have previously created a data science project that is available and contains a pipeline server.
- You have imported a pipeline to an active and available pipeline server.
- You have created and scheduled a pipeline run.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Runs**.
The **Runs** page opens.
2. From the **Project** list, select the project whose scheduled pipeline runs you want to view.
3. Click the **Scheduled** tab.
4. Study the table showing a list of scheduled runs.
After a run has been scheduled, the run's status is displayed in the **Status** column in the table, indicating whether the run is ready for execution or unavailable for execution. To enable or disable a previously imported notebook image, on the row containing the relevant notebook image, click the toggle in the **Enabled** column.

Verification

- A list of scheduled runs are displayed in the **Scheduled** tab on the **Runs** page.

4.2.8. Viewing triggered pipeline runs

You can view a list of pipeline runs that were previously executed in OpenShift Data Science. From this list, you can view details relating to your pipeline's runs, such as the pipeline that the run belongs to, along with the run's status, duration, and execution start time.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- You have installed the OpenShift Pipelines operator.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have previously created a data science project that is available and has a pipeline server.
- You have imported a pipeline to an active and available pipeline server.
- You have previously triggered a pipeline run.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines → Runs**.
The **Runs** page opens.
2. From the **Project** list, select the project whose previously executed pipeline runs you want to view.
The **Run details** page opens.
3. Click the **Triggered** tab.
A table opens that shows list of triggered runs. After a run has completed its execution, the run's status is displayed in the **Status** column in the table, indicating whether the run has succeeded or failed.

Verification

- A list of previously triggered runs are displayed in the **Triggered** tab on the **Runs** page.

4.2.9. Viewing the details of a pipeline run


To gain a clearer understanding of your pipeline runs, you can view the details of a previously triggered pipeline run, such as its graph, execution details, and run output.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.

- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have previously created a data science project that is available and contains a pipeline server.
- You have imported a pipeline to an active and available pipeline server.
- You have previously triggered a pipeline run.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Pipelines** → **Pipelines**. The **Pipelines** page opens.
2. From the **Project** list, select the project whose pipeline runs you want to view.
3. For a pipeline that you want to see run details for, click **Expand** ().
4. From the **Runs** section, click the name of the run that you want to view the details of. The **Run details** page opens.

Verification

- On the **Run details** page, you can view the run's graph, execution details, input parameters, and run output.

4.3. WORKING WITH PIPELINES IN JUPYTERLAB

4.3.1. Overview of pipelines in JupyterLab

You can use Elyra to create visual end-to-end pipeline workflows in JupyterLab. Elyra is an extension for JupyterLab that provides you with a Pipeline Editor to create pipeline workflows that can be executed in OpenShift Data Science.

Before you can work with pipelines in JupyterLab, you must install the OpenShift Pipelines operator. For more information about installing a compatible version of the OpenShift Pipelines operator, see [Red Hat OpenShift Pipelines release notes](#) and [Red Hat OpenShift Data Science: Supported Configurations](#).

You can access the Elyra extension within JupyterLab when you create the most recent version of one of the following notebook images:

- Standard Data Science
- PyTorch
- TensorFlow
- TrustyAI

As you can use the Pipeline Editor to visually design your pipelines, minimal coding is required to create and run pipelines. For more information about Elyra, see [Elyra Documentation](#). For more information on the Pipeline Editor, see [Visual Pipeline Editor](#). After you have created your pipeline, you can run it locally in JupyterLab, or remotely using data science pipelines in OpenShift Data Science.

The pipeline creation process consists of the following tasks:

- Create a data science project that contains a workbench.
- Create a pipeline server.
- Create a new pipeline in the Pipeline Editor in JupyterLab.
- Develop your pipeline by adding Python notebooks or Python scripts and defining their runtime properties.
- Define execution dependencies.
- Run or export your pipeline.

Before you can run a pipeline in JupyterLab, your pipeline instance must contain a runtime configuration. A runtime configuration defines connectivity information for your pipeline instance and S3-compatible cloud storage.

If you create a workbench as part of a data science project, a default runtime configuration is created automatically. However, if you create a notebook from the Jupyter tile in the OpenShift Data Science dashboard, you must create a runtime configuration before you can run your pipeline in JupyterLab. For more information about runtime configurations, see [Runtime Configuration](#). As a prerequisite, before you create a workbench, ensure that you have created and configured a pipeline server within the same data science project as your workbench.

You can use S3-compatible cloud storage to make data available to your notebooks and scripts while they are executed. Your cloud storage must be accessible from the machine in your deployment that runs JupyterLab and from the cluster that hosts Data Science Pipelines. Before you create and run pipelines in JupyterLab, ensure that you have your s3-compatible storage credentials readily available.

Additional resources

- [Elyra Documentation](#)
- [Visual Pipeline Editor](#)
- [Runtime Configuration](#).

4.3.2. Accessing the pipeline editor

You can use Elyra to create visual end-to-end pipeline workflows in JupyterLab. Elyra is an extension for JupyterLab that provides you with a Pipeline Editor to create pipeline workflows that can be executed in OpenShift Data Science.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that contains a workbench.

- You have created and configured a pipeline server within the data science project that contains your workbench.
- You have created and launched a Jupyter server from a notebook image that contains the Elyra extension (Standard data science, TensorFlow, TrustyAI, or PyTorch).
- You have access to S3-compatible storage.

Procedure

1. After you open JupyterLab, confirm that the JupyterLab launcher is automatically displayed.
2. In the **Elyra** section of the JupyterLab launcher, click the **Pipeline Editor** tile. The Pipeline Editor opens.

Verification

- You can view the Pipeline Editor in JupyterLab.



4.3.3. Creating a runtime configuration

If you create a workbench as part of a data science project, a default runtime configuration is created automatically. However, if you create a notebook from the Jupyter tile in the OpenShift Data Science dashboard, you must create a runtime configuration before you can run your pipeline in JupyterLab. This enables you to specify connectivity information for your pipeline instance and S3-compatible cloud storage.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have access to S3-compatible cloud storage.
- You have created a data science project that contains a workbench.
- You have created and configured a pipeline server within the data science project that contains your workbench.
- You have created and launched a Jupyter server from a notebook image that contains the Elyra extension (Standard data science, TensorFlow, TrustyAI, or PyTorch).

Procedure

1. In the left sidebar of JupyterLab, click **Runtimes** ().
2. Click the **Create new runtime configuration** button (). The **Add new Data Science Pipelines runtime configuration** page opens.
3. Fill in the relevant fields to define your runtime configuration.

- a. In the **Display Name** field, enter a name for your runtime configuration.
- b. Optional: In the **Description** field, enter a description to define your runtime configuration.
- c. Optional: In the **Tags** field, click **Add Tag** to define a category for your pipeline instance. Enter a name for the tag and press Enter.
- d. Define the credentials of your data science pipeline:
 - i. In the **Data Science Pipelines API Endpoint** field, enter the API endpoint of your data science pipeline. Do not specify the pipelines namespace in this field.
 - ii. In the **Public Data Science Pipelines API Endpoint** field, enter the public API endpoint of your data science pipeline.



IMPORTANT

You can obtain the Data Science Pipelines API endpoint from the **Data Science Pipelines → Runs** page in the dashboard. Copy the relevant endpoint and enter it in the **Public Data Science Pipelines API Endpoint** field.

- iii. Optional: In the **Data Science Pipelines User Namespace** field, enter the relevant user namespace to run pipelines.
- iv. From the **Data Science Pipelines engine** list, select **Tekton**.
- v. From the **Authentication Type** list, select the authentication type required to authenticate your pipeline.



IMPORTANT

If you created a notebook directly from the Jupyter tile on the dashboard, select **EXISTING_BEARER_TOKEN** from the **Authentication Type** list.

- vi. In the **Data Science Pipelines API Endpoint Username** field, enter the user name required for the authentication type.
- vii. In the **Data Science Pipelines API Endpoint Password Or Token** enter the password or token required for the authentication type.




IMPORTANT

To obtain the Data Science Pipelines API endpoint token, in the upper-right corner of the OpenShift web console, click your user name and select **Copy login command**. After you have logged in, click **Display token** and copy the value of **--token=** from the **Log in with this token** command.

- e. Define the connectivity information of your S3-compatible storage:
 - i. In the **Cloud Object Storage Endpoint** field, enter the endpoint of your S3-compatible storage. For more information about Amazon s3 endpoints, see [Amazon Simple Storage Service endpoints and quotas](#).

- ii. Optional: In the **Public Cloud Object Storage Endpoint** field, enter the URL of your S3-compatible storage.
 - iii. In the **Cloud Object Storage Bucket Name** field, enter the name of the bucket where your pipeline artifacts are stored. If the bucket name does not exist, it is created automatically.
 - iv. From the **Cloud Object Storage Authentication Type** list, select the authentication type required to access to your S3-compatible cloud storage. If you use AWS S3 buckets, select **KUBERNETES_SECRET** from the list.
 - v. In the **Cloud Object Storage Credentials Secret** field, enter the secret that contains the storage user name and password. This secret is defined in the relevant user namespace, if applicable. In addition, it must be stored on the cluster that hosts your pipeline runtime.
 - vi. In the **Cloud Object Storage Username** field, enter the user name to connect to your S3-compatible cloud storage, if applicable. If you use AWS S3 buckets, enter your AWS Secret Access Key ID.
 - vii. In the **Cloud Object Storage Password** field, enter the password to connect to your S3-compatible cloud storage, if applicable. If you use AWS S3 buckets, enter your AWS Secret Access Key.
- f. Click **Save & Close**.

Verification

- The runtime configuration that you created is shown in the **Runtimes** tab () in the left sidebar of JupyterLab.

4.3.4. Updating a runtime configuration



To ensure that your runtime configuration is accurate and updated, you can change the settings of an existing runtime configuration.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have access to S3-compatible storage.
- You have created a data science project that contains a workbench.
- You have created and configured a pipeline server within the data science project that contains your workbench.
- A previously created runtime configuration is available in the JupyterLab interface.

- You have created and launched a Jupyter server from a notebook image that contains the Elyra extension (Standard data science, TensorFlow, TrustyAI, or PyTorch).

Procedure

1. In the left sidebar of JupyterLab, click **Runtimes** ().
2. Hover the cursor over the runtime configuration that you want to update and click the **Edit** button ().
The **Data Science Pipelines runtime configuration** page opens.
3. Fill in the relevant fields to update your runtime configuration.
 - a. In the **Display Name** field, update name for your runtime configuration, if applicable.
 - b. Optional: In the **Description** field, update the description of your runtime configuration, if applicable.
 - c. Optional: In the **Tags** field, click **Add Tag** to define a category for your pipeline instance. Enter a name for the tag and press Enter.
 - d. Define the credentials of your data science pipeline:
 - i. In the **Data Science Pipelines API Endpoint** field, update the API endpoint of your data science pipeline, if applicable. Do not specify the pipelines namespace in this field.
 - ii. In the **Public Data Science Pipelines API Endpoint** field, update the API endpoint of your data science pipeline, if applicable.
 - iii. Optional: In the **Data Science Pipelines User Namespace** field, update the relevant user namespace to run pipelines, if applicable.
 - iv. From the **Data Science Pipelines engine** list, select **Tekton**.
 - v. From the **Authentication Type** list, select a new authentication type required to authenticate your pipeline, if applicable.



IMPORTANT

If you created a notebook directly from the Jupyter tile on the dashboard, select **EXISTING_BEARER_TOKEN** from the **Authentication Type** list.

- vi. In the **Data Science Pipelines API Endpoint Username** field, update the user name required for the authentication type, if applicable.
- vii. In the **Data Science Pipelines API Endpoint Password Or Token** field, update the password or token required for the authentication type, if applicable.




IMPORTANT

To obtain the Data Science Pipelines API endpoint token, in the upper-right corner of the OpenShift web console, click your user name and select **Copy login command**. After you have logged in, click **Display token** and copy the value of **--token=** from the **Log in with this token** command.

- e. Define the connectivity information of your S3-compatible storage:
 - i. In the **Cloud Object Storage Endpoint** field, update the endpoint of your S3-compatible storage, if applicable. For more information about Amazon s3 endpoints, see [Amazon Simple Storage Service endpoints and quotas](#) .
 - ii. Optional: In the **Public Cloud Object Storage Endpoint** field, update the URL of your S3-compatible storage, if applicable.
 - iii. In the **Cloud Object Storage Bucket Name** field, update the name of the bucket where your pipeline artifacts are stored, if applicable. If the bucket name does not exist, it is created automatically.
 - iv. From the **Cloud Object Storage Authentication Type** list, update the authentication type required to access to your S3-compatible cloud storage, if applicable. If you use AWS S3 buckets, you must select **USER_CREDENTIALS** from the list.
 - v. Optional: In the **Cloud Object Storage Credentials Secret** field, update the secret that contains the storage user name and password, if applicable. This secret is defined in the relevant user namespace. You must save the secret on the cluster that hosts your pipeline runtime.
 - vi. Optional: In the **Cloud Object Storage Username** field, update the user name to connect to your S3-compatible cloud storage, if applicable. If you use AWS S3 buckets, update your AWS Secret Access Key ID.
 - vii. Optional: In the **Cloud Object Storage Password** field, update the password to connect to your S3-compatible cloud storage, if applicable. If you use AWS S3 buckets, update your AWS Secret Access Key.
- f. Click **Save & Close**

Verification

- The runtime configuration that you updated is shown in the **Runtimes** tab () in the left sidebar of JupyterLab.

4.3.5. Deleting a runtime configuration



After you have finished using your runtime configuration, you can delete it from the JupyterLab interface. After deleting a runtime configuration, you cannot run pipelines in JupyterLab until you create another runtime configuration.

Prerequisites


- You have installed the OpenShift Pipelines operator.

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that contains a workbench.
- You have created and configured a pipeline server within the data science project that contains your workbench.
- A previously created runtime configuration is visible in the JupyterLab interface.
- You have created and launched a Jupyter server from a notebook image that contains the Elyra extension (Standard data science, TensorFlow, TrustyAI, or PyTorch).

Procedure

1. In the left sidebar of JupyterLab, click **Runtimes** ().
2. Hover the cursor over the runtime configuration that you want to delete and click the **Delete Item** button ().
A dialog box appears prompting you to confirm the deletion of your runtime configuration.
3. Click **OK**.

Verification

- The runtime configuration that you deleted is no longer shown in the **Runtimes** tab () in the left sidebar of JupyterLab.



4.3.6. Duplicating a runtime configuration

To prevent you from re-creating runtime configurations with similar values in their entirety, you can duplicate an existing runtime configuration in the JupyterLab interface.


Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that contains a workbench.
- You have created and configured a pipeline server within the data science project that contains your workbench.
- A previously created runtime configuration is visible in the JupyterLab interface.
- You have created and launched a Jupyter server from a notebook image that contains the Elyra extension (Standard data science, TensorFlow, TrustyAI, or PyTorch).

Procedure

1. In the left sidebar of JupyterLab, click **Runtimes** ().
2. Hover the cursor over the runtime configuration that you want to duplicate and click the **Duplicate** button ().

Verification

- The runtime configuration that you duplicated is shown in the **Runtimes** tab () in the left sidebar of JupyterLab.

4.3.7. Running a pipeline in JupyterLab

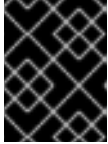
You can run pipelines that you have created in JupyterLab from the Pipeline Editor user interface. Before you can run a pipeline, you must create a data science project and a pipeline server. After you create a pipeline server, you must create a workbench within the same project as your pipeline server. Your pipeline instance in JupyterLab must contain a runtime configuration. If you create a workbench as part of a data science project, a default runtime configuration is created automatically. However, if you create a notebook from the Jupyter tile in the OpenShift Data Science dashboard, you must create a runtime configuration before you can run your pipeline in JupyterLab. A runtime configuration defines connectivity information for your pipeline instance and S3-compatible cloud storage.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have access to S3-compatible storage.
- You have created a pipeline in JupyterLab.
- You have opened your pipeline in the Pipeline Editor in JupyterLab.
- Your pipeline instance contains a runtime configuration.
- You have created and configured a pipeline server within the data science project that contains your workbench.
- You have created and launched a Jupyter server from a notebook image that contains the Elyra extension (Standard data science, TensorFlow, TrustyAI, or PyTorch).

Procedure

1. In the Pipeline Editor user interface, click **Run Pipeline** ().
The **Run Pipeline** dialog appears. The **Pipeline Name** field is automatically populated with the pipeline file name.



IMPORTANT

You must enter a unique pipeline name. The pipeline name that you enter must not match the name of any previously executed pipelines.

2. Define the settings for your pipeline run.
 - a. From the **Runtime Configuration** list, select the relevant runtime configuration to run your pipeline.
 - b. Optional: Configure your pipeline parameters, if applicable. If your pipeline contains nodes that reference pipeline parameters, you can change the default parameter values. If a parameter is required and has no default value, you must enter a value.
3. Click **OK**.

Verification

- You can view the output artifacts of your pipeline run. The artifacts are stored in your designated object storage bucket.

4.3.8. Exporting a pipeline in JupyterLab

You can export pipelines that you have created in JupyterLab. When you export a pipeline, the pipeline is prepared for later execution, but is not uploaded or executed immediately. During the export process, any package dependencies are uploaded to S3-compatible storage. Also, pipeline code is generated for the target runtime.


Before you can export a pipeline, you must create a data science project and a pipeline server. After you create a pipeline server, you must create a workbench within the same project as your pipeline server. In addition, your pipeline instance in JupyterLab must contain a runtime configuration. If you create a workbench as part of a data science project, a default runtime configuration is created automatically. However, if you create a notebook from the Jupyter tile in the OpenShift Data Science dashboard, you must create a runtime configuration before you can export your pipeline in JupyterLab. A runtime configuration defines connectivity information for your pipeline instance and S3-compatible cloud storage.

Prerequisites

- You have installed the OpenShift Pipelines operator.
- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that contains a workbench.
- You have created and configured a pipeline server within the data science project that contains your workbench.
- You have access to S3-compatible storage.
- You have created a pipeline in JupyterLab.
- You have opened your pipeline in the Pipeline Editor in JupyterLab.

- Your pipeline instance contains a runtime configuration.
- You have created and launched a Jupyter server from a notebook image that contains the Elyra extension (Standard data science, TensorFlow, TrustyAI, or PyTorch).

Procedure

1. In the Pipeline Editor user interface, click **Export Pipeline** ().
The **Export Pipeline** dialog appears. The **Pipeline Name** field is automatically populated with the pipeline file name.
2. Define the settings to export your pipeline.
 - a. From the **Runtime Configuration** list, select the relevant runtime configuration to export your pipeline.
 - b. From the **Export Pipeline as** select an appropriate file format
 - c. In the **Export Filename** field, enter a file name for the exported pipeline.
 - d. Select the **Replace if file already exists** check box to replace an existing file of the same name as the pipeline you are exporting.
 - e. Optional: Configure your pipeline parameters, if applicable. If your pipeline contains nodes that reference pipeline parameters, you can change the default parameter values. If a parameter is required and has no default value, you must enter a value.
3. Click **OK**.

Verification

- You can view the file containing the pipeline that you exported in your designated object storage bucket.

4.4. ADDITIONAL RESOURCES

- [Kubeflow Pipelines v1 Documentation](#)
- [Working with pipelines in JupyterLab](#).

CHAPTER 5. MODEL SERVING ON OPENSIFT DATA SCIENCE

As a data scientist, you can deploy your trained machine-learning models to serve intelligent applications in production. After you have deployed your model, applications can send requests to the model using its deployed API endpoint.

5.1. CONFIGURING MODEL SERVERS

5.1.1. Adding a model server to your data science project

Before you can successfully deploy a data science model on OpenShift Data Science, you must configure a model server. This includes configuring the number of replicas to deploy, the server size, the token authorization, and how to access the project. If you require extra power for use with large datasets, you can assign accelerators to your model server to optimize performance.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you use specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.
- You have created a data science project that you can add a model server to.
- If you want to use a custom model-serving runtime for your model server, you have added and enabled the runtime. See [Adding a custom model-serving runtime](#).
- If you want to use graphics processing units (GPUs) with your model server, you have enabled GPU support in OpenShift Data Science. See [Enabling GPU support in OpenShift Data Science](#).

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project that you want to configure a model server for.
A project details page opens.
3. In the **Models and model servers** section, click **Add server**.
The **Add model server** dialog opens.
4. In the **Model server name** field, enter a unique name for the model server.
5. From the **Serving runtime** list, select a model-serving runtime that is installed and enabled in your OpenShift Data Science deployment.
6. In the **Number of model replicas to deploy** field, specify a value.
7. From the **Model server size** list, select one of the following server sizes:
 - Small
 - Medium

- Large
 - Custom
8. Optional: If you selected **Custom** in the preceding step, configure the following settings in the **Model server size** section to customize your model server:
 - a. In the **CPUs requested** field, specify a number of CPUs to use with your model server. Use the list beside this field to specify the value in cores or millicores.
 - b. In the **CPU limit** field, specify the maximum number of CPUs to use with your model server. Use the list beside this field to specify the value in cores or millicores.
 - c. In the **Memory requested** field, specify the requested memory for the model server in gibibytes (Gi).
 - d. In the **Memory limit** field, specify the maximum memory limit for the model server in gibibytes (Gi).



IMPORTANT

OpenShift Data Science includes two versions of the OpenVINO Model Server (OVMS) runtime by default; a version that supports GPUs and one that does not. To use GPUs, from the **Serving runtime** list, you must select the version whose display name includes **Supports GPUs**.

If you are using a *custom* model-serving runtime with your model server, you must ensure that your custom runtime supports GPUs and is appropriately configured to use them.

9. Optional: From the **Accelerator** list, select an accelerator.
 - a. If you selected an accelerator in the preceding step, specify the number of accelerators to use.
10. Optional: In the **Model route** section, select the **Make deployed models available through an external route** check box to make your deployed models available to external clients.
11. Optional: In the **Token authorization** section, select the **Require token authentication** check box to require token authentication for your model server. To finish configuring token authentication, perform the following actions:
 - a. In the **Service account name** field, enter a service account name for which the token will be generated. The generated token is created and displayed in the **Token secret** field when the model server is configured.
 - b. To add an additional service account, click **Add a service account** and enter another service account name.
12. Click **Add**.

Verification

- The model server that you configured appears in the **Models and model servers** section of the project details page.

5.1.2. Adding a custom model-serving runtime

A model-serving runtime provides integration with a specified model server and the model frameworks that it supports. By default, Red Hat OpenShift Data Science includes the OpenVINO Model Server runtime. However, if this runtime doesn't meet your needs (it doesn't support a particular model framework, for example), you might want to add your own, custom runtimes.

As an administrator, you can use the OpenShift Data Science interface to add and enable custom model-serving runtimes. You can then choose from your enabled runtimes when you create a new model server.

Prerequisites

- You have logged in to OpenShift Data Science as an administrator.
- You are familiar with how to [add a model server to your project](#) . When you have added a custom model-serving runtime, you must configure a new model server to use the runtime.
- You have reviewed the example runtimes in the [kserve/modelmesh-serving](#) repository. You can use these examples as *starting points*. However, each runtime requires some further modification before you can deploy it in OpenShift Data Science. The required modifications are described in the following procedure.



NOTE

OpenShift Data Science includes the OpenVINO Model Server model-serving runtime by default. You do not need to add this runtime to OpenShift Data Science.

Procedure

1. From the OpenShift Data Science dashboard, click **Settings > Serving runtimes**.
The **Serving runtimes** page opens and shows the model-serving runtimes that are already installed and enabled in your OpenShift Data Science deployment. By default, the OpenVINO Model Server runtime is pre-installed and enabled in OpenShift Data Science.
2. To add a new, custom runtime, click **Add serving runtime**.
The **Add serving runtime** page opens.
3. To start adding a new runtime, perform one of the following sets of actions:
 - **To upload a YAML file**
 - a. Click **Upload files**.
A file browser opens.
 - b. In the file browser, select a YAML file on your computer. This file might be the one of the example runtimes that you downloaded from the [kserve/modelmesh-serving](#) repository.
The embedded YAML editor opens and shows the contents of the file that you uploaded.
 - **To enter YAML code directly in the editor**
 - a. Click **Start from scratch**
The embedded YAML editor opens with no content.

- b. Enter or paste YAML code directly in the embedded editor. The YAML that you paste might be copied from one of the example runtimes in the [kserve/modelmesh-serving](#) repository.
4. Optional: If you are adding one of the example runtimes in the [kserve/modelmesh-serving](#) repository, perform the following modifications:
 - a. In the YAML editor, locate the **kind** field for your runtime. Update the value of this field to **ServingRuntime**.
 - b. In the YAML editor, locate the **containers.image** field for your runtime. Based on the runtime that you are adding, update the field to one of the following:

Nvidia Triton Inference Server

image: nvcr.io/nvidia/tritonserver:21.06.1-py3

Seldon Python MLServer

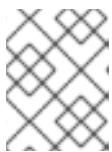
image: seldonio/mlserver:0.5.2

TorchServe

image: pytorch/torchserve:0.6.0-cpu

5. In the **metadata.name** field, ensure that the value of the runtime you are adding is unique (that is, the value isn't the same as for a runtime you have already added).
6. Optional: To configure a custom display name for the runtime that you are adding, add a **metadata.annotations.openshift.io/display-name** field and specify a value, as shown in the following example:

```
apiVersion: serving.kserve.io/v1alpha1
kind: ServingRuntime
metadata:
  name: mlserver-0.x
  annotations:
    openshift.io/display-name: MLServer
```



NOTE

If you do not configure a custom display name for your runtime, OpenShift Data Science shows the value of the **metadata.name** field.

7. Click **Add**.
The **Serving runtimes** page opens and shows the updated list of runtimes that are installed. Observe that the runtime you added is automatically enabled.
8. Optional: To edit your custom runtime, click the action menu (**:**) and select **Edit**.



NOTE

You cannot directly edit the OpenVINO Model Server runtime that is included in OpenShift Data Science by default. However, you can *clone* this runtime and edit the cloned version. You can then add the edited clone as a new, custom runtime. To do this, click the action menu beside the OpenVINO Model Server and select **Clone**.

Verification

- The model-serving runtime you added is shown in an enabled state on the **Serving runtimes** page.

Additional resources

- To learn how to configure a model server that uses a custom model-serving runtime that you have added, see [Adding a model server to your data science project](#) .


5.1.3. Updating a model server

You can update your data science project's model server by changing details, such as the number of deployed replicas, the server size, the token authorization, and how to access the project. If you require extra power for use with large datasets, you can update you can assign accelerators to your model server to optimize performance.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you use specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have created a data science project that has a model server assigned.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**. The **Data science projects** page opens.
2. Click the name of the project whose model server details you want to update. A project details page opens.
3. In the **Models and model servers** section, locate the model server you want to update. Click the action menu () and select **Edit model server**. The **Configure model server** dialog opens.
4. Update the model server properties, as follows:



NOTE

You cannot change the **Serving runtime** selection for a model server that is already configured. This protects against changing to a runtime that does not support already-deployed models.

- a. In the **Model server name** field, enter a new, unique name for the model server.
- b. In the **Number of model replicas to deploy** field, specify a value.
- c. From the **Model server size** list, select one of the following server sizes:
 - Small
 - Medium

- Large
 - Custom
- d. Optional: If you selected **Custom** in the preceding step, configure the following settings in the **Model server size** section to customize your model server:
- i. In the **CPUs requested** field, specify a number of CPUs to use with your model server. Use the list beside this field to specify the value in cores or millicores.
 - ii. In the **CPU limit** field, specify the maximum number of CPUs to use with your model server. Use the list beside this field to specify the value in cores or millicores.
 - iii. In the **Memory requested** field, specify the requested memory for the model server in gibibytes (Gi).
 - iv. In the **Memory limit** field, specify the maximum memory limit for the model server in gibibytes (Gi).



IMPORTANT

OpenShift Data Science includes two versions of the OpenVINO Model Server (OVMS) runtime by default; a version that supports GPUs and one that does not. To use GPUs, from the **Serving runtime** list, you must select the version whose display name includes **Supports GPUs**.

If you use a *custom* model-serving runtime with your model server, you must ensure that your custom runtime supports GPUs and is appropriately configured to use them.

- e. Optional: From the **Accelerator** list, select an accelerator.
 - f. If you selected an accelerator in the preceding step, specify the number of accelerators to use.
 - g. Optional: In the **Model route** section, select the **Make deployed models available through an external route** check box to make your deployed models available to external clients.
 - h. Optional: In the **Token authorization** section, select the **Require token authentication** check box to require token authentication for your model server. To finish configuring token authentication, perform the following actions:
 - i. In the **Service account name** field, enter a service account name for which the token will be generated. The generated token is created and displayed in the **Token secret** field when the model server is configured.
 - ii. To add an additional service account, click **Add a service account** and enter another service account name.
5. Click **Configure**.

Verification

- The model server that you updated appears in the **Models and model servers** section on the project details page.


5.1.4. Duplicating a model server

To avoid re-creating a model server runtime with similar values, you can duplicate an existing model server runtime and then edit some of its settings.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admins**) in OpenShift.
- You have created a data science project that has a model server assigned.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**. The **Data science projects** page opens.
2. Click the name of the project whose model server details you want to duplicate. A project details page opens.
3. In the **Models and model servers** section, locate the model server that you want to copy. Click the action menu () and then select **Duplicate**. The **Configure model server** dialog opens.
4. Update the model server properties as needed.



IMPORTANT

OpenShift Data Science includes two versions of the OpenVINO Model Server (OVMS) runtime by default; a version that supports GPUs and one that does not. To use GPUs, from the **Serving runtime** list, you must select the version whose display name includes **Supports GPUs**.

If you are using a *custom* model-serving runtime with your model server, you must ensure that your custom runtime supports GPUs and is appropriately configured to use them.

5. Click **Configure**.

Verification

- The model server that you created is displayed in the **Models and model servers** section on the project details page.

5.1.5. Deleting a model server

When you no longer need a model server to host models, you can remove it from your data science project.




NOTE

When you remove a model server, you also remove the models that are hosted on that model server. As a result, the models are no longer available to applications.

Prerequisites

- You have created a data science project and an associated model server.
- You have notified the users of the applications that access the models that the models will no longer be available.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users** or **rhods-admin**) in OpenShift.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project from which you want to delete the model server.
A project details page opens.
3. Click the action menu () beside the project whose model server you want to delete in the **Models and model servers** section and then click **Delete model server**.
The **Delete model server** dialog opens.
4. Enter the name of the model server in the text field to confirm that you intend to delete it.
5. Click **Delete model server**.

Verification

- The model server that you deleted is no longer displayed in the **Models and model servers** section on the project details page.

5.2. WORKING WITH DEPLOYED MODELS

5.2.1. Deploying a model in OpenShift Data Science

You can deploy trained models on OpenShift Data Science to enable you to test and implement them into intelligent applications. Deploying a model makes it available as a service that you can access using an API. This enables you to return predictions based on data inputs.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users**) in OpenShift.
- You have created a data science project that contains an associated model server.
- You know the folder path for the data connection that you want the model to access.

Procedure

1. From the OpenShift Data Science dashboard, click **Data Science Projects**.
The **Data science projects** page opens.
2. Click the name of the project containing the model that you want to deploy.
A project details opens.
3. In the **Models and model servers** section, next to the name of your model server, click **Deploy model**.
The **Deploy model** dialog opens.
4. Configure properties for deploying your model as follows:
 - a. In the **Model Name** field, enter a unique name for the model that you are deploying.
 - b. From the **Model framework** list, select a framework for your model.



NOTE

The **Model framework** list shows only the frameworks that are supported by the model-serving runtime that you specified when you configured your model server.

- c. To specify the location of your model, perform one of the following sets of actions:
 - **To use an existing data connection**
 - i. Select **Existing data connection**
 - ii. From the **Name** list, select a data connection that you previously defined.
 - iii. In the **Folder path** field, enter the folder path that contains the model in your specified data source.
 - **To use a new data connection**
 - i. To define a new data connection that your model can access, select **New data connection**.
 - ii. In the **Name** field, enter a unique name for the data connection.
 - iii. In the **Access key** field, enter your access key ID for Amazon Web Services (AWS).
 - iv. In the **Secret key** field, enter your secret access key for the AWS account you specified.
 - v. In the **Endpoint** field, enter the endpoint of your AWS S3 storage.
 - vi. In the **Region** field, enter the default region of your AWS account.
 - vii. In the **Bucket** field, enter the name of the AWS S3 bucket.
 - viii. In the **Folder path** field, enter the folder path in your AWS S3 bucket that contains your data file.
- d. Click **Deploy**.

Verification

- The model you deployed is displayed on the **Model Serving** page of the dashboard.


5.2.2. Viewing a deployed model

To analyse the results of your work, you can view a list of deployed models on Red Hat OpenShift Data Science. You can also view the current statuses of deployed models and their endpoints.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users**) in OpenShift.
- There are active and deployed data science models available on the **Model Serving** page.

Procedure

1. From the OpenShift Data Science dashboard, click **Model Serving**.
The **Model Serving** page opens.
2. Review the list of deployed models.
Inference endpoints are displayed in the **Inference endpoint** column in the **Deployed models** table.
3. Optional: Click the **Copy** button () on the relevant row to copy the model's inference endpoint to the clipboard.

Verification

- A list of previously deployed data science models is displayed on the **Model Serving** page.

5.2.3. Updating the deployment properties of a deployed model


You can update the deployment properties of a model that has been deployed previously. This allows you to change the model's data connection and name.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users**) in OpenShift.
- You have deployed a model on OpenShift Data Science.

Procedure

1. From the OpenShift Data Science dashboard, click **Model serving**.
The **Model Serving** page opens.

2. Click the action menu () beside the model whose deployment properties you want to update and click **Edit**.
The **Deploy model** dialog opens.
3. Update the deployment properties of the model as follows:
 - a. In the **Model Name** field, enter a new, unique name for the model.
 - b. From the **Model framework** list, select a framework for your model.



NOTE

The **Model framework** list shows only the frameworks that are supported by the model-serving runtime that you specified when you configured your model server.

- c. To update how you have specified the location of your model, perform one of the following sets of actions:
 - **If you previously specified an existing data connection**
 - i. In the **Folder path** field, update the folder path that contains the model in your specified data source.
 - **If you previously specified a new data connection**
 - i. In the **Name** field, update the name of the data connection.
 - ii. In the **Access key** field, update your access key ID for Amazon Web Services (AWS).
 - iii. In the **Secret key** field, update your secret access key for the AWS account you specified.
 - iv. In the **Endpoint** field, update the endpoint of your AWS S3 storage.
 - v. In the **Region** field, update the default region of your AWS account.
 - vi. In the **Bucket** field, update the name of the AWS S3 bucket.
 - vii. In the **Folder path** field, update the folder path in your AWS S3 bucket that contains your data file.
- d. Click **Configure**.

Verification

- The model whose deployment properties you updated is displayed on the **Model Serving** page.


5.2.4. Deleting a deployed model

You can delete models you have previously deployed. This enables you to remove deployed models that are no longer required.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you are using specialized OpenShift Data Science groups, you are part of the user group or admin group (for example, **rhods-users**) in OpenShift.
- You have deployed a model.

Procedure

1. From the OpenShift Data Science dashboard, click **Model serving**.
The **Model Serving** page opens.
2. Click the action menu () beside the deployed model that you want to delete and click **Delete**.
The **Delete deployed model** dialog opens.
3. Enter the name of the deployed model in the text field to confirm that you intend to delete it.
4. Click **Delete deployed model**.

Verification

- The model that you deleted is no longer displayed on the **Model Serving** page.

CHAPTER 6. WORKING WITH ACCELERATORS

Use accelerators, such as NVIDIA GPUs and Habana Gaudi devices, to optimize the performance of your end-to-end data science workflows.

6.1. OVERVIEW OF ACCELERATORS

If you work with large data sets, you can use accelerators to optimize the performance of your data science models in OpenShift Data Science. With accelerators, you can scale your work, reduce latency, and increase productivity. You can use accelerators in OpenShift Data Science to assist your data scientists in the following tasks:

- Natural language processing (NLP)
- Inference
- Training deep neural networks
- Data cleansing and data processing

OpenShift Data Science supports the following accelerators:

- NVIDIA graphics processing units (GPUs)
 - To use compute-heavy workloads in your models, you can enable NVIDIA graphics processing units (GPUs) in OpenShift Data Science.
 - To enable GPUs on OpenShift, you must install the NVIDIA GPU Operator.
- Habana Gaudi devices (HPUs)
 - Habana, an Intel company, provides hardware accelerators intended for deep learning workloads. You can use Habana Gaudi accelerators on OpenShift Data Science with version 1.10.0 of the Habana Gaudi Operator.
 - You can enable Habana Gaudi devices on-premises or with AWS DL1 compute nodes on an AWS instance.

Before you can use an accelerator in OpenShift Data Science, your OpenShift instance must contain an associated accelerator profile. For accelerators that are new to your deployment, you must manually configure an accelerator profile for the accelerator in context. If you use NVIDIA GPUs, an accelerator profile is automatically created after you upgrade to the latest version of OpenShift Data Science.

Additional resources

- [Habana, an Intel Company](#)
- [Amazon EC2 DL1 Instances](#)
- [lspci\(8\) - Linux man page](#)

6.2. WORKING WITH ACCELERATOR PROFILES

The accelerator profiles feature is currently available as a Technology Preview. For further information about the scope of Technology Preview status, and the associated support implications, see [Technology Preview Features Support Scope](#).

To configure accelerators for your data scientists to use in OpenShift Data Science, you must create an associated accelerator profile. An accelerator profile is a custom resource definition (CRD) on OpenShift that has an `AcceleratorProfile` resource, and defines the specification of the accelerator.

For accelerators that are new to your deployment, you must manually configure an accelerator profile for each accelerator. If your deployment contained an accelerator before you upgrade, the associated accelerator profile preserves after the upgrade. You can manage the accelerators that appear to your data scientists by assigning specific accelerator profiles to your custom notebook images.

For more information about accelerator profile attributes, see the following table:

Table 6.1. Accelerator profile attributes

| Attribute | Type | Required | Description |
|--------------------------|---------|----------|--|
| <code>displayName</code> | String | Required | The display name of the accelerator. |
| <code>description</code> | String | Optional | Descriptive text defining the accelerator. |
| <code>identifier</code> | String | Required | A unique identifier defining the accelerator. |
| <code>enabled</code> | Boolean | Required | Determines if the accelerator is visible in OpenShift Data Science. |
| <code>tolerations</code> | Array | Optional | The tolerations that can apply to notebooks and serving runtimes that use the accelerator. For more information about the toleration attributes that OpenShift Data Science supports, see Toleration v1 core . |

Additional resources

- [Toleration v1 core](#)
- [Understanding taints and tolerations](#)
- [Managing resources from custom resource definitions](#)

6.2.1. Creating an accelerator profile

To configure accelerators for your data scientists to use in OpenShift Data Science, you must create an associated accelerator profile.

Prerequisites

- You have logged in to OpenShift Container Platform.
- You have the **cluster-admin** role in OpenShift Container Platform.
- The **AcceleratorProfile** custom resource definition (CRD) exists in your deployment.
- You are aware of the correct namespace to create the accelerator profile in.

Procedure

1. In the OpenShift Container Platform web console, in the **Administrator** perspective, click **Administration** → **CustomResourceDefinitions**.
2. In the search bar, enter **acceleratorprofile** to search by name.
The **CustomResourceDefinitions** page reloads to display the search results.
3. Click the **AcceleratorProfile** custom resource definition (CRD).
A details page for the custom resource definition (CRD) opens.
4. Click the **Instances** tab.
5. Click **Create AcceleratorProfile**.
The **Create AcceleratorProfile** page opens with an embedded YAML editor.
6. Enter or paste your accelerator profile YAML code directly in the embedded editor. This example shows the code for a Habana Gaudi 1 accelerator profile:

```
---
apiVersion: dashboard.opendatahub.io/v1alpha
kind: AcceleratorProfile
metadata:
  name: hpu-profile-first-gen-gaudi
spec:
  displayName: Habana HPU - 1st Gen Gaudi
  description: First Generation Habana Gaudi device
  enabled: true
  identifier: habana.ai/gaudi
  tolerations:
    - effect: NoSchedule
      key: habana.ai/gaudi
      operator: Exists
---
```

7. Click **Create**.

Verification

- The **Accelerator** list appears on the **Start a notebook server** page. After you select an accelerator, the **Number of accelerators** field appears, which you can use to choose the number of accelerators for your notebook server.
- The accelerator profile appears on the **Instances** tab on the details page for the **AcceleratorProfile** custom resource definition (CRD).

Additional resources

- [Toleration v1 core](#)
- [Understanding taints and tolerations](#)
- [Managing resources from custom resource definitions](#)

6.2.2. Updating an accelerator profile

You can update the existing accelerator profiles in your deployment. You might want to change important identifying information, such as the display name, the identifier, or the description.

Prerequisites

- You have logged in to OpenShift Container Platform.
- You have the **cluster-admin** role in OpenShift Container Platform.
- The **AcceleratorProfile** custom resource definition (CRD) exists in your deployment and has an accelerator profile.

Procedure

1. In the OpenShift Container Platform web console, in the **Administrator** perspective, click **Administration** → **CustomResourceDefinitions**.
2. In the search bar, enter **acceleratorprofile** to search by name.
The **CustomResourceDefinitions** page reloads to display the search results.
3. Click the **AcceleratorProfile** custom resource definition (CRD).
A details page for the custom resource definition (CRD) opens.
4. Click the **Instances** tab.
5. Click the accelerator profile that you want to update.
A details page for the accelerator profile opens.
6. Update the YAML code in the embedded editor.
7. Click **Save**.

Verification

- If your accelerator profile has new identifying information, this information appears in the **Accelerator** list on the **Start a notebook server** page.

Additional resources

- [Toleration v1 core](#)
- [Understanding taints and tolerations](#)
- [Managing resources from custom resource definitions](#)

6.2.3. Deleting an accelerator profile


To discard accelerator profiles that you no longer require, you can delete them so that they do not appear on the dashboard.

Prerequisites

- You have logged in to OpenShift Container Platform.
- You have the **cluster-admin** role in OpenShift Container Platform.

- The **AcceleratorProfile** custom resource definition (CRD) exists in your deployment and has an accelerator profile.

Procedure

1. In the OpenShift Container Platform web console, in the **Administrator** perspective, click **Administration** → **CustomResourceDefinitions**.
2. In the search bar, enter **acceleratorprofile** to search by name.
The **CustomResourceDefinitions** page reloads to display the search results.
3. Click the **AcceleratorProfile** custom resource definition (CRD).
A details page for the custom resource definition (CRD) opens.
4. Click the **Instances** tab.
5. Click the action menu () beside the accelerator profile that you want to delete and click **Delete AcceleratorProfile**.
The **Delete accelerator profile** dialog opens.
6. Click **Delete**.

Verification

- The accelerator profile no longer appears in the **Accelerator** list on the **Start a notebook server** page.
- The accelerator profile no longer appears on the **Instances** tab on the details page for the **AcceleratorProfile** custom resource definition (CRD).

Additional resources

- [Toleration v1 core](#)
- [Understanding taints and tolerations](#)
- [Managing resources from custom resource definitions](#)

6.2.4. Configuring a recommended accelerator for notebook images

To help you indicate the most suitable accelerators to your data scientists, you can configure a recommended tag to appear on the dashboard.

Prerequisites

- You have logged in to OpenShift Container Platform.
- You have the **cluster-admin** role in OpenShift Container Platform.
- You have configured an ImageStream that contains your notebook image settings.

Procedure

1. In the OpenShift Container Platform web console, in the **Administrator** perspective, click **Builds** → **ImageStreams**.

2. Click the ImageStream that you want to add a recommended accelerator tag to. A details page opens.
3. Click the **YAML** tab. A page with an embedded YAML editor opens.
4. In the editor, enter or paste the YAML code directly to apply the annotation **opendatahub.io/recommended-accelerators**. The excerpt in this example shows the annotation to set a recommended tag for an NVIDIA GPU accelerator:

```
---
metadata:
  annotations:
    opendatahub.io/recommended-accelerators: ["nvidia.com/gpu"]
---
```

5. Click **Save**.

Verification

- When your data scientists select an accelerator with a specific notebook image, a tag appears next to the corresponding accelerator indicating its compatibility.


6.2.5. Configuring a recommended accelerator for serving runtimes

To help you indicate the most suitable accelerators to your data scientists, you can configure a recommended accelerator tag for your serving runtimes.

Prerequisites

- You have logged in to Red Hat OpenShift Data Science.
- If you use specialized OpenShift Data Science groups, you are part of the admin group (for example, **rhods-admin**) in OpenShift.

Procedure

1. From the OpenShift Data Science dashboard, click **Settings > Serving runtimes**. The **Serving runtimes** page opens and shows the model-serving runtimes that are already installed and enabled in your OpenShift Data Science deployment. By default, the OpenVINO Model Server runtime is pre-installed and enabled in OpenShift Data Science.
2. Edit your custom runtime that you want to add the recommended accelerator tag to, click the action menu () and select **Edit**. A page with an embedded YAML editor opens.



NOTE

You cannot directly edit the OpenVINO Model Server runtime that is included in OpenShift Data Science by default. However, you can *clone* this runtime and edit the cloned version. You can then add the edited clone as a new, custom runtime. To do this, click the action menu beside the OpenVINO Model Server and select **Duplicate**.

3. In the editor, enter the YAML code to apply the annotation **opendatahub.io/recommended-accelerators**. The excerpt in this example shows the annotation to set a recommended tag for an NVIDIA GPU accelerator:

```

metadata:
  annotations:
    opendatahub.io/recommended-accelerators: ["nvidia.com/gpu"]

```

4. Click **Update**.

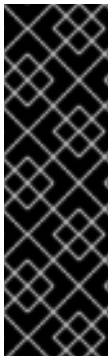
Verification

- When your data scientists select an accelerator with a specific serving runtime, a tag appears next to the corresponding accelerator indicating its compatibility.

6.3. HABANA GAUDI INTEGRATION

To accelerate your high-performance deep learning (DL) models, you can integrate Habana Gaudi devices in OpenShift Data Science. OpenShift Data Science also includes the HabanaAI notebook image. This notebook image is pre-built and ready for your data scientists to use after you install or upgrade OpenShift Data Science.

Before you can successfully enable Habana Gaudi devices in OpenShift Data Science, you must install the necessary dependencies and install the HabanaAI Operator. This allows your data scientists to use Habana libraries and software associated with Habana Gaudi devices from their notebooks. For more information about how to enable your OpenShift environment for Habana Gaudi devices, see [HabanaAI Operator for OpenShift](#).



IMPORTANT

Currently, Habana Gaudi integration is only supported in OpenShift 4.11 and 4.12.

You can use Habana Gaudi accelerators on OpenShift Data Science with version 1.10.0 of the Habana Gaudi Operator. For information about the supported configurations for version 1.10 of the Habana Gaudi Operator, see [Support Matrix v1.10.0](#).

In addition, the version of the HabanaAI Operator that you install must match the version of the HabanaAI notebook image in your deployment.

You can use Habana Gaudi devices in an Amazon EC2 DL1 instance on OpenShift. Therefore, your OpenShift platform must support EC2 DL1 instances. Habana Gaudi accelerators are available to your data scientists when they create a workbench, serve a model, and create a notebook.

To identify the Habana Gaudi devices present in your deployment, use the **lspci** utility. For more information, see [lspci\(8\) - Linux man page](#).



IMPORTANT

If the **lspci** utility indicates that Habana Gaudi devices are present in your deployment, it does not necessarily mean that the devices are ready to use.

Before you can use your Habana Gaudi devices, you must enable them in your OpenShift environment and configure an accelerator profile for each device. For more information about how to enable your OpenShift environment for Habana Gaudi devices, see [HabanaAI Operator for OpenShift](#).

Additional resources

- [HabanaAI Operator for OpenShift](#)
- [lspci\(8\) - Linux man page](#)
- [Amazon EC2 DL1 Instances](#)
- [Support Matrix v1.10.0](#)
- [What version of the Kubernetes API is included with each OpenShift 4.x release?](#)

6.3.1. Enabling Habana Gaudi devices

Before you can use Habana Gaudi devices in OpenShift Data Science, you must install the necessary dependencies and deploy the HabanaAI Operator in your deployment.

Prerequisites

- You have logged in to OpenShift Container Platform.
- You have the **cluster-admin** role in OpenShift Container Platform.

Procedure

1. To enable Habana Gaudi devices in OpenShift Data Science, follow the instructions at [HabanaAI Operator for OpenShift](#).
2. Create an accelerator profile for your Habana Gaudi device.
 - a. In the OpenShift Container Platform web console in the **Administrator** perspective, click **Administration** → **CustomResourceDefinitions**.
 - b. In the search bar, enter **acceleratorprofile** to search by name. The **CustomResourceDefinitions** page reloads to display the search results.
 - c. Click the **AcceleratorProfile** custom resource definition (CRD). A details page for the custom resource definition (CRD) opens.
 - d. Click the **Instances** tab.
 - e. Click **Create AcceleratorProfile**. The **Create AcceleratorProfile** page opens with an embedded YAML editor.
 - f. Enter or paste your accelerator profile YAML code directly in the embedded editor. This example shows the code for a Habana Gaudi 1 accelerator profile:


```
---
apiVersion: dashboard.opendatahub.io/v1alpha
kind: AcceleratorProfile
metadata:
  name: hpu-profile-first-gen-gaudi
spec:
  displayName: Habana HPU - 1st Gen Gaudi
  description: First Generation Habana Gaudi device
  enabled: true
  identifier: habana.ai/gaudi
  tolerations:
    - effect: NoSchedule
      key: habana.ai/gaudi
      operator: Exists
---
```

g. Click **Create**.

Verification

- From the **Administrator** perspective, the following Operators appear on the **Operators** → **Installed Operators** page.
 - HabanaAI
 - Node Feature Discovery (NFD)
 - Kernel Module Management (KMM)
- The **Accelerator** list displays the Habana Gaudi accelerator on the **Start a notebook server** page.
- The accelerator profile appears on the **Instances** tab on the details page for the **AcceleratorProfile** custom resource definition (CRD).

Additional resources

- [HabanaAI Operator for OpenShift](#).

CHAPTER 7. WORKING WITH DISTRIBUTED WORKLOADS

To train complex machine-learning models or process data more quickly, data scientists can use the distributed workloads feature to run their jobs on multiple OpenShift worker nodes in parallel. This approach significantly reduces the task completion time, and enables the use of larger datasets and more complex models.



IMPORTANT

The distributed workloads feature is currently available in Red Hat OpenShift Data Science 2.4 as a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

7.1. OVERVIEW OF DISTRIBUTED WORKLOADS

You can use the distributed workloads feature to queue, scale, and manage the resources required to run data science workloads across multiple nodes in an OpenShift cluster simultaneously. Typically, data science workloads include several types of artificial intelligence (AI) workloads, including machine learning (ML) and Python workloads.

Distributed workloads provide the following benefits:

- You can iterate faster and experiment more frequently because of the reduced processing time.
- You can use larger datasets, which can lead to more accurate models.
- You can use complex models that could not be trained on a single node.

The distributed workloads infrastructure includes the following components:

CodeFlare Operator

Manages the queuing of batch jobs

CodeFlare SDK

Defines and controls the remote distributed compute jobs and infrastructure for any Python-based environment

KubeRay

Manages remote Ray clusters on OpenShift for running distributed compute workloads

You can run distributed data science workloads from data science pipelines or from notebooks.

7.2. CONFIGURING DISTRIBUTED WORKLOADS

To configure the distributed workloads feature for your data scientists to use in OpenShift Data Science, you must enable several components.

Prerequisites

- You have logged in to OpenShift Container Platform with the **cluster-admin** role.
- You have sufficient resources. In addition to the base OpenShift Data Science resources, you need 1.1 vCPU and 1.6 GB memory to deploy the distributed workloads infrastructure.
- You have access to a Ray cluster image. For information about how to create a Ray cluster, see the [Ray documentation](#).
- You have removed any previously installed instances of the CodeFlare Operator, as described in the Knowledgebase solution [How to migrate from CodeFlare Operator to Red Hat OpenShift Data Science Cluster](#).
- If you want to use graphics processing units (GPUs), you have enabled GPU support in OpenShift Data Science. See [Enabling GPU support in OpenShift Data Science](#).

Procedure

1. In the OpenShift Container Platform console, click **Operators** → **Installed Operators**.
2. Search for the **Red Hat OpenShift Data Science** Operator, and then click the Operator name to open the Operator details page.
3. Click the **Data Science Cluster** tab.
4. Click the default instance name to open the instance details page.



NOTE

Starting from Red Hat OpenShift Data Science 2.4, the default instance name for new installations is **default-dsc**. The default instance name for earlier installations, **rhods**, is preserved during upgrade.

5. Click the **YAML** tab to show the instance specifications.
6. In the **spec.components** section, ensure that the **managementState** field is set correctly for the required components depending on whether the distributed workload is run from a pipeline or notebook or both, as shown in the following table.

Table 7.1. Components required for distributed workloads

| Component | Pipelines only | Notebooks only | Pipelines and notebooks |
|-----------------------------|----------------|----------------|-------------------------|
| codeflare | Managed | Managed | Managed |
| dashboard | Managed | Managed | Managed |
| datasciencepipelines | Managed | Removed | Managed |
| ray | Managed | Managed | Managed |
| workbenches | Removed | Managed | Managed |

7. Click **Save**. After a short time, the components with a **Managed** state are ready.

Verification

Check the status of the **codeflare-operator-manager** pod, as follows:

1. In the OpenShift Container Platform console, from the **Project** list, select **redhat-ods-applications**.
2. Click **Workloads** → **Deployments**.
3. Search for the **codeflare-operator-manager** deployment, and click the deployment name to open the deployment details page.
4. Click the **Pods** tab. When the status of the **codeflare-operator-manager-_`pod-id`_** pod is **Running**, the pod is ready to use. To see more information about the pod, click the pod name to open the pod details page, and click the **Logs** tab.

7.3. RUNNING DISTRIBUTED DATA SCIENCE WORKLOADS FROM NOTEBOOKS

To run a distributed data science workload from a notebook, you must first provide the link to your Ray cluster image.

Prerequisites

- You have created a data science project that contains a workbench that is running one of the default notebook images, for example, the **Standard Data Science** notebook. See the table in [Notebook images for data scientists](#) for a complete list of default notebook images.
- You have launched your notebook server and logged in to Jupyter.

Procedure

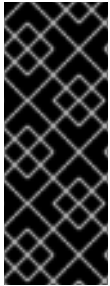
1. To access the demo notebooks, clone the **codeflare-sdk** repository as follows:
 - a. In the JupyterLab interface, click **Git > Clone a Repository**
 - b. In the "Clone a repo" dialog, enter **<https://github.com/project-codeflare/codeflare-sdk.git>** and then click **Clone**. The **codeflare-sdk** repository is listed in the left navigation pane.
2. Run a distributed workload job as shown in the following example:
 - a. In the JupyterLab interface, in the left navigation pane, double-click **codeflare-sdk**.
 - b. Double-click **demo-notebooks**, and then double-click **guided-demos**.
 - c. Update each example demo notebook as follows:
 - Replace the links to the example community image with a link to your Ray cluster image.
 - Set **instascale** to **False**. InstaScale is not deployed in the Technology Preview version of the distributed workloads feature.
 - d. Run the notebooks.

Verification

The notebooks run to completion without errors. In the notebooks, the output from the `cluster.status()` function or `cluster.details()` function indicates that the Ray cluster is **Active**.

CHAPTER 8. SERVING LARGE LANGUAGE MODELS

For serving large language models (LLMs), Red Hat OpenShift Data Science includes a model-serving stack that is based on the Compositional AI Toolkit (Caikit), Text Generation Inference Service (TGIS), and KServe components. The stack helps you to deploy, monitor, scale, and maintain LLMs.



IMPORTANT

The Caikit-TGIS-KServe model-serving stack is a Limited Availability feature. Limited Availability means that you can install and receive support for the feature only with specific approval from Red Hat. Without such approval, the feature is unsupported.

You must use version 2 of the Red Hat OpenShift Data Science Operator to install the Caikit-TGIS-KServe stack.

The Caikit-TGIS-KServe model-serving stack consists of the following components:

- **Caikit**: Wrapper layer that handles the lifecycle of the TGIS process, provides the inference endpoints, and has modules to handle different model types.
- **Caikit-nlp**: Caikit module that handles natural language processing (NLP)-style models.
- **Text Generation Inference Server (TGIS)**: Model-serving backend that loads models and provides an inference engine.
- **KServe**: Kubernetes custom resource definition (CRD) that orchestrates model serving for all types of models. It includes serving runtimes that implement the loading of given types of model servers. KServe handles the lifecycle of the deployment object, storage access, and networking setup.
- **Red Hat OpenShift Serverless**: Cloud-native development model that allows for serverless deployments of data models. OpenShift Serverless is based on the open source [Knative](#) project.
- **Red Hat OpenShift Service Mesh**: Service mesh networking layer that manages traffic flows and enforces access policies. OpenShift Service Mesh is based on the open source [Istio](#) project.

8.1. OVERVIEW OF REQUIRED OPERATORS FOR THE CAIKIT-TGIS-KSERVE STACK

In addition to version 2 of the Red Hat OpenShift Data Science Operator, the Caikit-TGIS-KServe stack requires installation of the following Operators:

- Red Hat OpenShift distributed tracing platform (Jaeger)
- Kiali Operator provided by Red Hat
- Red Hat OpenShift Serverless
- Red Hat OpenShift Service Mesh

The installation instructions referenced in the following section describe how to install each of these required Operators.

8.2. INSTALLING THE CAIKIT-TGIS-KSERVE STACK

To install the Caikit-TGIS-KServe stack, see the [step-by-step instructions](#) in the [caikit-tgis-serving](#) repository.



NOTE

The step-by-step instructions use OpenShift manifest files that are also in the [caikit-tgis-serving](#) repository. These YAML-based manifest files are not part of Red Hat OpenShift Data Science, but are a suitable reference to use for the installation.



NOTE

Step 11 of the [step-by-step instructions](#) shows how to deploy KServe using the Open Data Hub Operator. For equivalent instructions that use version 2 of the Red Hat OpenShift Data Science Operator, see [Installing Red Hat OpenShift Data Science components by using the CLI](#).

8.3. ADDITIONAL RESOURCES

- To learn how to access metrics provided by the Caikit-TGIS-KServe model-serving stack, see [Accessing metrics](#).

CHAPTER 9. TROUBLESHOOTING COMMON PROBLEMS IN JUPYTER FOR ADMINISTRATORS

If your users are experiencing errors in Red Hat OpenShift Data Science relating to Jupyter, their notebooks, or their notebook server, read this section to understand what could be causing the problem, and how to resolve the problem.

If you cannot see the problem here or in the release notes, contact Red Hat Support.

9.1. A USER RECEIVES A 404: PAGE NOT FOUND ERROR WHEN LOGGING IN TO JUPYTER

Problem

If you have configured specialized OpenShift Data Science user groups, the user name might not be added to the default user group for OpenShift Data Science.

Diagnosis

Check whether the user is part of the default user group.

1. Find the names of groups allowed access to Jupyter.
 - a. Log in to OpenShift Container Platform web console.
 - a. Click **User Management** → **Groups**.
 - b. Click the name of your user group, for example, **rhods-users**.
The **Group details** page for that group appears.
 1. Click the **Details** tab for the group and confirm that the **Users** section for the relevant group, contains the users who have permission to access Jupyter.

Resolution

- If the user is not added to any of the groups allowed access to Jupyter, follow [Adding users for OpenShift Data Science](#) to add them.
- If the user is already added to a group that is allowed to access Jupyter, contact Red Hat Support.

9.2. A USER'S NOTEBOOK SERVER DOES NOT START

Problem

The OpenShift Container Platform cluster that hosts the user's notebook server might not have access to enough resources, or the Jupyter pod may have failed.

1. Log in to OpenShift Container Platform web console.
 1. Delete and restart the notebook server pod for this user.
 - a. Click **Workloads** → **Pods** and set the **Project** to **rhods-notebooks**.

- b. Search for the notebook server pod that belongs to this user, for example, **jupyter-nb-`<username>-*`**.

If the notebook server pod exists, an intermittent failure may have occurred in the notebook server pod.

If the notebook server pod for the user does not exist, continue with diagnosis.

2. Check the resources currently available in the OpenShift cluster against the resources required by the selected notebook server image.
If worker nodes with sufficient CPU and RAM are available for scheduling in the cluster, continue with diagnosis.
3. Check the state of the Jupyter pod.

Resolution

- If there was an intermittent failure of the notebook server pod:
 - a. Delete the notebook server pod that belongs to the user.
 - b. Ask the user to start their notebook server again.
- If the notebook server does not have sufficient resources to run the selected notebook server image, either add more resources to the OpenShift cluster, or choose a smaller image size.
- If the Jupyter pod is in a **FAILED** state:
 - a. Retrieve the logs for the **jupyter-nb-*** pod and send them to Red Hat Support for further evaluation.
 - b. Delete the **jupyter-nb-*** pod.
- If none of the previous resolutions apply, contact Red Hat Support.

9.3. THE USER RECEIVES A DATABASE OR DISK IS FULL ERROR OR A NO SPACE LEFT ON DEVICE ERROR WHEN THEY RUN NOTEBOOK CELLS

Problem

The user might have run out of storage space on their notebook server.

Diagnosis

1. Log in to Jupyter and start the notebook server that belongs to the user having problems. If the notebook server does not start, follow these steps to check whether the user has run out of storage space:
 - a. Log in to OpenShift Container Platform web console.
 - a. Click **Workloads** → **Pods** and set the **Project** to **rhods-notebooks**.
 - b. Click the notebook server pod that belongs to this user, for example, **jupyter-nb-`<idp>-<username>-*`**.

- c. Click **Logs**. The user has exceeded their available capacity if you see lines similar to the following:

```
Unexpected error while saving file: XXXX database or disk is full
```

Resolution

- Increase the user's available storage by expanding their persistent volume: [Expanding persistent volumes](#)
- Work with the user to identify files that can be deleted from the **/opt/app-root/src** directory on their notebook server to free up their existing storage space.



NOTE

When you delete files using the JupyterLab file explorer, the files move to the hidden **/opt/app-root/src/.local/share/Trash/files** folder in the persistent storage for the notebook. To free up storage space for notebooks, you must permanently delete these files.

CHAPTER 10. TROUBLESHOOTING COMMON PROBLEMS IN JUPYTER FOR USERS

If you are seeing errors in Red Hat OpenShift Data Science related to Jupyter, your notebooks, or your notebook server, read this section to understand what could be causing the problem.

If you cannot see your problem here or in the release notes, contact Red Hat Support.

10.1. I SEE A 403: FORBIDDEN ERROR WHEN I LOG IN TO JUPYTER

Problem

If your administrator has configured specialized OpenShift Data Science user groups, your user name might not be added to the default user group or the default administrator group for OpenShift Data Science.

Resolution

Contact your administrator so that they can add you to the correct group/s.

10.2. MY NOTEBOOK SERVER DOES NOT START

The OpenShift Platform Container cluster that hosts your notebook server might not have access to enough resources, or the Jupyter pod may have failed.

Resolution

Check the logs in the **Events** section in OpenShift for error messages associated with the problem. For example:

```
Server requested  
2021-10-28T13:31:29.830991Z [Warning] 0/7 nodes are available: 2 Insufficient memory,  
2 node(s) had taint {node-role.kubernetes.io/infra: }, that the pod didn't tolerate, 3 node(s) had taint  
{node-role.kubernetes.io/master: },  
that the pod didn't tolerate.
```

Contact your administrator with details of any relevant error messages so that they can perform further checks.

10.3. I SEE A DATABASE OR DISK IS FULL ERROR OR A NO SPACE LEFT ON DEVICE ERROR WHEN I RUN MY NOTEBOOK CELLS

Problem

You might have run out of storage space on your notebook server.

Resolution

Contact your administrator so that they can perform further checks.

