RED HAT
OPENSHIFT
DATA SCIENCE
1

GETTING STARTED WITH RED HAT OPENSHIFT DATA SCIENCE

Legal Notice

Abstract

Documentation for the Technology Preview release of Red Hat OpenShift Data Science. Technology Previews are provided with a limited support scope, as detailed on the Red Hat Customer Portal: https://access.redhat.com/support/offerings/techpreview

Making open source more inclusive
How this document works
1. Overview of OpenShift Data Science
I. Managing OpenShift Data Science
  2. Overview of deploying OpenShift Data Science
  3. Deploying OpenShift Data Science on your cluster
    3.1. Configuring an identity provider for OpenShift Dedicated
    3.2. Adding administrative users for OpenShift Dedicated
    3.3. Installing OpenShift Data Science on OpenShift Dedicated
  4. Adding users
    4.1. Identity management options for OpenShift Data Science
    4.2. Adding users for OpenShift Data Science
      4.2.1. Adding existing user groups from an identity provider to OpenShift Data Science
      4.2.2. Adding users for OpenShift Data Science using default user groups
II. Working with OpenShift Data Science
  5. Getting started
    5.1. Logging in to OpenShift Data Science

5.2. Launching JupyterHub and starting a notebook server

6. Creating notebooks
6.1. Creating a new notebook from a notebook image
6.2. Importing an existing notebook file from local storage
6.3. Importing an existing notebook file from a git repository using JupyterLab
6.4. Importing an existing notebook file from a git repository using a notebook cell
6.5. Importing an existing notebook file from a git repository using the command line interface

7. Collaborating on notebooks using Git
7.1. Pushing project changes to a git repository

8. Deploying data models as applications
8.1. Overview of developing and deploying data models
8.2. Creating a Python S2I application for OpenShift Data Science from a GitHub template
8.3. Creating an application for OpenShift Data Science using Cookiecutter
8.4. Creating an OpenShift application from a Git repository
8.5. Updating a deployed application
8.6. Viewing your published models

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

HOW THIS DOCUMENT WORKS

You can read this document as a whole or jump to the part that is appropriate for you.

Part I Managing OpenShift Data Science is for administrators who want to set up OpenShift Data Science on their Red Hat OpenShift Dedicated cluster.

Part II Working with OpenShift Data Science is for data scientists starting to work in the OpenShift Data Science environment.

CHAPTER 1. OVERVIEW OF OPENSHIFT DATA SCIENCE

Using Red Hat OpenShift Data Science, users can integrate data, artificial intelligence and machine learning software to execute end-to-end machine learning workflows. OpenShift Data Science is available as an add-on to Red Hat managed environments such as Red Hat OpenShift Dedicated and Red Hat OpenShift Service on Amazon Web Services (AWS).

For data scientists, OpenShift Data Science includes JupyterHub and a collection of default notebook images optimized with the tools and libraries required for model development, including support for graphics processing units (GPU) and the Tensorflow and Pytorch frameworks. Deploy and host your models, integrate models into external applications, and export models to host them in any hybrid cloud environment.

For administrators, OpenShift Data Science provides a simple way to enable data science workloads in an existing Red Hat OpenShift Dedicated or Red Hat OpenShift on AWS environment. Manage users with your existing OpenShift identity provider, and manage resource usage on notebook servers to ensure data scientists have what they need to create, train, and host models.

PART I. MANAGING OPENSHIFT DATA SCIENCE

Table of Contents

- 2. Overview of deploying OpenShift Data Science
- 3. Deploying OpenShift Data Science on your cluster
  - 3.1. Configuring an identity provider for OpenShift Dedicated
  - 3.2. Adding administrative users for OpenShift Dedicated
  - 3.3. Installing OpenShift Data Science on OpenShift Dedicated
- 4. Adding users
  - 4.1. Identity management options for OpenShift Data Science
  - 4.2. Adding users for OpenShift Data Science
    - 4.2.1. Adding existing user groups from an identity provider to OpenShift Data Science
    - 4.2.2. Adding users for OpenShift Data Science using default user groups

CHAPTER 2. OVERVIEW OF DEPLOYING OPENSHIFT DATA SCIENCE

Read this section to understand how to deploy Red Hat OpenShift Data Science as development and testing environment for data scientists.

Prerequisites

- **A Red Hat customer account**

  Go to cloud.redhat.com and log in or register for a new account.

- **Product subscriptions**

  Subscriptions for the following product and add-on:

  - Red Hat OpenShift Dedicated
  - Red Hat OpenShift Data Science Add-On

  Contact your Red Hat account manager to purchase new subscriptions. If you do not yet have an account manager, complete this form to request one.

- **An OpenShift Dedicated cluster**

  Use an existing cluster or create a new cluster by following the OpenShift Dedicated documentation: Creating your cluster.

Procedure

1. Configure an identity provider for OpenShift Dedicated.
2. Add administrative users for OpenShift Dedicated.
3. Install the OpenShift Data Science Add-on.
4. Configure user and administrator groups to provide user access to OpenShift Data Science.
5. Provide your users with the URL for the OpenShift Data Science instance.

Additional resources

- Deploying OpenShift Data Science on your cluster
- Adding users for OpenShift Data Science

CHAPTER 3. DEPLOYING OPENSHIFT DATA SCIENCE ON YOUR CLUSTER

3.1. CONFIGURING AN IDENTITY PROVIDER FOR OPENSHIFT DEDICATED

Configure an identity provider for your OpenShift Dedicated cluster to manage users and groups.

Prerequisites

- Credentials for OpenShift Cluster Manager (https://cloud.redhat.com/openshift/).
- An existing OpenShift Dedicated cluster.

Procedure

1. Log in to OpenShift Cluster Manager (https://cloud.redhat.com/openshift/).
2. Click **Clusters**. The **Clusters** page opens.
3. Click the name of the cluster to configure.
4. Click the **Access control** tab.
5. Click **Add identity provider**.
    a. Select your provider from the **Identity Provider** list.
    b. Enter a unique **Name** for the identity provider. You cannot change this name later.
    c. Select a **Mapping method** from the drop-down menu.
    d. Enter a **Client ID** and **Client secret**.
    e. Optional: Enter a **Hostname** to connect to either GitHub or GitHub Enterprise.
    f. Select **Use organizations** or **Use teams** to restrict access to a particular GitHub organization or a GitHub team.
    g. Enter the name of the organization or team you would like to restrict access to. Click **Add more** to specify multiple organizations or teams that users can be a member of.
    h. Click **Confirm**.

Verification

- The configured identity providers are visible on the **Access control** tab of the **Cluster details** page.

Additional resources

- Configuring identity providers

3.2. ADDING ADMINISTRATIVE USERS FOR OPENSHIFT DEDICATED

You need administrative access to install and configure OpenShift Data Science for your data scientist users.

Prerequisites

- Credentials for OpenShift Cluster Manager (https://cloud.redhat.com/openshift/).
- An existing OpenShift Dedicated cluster with an identity provider configured.

Procedure

1. Log in to OpenShift Cluster Manager (https://cloud.redhat.com/openshift/).
2. Click **Clusters**. The **Clusters** page opens.
3. Click the name of the cluster to configure.
4. Click the **Access control** tab.
5. Under **Cluster administrative users** click the **Add user** button.

   The **Add cluster user** popover appears.
6. Enter the user name in the **User ID** field.
7. Select an appropriate **Group** for the user.

   Important

   If this user needs to use existing groups in an identity provider to control OpenShift Data Science access, select **`cluster-admin`**.

Check Administering a cluster in the OpenShift Dedicated documentation for more information about these user types.

8. Click **Add user**.

Verification

- The user name and selected group are visible in the list of **Cluster administrative users**.

Additional resources

- Administering a cluster

3.3. INSTALLING OPENSHIFT DATA SCIENCE ON OPENSHIFT DEDICATED

You can install Red Hat OpenShift Data Science as an Add-on to your Red Hat OpenShift Dedicated cluster using Red Hat OpenShift Cluster Manager.

Prerequisites

- Purchase entitlements for OpenShift Data Science.
- Credentials for OpenShift Cluster Manager (https://cloud.redhat.com/openshift/).
- Administrator access to the OpenShift Dedicated cluster.

Procedure

1. Log in to OpenShift Cluster Manager (https://cloud.redhat.com/openshift/).
2. Click **Clusters**.

   The **Clusters** page opens.
3. Click the name of the cluster you want to install OpenShift Data Science on.

   The **Details** page for the cluster opens.
4. Click the **Add-ons** tab and locate the **Red Hat OpenShift Data Science** card.
5. Click **Install**.

   The status on the card changes to **`Installing`**.

Verification

- In the **Add-ons** tab for the cluster, confirm that the OpenShift Data Science card shows a status of **`Installed`** and that the **View in console** button is visible.
- Click **Projects** and confirm that the following project namespaces are visible and listed as **Active**:
  - **`redhat-ods-applications`**
  - **`redhat-ods-monitoring`**
  - **`redhat-ods-operator`**

CHAPTER 4. ADDING USERS

4.1. IDENTITY MANAGEMENT OPTIONS FOR OPENSHIFT DATA SCIENCE

OpenShift Data Science supports the same authentication systems as Red Hat OpenShift Dedicated and Red Hat OpenShift Service on Amazon Web Services (AWS).

Check the appropriate documentation for your cluster for more information.

Additional resources

- Supported identity providers on OpenShift Dedicated
- Supported identity providers for Red Hat OpenShift Service on AWS

4.2. ADDING USERS FOR OPENSHIFT DATA SCIENCE

You can grant users permission to access Red Hat OpenShift Data Science by adding user accounts to the Red Hat OpenShift Data Science user group, administrator group, or both. You can either use the default group name, or specify a group name that already exists in your identity provider.

The **user group** provides the user with access to developer functions in the Red Hat OpenShift Data Science dashboard, and associated services, such as JupyterHub. The default user group name is **`rhods-users`**.

The **administrator group** provides the user with access to developer and administrator functions in the Red Hat

OpenShift Data Science dashboard and associated services, such as JupyterHub. The default administrator group name is **`rhods-admins`**.

To use the default group names, go to Adding users for OpenShift Data Science using default user groups. This method is easy to set up, but you need to manage the user lists manually in the OpenShift Dedicated web console.

To use groups that already exist in your identity provider, go to Adding exiwsting user groups from an identity provider to OpenShift Data Science. With this method you can manage users through your identity provider as you normally would.

4.2.1. Adding existing user groups from an identity provider to OpenShift Data Science

You can grant a user access to Red Hat OpenShift Data Science by adding their user name to the OpenShift Data Science user group, administrator group, or both. Follow the steps in this section to use an existing group from your identity provider that does not use one of the default group names, **`rhods-admins`** or **`rhods-users`**. You can add users to these groups as you normally would with that identity provider.

Prerequisites

- You have configured a supported identity provider for OpenShift Dedicated.
- You are part of the **`cluster-admins`** user group in OpenShift Dedicated.

Procedure

1. In the OpenShift Dedicated web console, change into the **Administrator** perspective.
2. Click **Workloads** → **ConfigMaps**.
3. Click the name of the **rhods-groups-config** ConfigMap.

   The **ConfigMap details** page appears.
4. Click the **YAML** tab.
5. Change the **opendatahub.io/modified** label to **`true`**.

```
labels:
  app: jupyterhub
  opendatahub.io/modified: true
```

6. Replace default values with your group names.

Change the value of **`admin_groups`** to the new name of your admin group and the value of **`allowed_groups`** to the new name of your user group, for example:

```
data:
  admin_groups: it-ops
  allowed_groups: datasci-devs
```

7. Click **Save**.

Verification

- Click the **Details** tab and confirm that the **Labels** field contains **`opendatahub.io/modified=true`**, and the updated group names appear under the **Data** heading.
- The user can access the Red Hat OpenShift Data Science dashboard, and associated services, such as JupyterHub.

4.2.2. Adding users for OpenShift Data Science using default user groups

You can grant a user access to Red Hat OpenShift Data Science by adding their user name to the OpenShift Data Science user group, administrator group, or both. Follow the steps in this section to create administrator and user groups that use the default group names, and manually add users to the groups. This method is easy to set up, but you need to manage the user lists manually in the OpenShift Dedicated web console.

Prerequisites

- You have configured a supported identity provider for OpenShift Dedicated.
- You are part of the **`dedicated-admins`** user group in OpenShift Dedicated.

Procedure

1. In the OpenShift Dedicated web console, click **User Management → Groups**.
2. **Optional:** If not present, create the **`rhods-admins`** group.
    a. Click **Create Group**.
    b. Change the **name** of the group to **`rhods-admins`**.

```
apiVersion: user.openshift.io/v1
kind: Group
metadata:
  name: rhods-admins
users:
  - user1
  - user2
```

   c. Skip to step 6 to add administrative users.

3. **Optional:** If not present, create the **`rhods-users`** group.

   a. Click **Create Group**.

   b. Change the **name** of the group to **`rhods-users`**.

```
apiVersion: user.openshift.io/v1
kind: Group
metadata:
  name: rhods-users
users:
  - user1
  - user2
```

   c. Skip to step 6 to add normal users.

4. Click the name of the group you want to add users to.
   - For administrative users, click **`rhods-admins`**.
   - For normal users, click **`rhods-users`**.

   The **Group details** page for that group appears.

5. Click the **YAML** tab.

6. In the **`users`** section, add the user name of the user that you want to add to the group. For example:

```
users:
 - jdoe
 - emustermann
```

7. Click **Save**.

Verification

- Click the **Details** tab and confirm that the **Labels** field contains **`opendatahub.io/modified=true`**, and the updated group names appear under the **Data** heading.

- The user can access the Red Hat OpenShift Data Science dashboard, and associated services, such as JupyterHub.

PART II. WORKING WITH OPENSHIFT DATA SCIENCE

Table of Contents

5. Getting started
   - 5.1. Logging in to OpenShift Data Science
   - 5.2. Launching JupyterHub and starting a notebook server

6. Creating notebooks
   - 6.1. Creating a new notebook from a notebook image
   - 6.2. Importing an existing notebook file from local storage
   - 6.3. Importing an existing notebook file from a git repository using JupyterLab
   - 6.4. Importing an existing notebook file from a git repository using a notebook cell
   - 6.5. Importing an existing notebook file from a git repository using the command line interface

7. Collaborating on notebooks using Git
   - 7.1. Pushing project changes to a git repository

8. Deploying data models as applications
   - 8.1. Overview of developing and deploying data models
   - 8.2. Creating a Python S2I application for OpenShift Data Science from a GitHub template
   - 8.3. Creating an application for OpenShift Data Science using Cookiecutter
   - 8.4. Creating an OpenShift application from a Git repository
   - 8.5. Updating a deployed application
   - 8.6. Viewing your published models

CHAPTER 5. GETTING STARTED

5.1. LOGGING IN TO OPENSHIFT DATA SCIENCE

Log in to OpenShift Data Science from a browser for easy access to JupyterHub and your data science projects.

Prerequisites

- Your administrator has sent you either your credentials or your identity provider and the OpenShift Data Science instance URL. For example, your identity provider might be GitHub and your instance URL might be **`https://rhods-dashboard.apps.example.abc1.p1.openshiftapps.com/`**.

Procedure

1. Browse to the OpenShift Data Science instance URL and click **Log in with OpenShift**.
2. Select your identity provider.
3. Enter your credentials and click **Log in** (or equivalent for your identity provider).

Verification

- OpenShift Data Science opens on the **Enabled applications** page.

Additional resources

- Launching JupyterHub and starting a notebook server

5.2. LAUNCHING JUPYTERHUB AND STARTING A NOTEBOOK SERVER

Launch JupyterHub and start a notebook server to start working with your notebooks.

Prerequisites

- Log in to OpenShift Data Science.

Procedure

1. Locate the **JupyterHub** card on the **Enabled applications** page.
2. Click **Launch**.
3. Start a notebook server.

   This is not required if you have previously launched JupyterHub.

   a. Select the **Notebook image** to use for your server.

   b. Select the **Container size** for your server.

c. Optional: Select the **Number of GPUs** (Graphics Processing Units) for your server.

d. Optional: Select and specify values for any new **Environment variables**.

   Environment variable names must be unique.

   Example variable names for common environment variables are automatically provided for frequently integrated environments and frameworks, such as Anaconda, Amazon Simple Storage Service (S3), or Source-to-Image (S2I).

   The interface remembers previously entered environment variables.

e. Click **Start server**.

   The **Starting server** progress indicator appears.

Verification

- The JupyterHub home page opens in a new tab.

Additional resources

- Options for notebook server environments

CHAPTER 6. CREATING NOTEBOOKS

6.1. CREATING A NEW NOTEBOOK FROM A NOTEBOOK IMAGE

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

Prerequisites

- You have logged in to the OpenShift Dedicated web console.
- You have credentials for logging in to JupyterHub.
- The notebook image exists in a registry, image stream, and is accessible.

Procedure

1. In the OpenShift Dedicated web console, change into the **Developer** perspective.

2. Click **Topology**.
3. The **Graph view** opens, where you can see the status of the applications in your Red Hat OpenShift Data Science deployment, and access JupyterHub.
4. On the JupyterHub application pod, click **Open URL** (↗).

   The JupyterHub Spawner appears.
5. From the **Jupyter Notebook Image** list, select the notebook image that you want to create a notebook from.
6. From the **Container size** list, select an appropriate container size for your notebook server.

   Note

   To view more information about the container size, hover the cursor over the menu options in the list.
7. Optional: In the **Number of required GPUs** field, enter the number of NVIDIA Graphics processing units (GPUs) to be used by the notebook. By default, no GPUs are used by the notebook server.
8. Optional: In the **Environment Variables** section, click **Add**.
9. Optional: In the **Variable_name** field, enter the name of the environment variable that you want to add, or select it from the list.
10. Optional: In the **Variable_value** field, enter a value for the environment variable.

    Note

    Selecting an environment variable from the list automatically populates the **Variable_value** field.
11. Click **Start**.

    The notebook server starts.

Verification

- Check that the notebook file is visible in the **Files** tab in JupyterHub.

6.2. IMPORTING AN EXISTING NOTEBOOK FILE FROM LOCAL

STORAGE

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

Prerequisites

- Credentials for logging in to JupyterHub.
- 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 JupyterHub 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 is displayed in the **File Browser** in the left sidebar of the JupyterHub interface.
- You can open the notebook file in JupyterHub.

6.3. IMPORTING 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 JupyterHub 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 **Files** tab on the JupyterHub home page, or run the **ls** command in the Terminal to verify that the repository is shown as a directory.

6.4. IMPORTING AN EXISTING NOTEBOOK FILE FROM A GIT REPOSITORY USING A NOTEBOOK CELL

You can use a cell in your notebook to clone a Git repository into your workspace to continue your work or integrate files from an external project.

Prerequisites

- A launched and running JupyterHub 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. Open your notebook in JupyterHub.
3. Add a cell to the notebook.

4. Enter `!` followed by the **`git clone`** command.

   ```
   !git clone git-clone-url
   ```

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

   ```
   !git clone https://github.com/example/myrepo.git
   executed in 20ms, finished 12:44:12 2021-04-07
   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 **Files** tab on the JupyterHub home page, or run the **ls** command in the Terminal to verify that the repository is shown as a directory.

6.5. IMPORTING 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 JupyterHub 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. Open a terminal in JupyterHub.

a. From the JupyterHub home page, click the **Files** tab.

b. Click **New → Terminal**.

3. Enter the **`git clone`** command.

```
git clone git-clone-url
```

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

```
[1234567890@jupyterhub-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 **Files** tab on the JupyterHub home page, or run the **ls** command in the terminal to verify that the repository is shown as a directory.

CHAPTER 7. COLLABORATING ON NOTEBOOKS USING GIT

7.1. 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

- A launched and running JupyterHub server.
- Optional: The notebook server is connected to an S3 storage account.
- A configured Git repository.
- You have installed the git version control extension.

- Permission to use the relevant git repository.
- You have configured your project and it is open in JupyterHub.

Procedure

1. Click **File → Save All** to save your most recent changes.
2. In the Jupyter interface, stage your changes and push them to the git repository.
    a. Click **Git → Simple Staging**.
    b. In the JupyterHub menu, click **Git → Push to Remote** to push your changes to the remote repository.

       The **Git credentials required** dialog opens.
    c. Enter your credentials to access the remote repository.
    d. Click **OK**.

Verification

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

CHAPTER 8. DEPLOYING DATA MODELS AS APPLICATIONS

8.1. OVERVIEW OF DEVELOPING AND DEPLOYING DATA MODELS

Read this section to understand the work required to develop and deploy an application using a predictive model created using Red Hat OpenShift Data Science.

Your organization might split responsibility for this process between several roles, such as a data scientist and an application developer, or this work might be done by a single role. An appropriate role is noted for each step.

Table 8.1. Development tasks by role

Application developer	Data scientist	Task description

Application developer	Data scientist	Task description
✔		Create a Python S2I project in Git using an OpenShift Data Science application template. • Use GitHub templates. • Use the Cookiecutter project generator.
✔		Configure the Git project to let data scientists push to and pull from the repository.
From this point, you can develop the model and the application that uses it simultaneously.		
✔		Create an OpenShift application using the project repository.
✔		Build the OpenShift application to verify your code.
✔		Automate the build process using webhooks.
	✔	Launch JupyterHub and create or import a notebook.
	✔	Import the Git project into JupyterHub
	✔	Develop and test your model using notebooks in JupyterHub.
	✔	Extract your model as a Python function in a separate Python file.
	✔	Update the **`requirements.txt`** file with dependencies your function requires.

Application developer	Data scientist	Task description
	✔	Test the function on your notebook server.
	✔	Add your function to the Git project.
	✔	Push your updates back to the remote Git project.
✔	✔	Test the deployed application endpoint.

8.2. CREATING A PYTHON S2I APPLICATION FOR OPENSHIFT DATA SCIENCE FROM A GITHUB TEMPLATE

You can create a Python S2I application quickly from a GitHub template. Use the template to generate a new repository with the same format, directory structure and files as an existing Red Hat OpenShift Data Science repository.

Prerequisites

- You have a GitHub account.
- You have credentials to access the GitHub repository containing the relevant template that you want to use.

Procedure

1. On GitHub, navigate to the main page of the template repository.
2. Click **Use this template**.
3. Optional: From the **Owner** list, select the account that you want to own the repository.
4. In the **Repository name** field, enter a name for the new repository.
5. Optional: In the **Description** field, enter a description for the new repository.
6. Set the repository’s visibility level.

   a. To ensure that the repository is visible to anyone, leave **Public** selected. By default, the repository’s visibility is set to **Public**.

   b. Click **Private** to restrict who can see and commit to the repository.

7. Optional: Select the **Include all branches** check box to copy the template repository’s branches to your new repository.

8. Click **Create repository from template**.

Verification

- The repository that you created from the template is visible and accessible from your GitHub account.

8.3. CREATING AN APPLICATION FOR OPENSHIFT DATA SCIENCE USING COOKIECUTTER

You can create a Red Hat OpenShift Data Science application quickly using Cookiecutter. Cookiecutter is a Python library that creates a flexible, standardized project structure for your data science work. You can use Cookiecutter to further customize your project’s repository. For example, you can modify the repository’s directory structure to suit your project’s requirements.

Prerequisites

- A launched and running JupyterHub server.
- You have a GitHub account.
- You have credentials to access the GitHub repository containing the template that you want to use.

Procedure

1. In the JupyterHub interface, click **File → New → Terminal**.

2. In the terminal, run the **`pip install`** command to install Cookiecutter.

```
pip install cookiecutter
```

3. Run the **`cookiecutter`** command to create a project from a Cookiecutter repository template.

```
cookiecutter template-repository-url
```

Replace *template-repository-url* with the template repository's URL. For example, https://github.com/cookiecutter-template/template.

4. When prompted, provide the following information:

    a. A name for your project.

    b. A name for your repository.

    c. A name for the project's author.

    d. A description for your project.

    e. Your open source license file type.

    The contents of the Cookiecutter template repository appear in the **File Browser** in the left sidebar.

5. Create a repository in GitHub.

    a. In the upper-right corner of the GitHub home page, click **+** → **New repository**.

    The **Create a new repository** page opens.

    b. In the **Repository template** field, select the template that you want to use.

    c. Optional: Select the **Include all branches** check box to copy the template repository's branches to your new repository.

    d. In the **Owner** field, select the repository owner's user name.

    e. In the **Repository** name field, enter a name for the repository.

    f. Optional: In the **Description** field, enter a description of the repository.

6. Set the repository's visibility level.

    a. To ensure that the repository is visible to anyone, leave **Public** selected. By default, the repository's visibility is set to **Public**.

    b. Click **Private** to choose who can see and commit to the repository.

    c. Click **Create repository**.

7. Clone the repository on your JupyterHub server.

    a. In the JupyterHub interface, click **Git** → **Clone a Repository**.

The **Clone a repo** dialog appears.

b. Enter the URL of the repository that you want to clone.

c. Click **Clone**.

The cloned repository appears in the **File Browser** in the left sidebar.

d. In the **File Browser**, move the files and directories created by Cookiecutter to the repository that you cloned.

8. Push your changes to the remote repository.

a. In the left sidebar, click .

b. If you have untracked changes, in the **Changes** tab, hover the cursor over the **Untracked** section bar and click .

c. If you have files that contain changes, in the **Changes tab**, hover the cursor over the **Changed** section bar and click .

d. In the **Required** field, enter a summary of your changes.

e. In the **Description** field, enter a description of your changes.

f. Click **Commit**.

g. In the JupyterHub interface, click **Git → Push to Remote** to push your changes to the remote repository.

The **Git credentials required** dialog opens.

h. Enter your credentials to access the remote repository.

i. Click **OK**.

Verification

- You can access the remote repository that you created from the template.
- You can see the changes that you pushed in the remote repository.

8.4. CREATING AN OPENSHIFT APPLICATION FROM A GIT REPOSITORY

You can import an existing codebase in a Git repository to create, build, and deploy a Red Hat OpenShift Data Science application on OpenShift Dedicated.

Prerequisites

- You have logged in to the web console.
- You are in the **Developer** perspective.
- You have the appropriate roles and permissions in a project to create applications and other workloads in OpenShift Dedicated.
- You have a configured Git repository.
- You have permissions for importing the Git repository.

Procedure

1. In the **Add** view, click **From Git** to see the **Import from git** form.
2. In the **Git** section, enter the Git repository URL for the codebase you want to use to create an application.
3. Optional: Click **Show Advanced Git Options** to add details such as:
   - **Git Reference** to point to code in a specific branch, tag, or commit to be used to build the application.
   - **Context Dir** to specify the subdirectory for the application source code you want to use to build the application.
   - **Source Secret** to create a **Secret Name** with credentials for pulling your source code from a private repository.
4. In the **Builder** section, after the URL is validated, an appropriate builder image is detected, indicated by a star, and automatically selected.
5. In the **General** section:

   a. In the **Application** field, enter a unique name for the application grouping. Ensure that the application name is unique in a namespace.

   b. The **Name** field to identify the resources created for this application is automatically populated based on the Git repository URL.

      Note

      The resource name must be unique in a namespace.

6. In the **Resources** section, select **Deployment Config**, to create an OpenShift style application.
7. In the **Advanced Options** section, the **Create a route to the application** is selected by default so that you can access your application using a publicly available URL. Clear the check box if you do not want to expose your application on a public route.
8. Optional: You can use the following advanced options to further customize your application:
   - **Routing**: Click the **Routing** link to:
     - Customize the hostname for the route.
     - Specify the path the router watches.
     - Select the target port for the traffic from the drop-down list.
     - Secure your route by selecting the **Secure Route** check box. Select the required TLS termination type and set a policy for insecure traffic from the respective drop-down lists.
   - **Build and Deployment Configuration**: Click the **Build Configuration** and **Deployment Configuration** links to see the respective configuration options. Some of the options are selected by default; you can customize them further by adding the necessary triggers and environment variables.
   - **Scaling**: Click the **Scaling** link to define the number of Pods or instances of the application you want to deploy initially.
   - **Resource Limit**: Click the **Resource Limit** link to set the amount of **CPU** and **Memory** resources a container is guaranteed or allowed to use when running.
   - **Labels**: Click the **Labels** link to add custom labels to your application.
   - Click **Create** to create the application and see its build status in the **Topology** view.

Verification

- You can view your application in the **Topology** view.

Additional resources

- Creating applications using the Developer perspective
- Accessing the web console
- About the Developer perspective in the web console
- Default cluster roles

8.5. UPDATING A DEPLOYED APPLICATION

You can deploy your data models on Red Hat OpenShift Data Science to derive insights from your data model.

Prerequisites

- You have logged in to the OpenShift Dedicated web console.
- Credentials for logging in to JupyterHub.
- A launched and running JupyterHub server.
- A configured Git repository.
- Credentials for accessing a remote Git repository.
- A working data model.
- An OpenShift application created on GitHub.

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 OpenShift Dedicated web console, locate the application launcher ( ).
3. Click and select **Red Hat OpenShift Data Science**.

   The **Enabled applications** page opens.

   Note

   Alternatively, you can access the **Enabled applications** page from the **Developer** perspective in the OpenShift Dedicated web console. Click **Topology**. If there is more than one project on your cluster, select your project from the list. The **Graph view** opens, where you can see the status of the applications in your Red Hat OpenShift Data Science deployment, and access JupyterHub. On the JupyterHub application pod, click **Open URL** ( ).
4. Locate the JupyterHub card on the **Enabled applications** page.

5. Click **Launch**.

6. 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.

7. Enter the HTTPS URL of the repository that contains your notebook.

8. Click **CLONE**.

9. If prompted, enter your username and password for the Git repository.

10. Create your data model in JupyterHub.

11. After you have completed your data model, click **Cell → Run all** to ensure it is working correctly.

12. From the **Running** tab on the JupyterHub home page, click the name of the terminal that you created previously.

13. Save your work to your Git repository.

    a. In the left sidebar, click **Git** ( ).

    b. If you have untracked changes, in the **Changes** tab, hover the cursor over the **Untracked** section bar and click **Track all changes** ( ).

    c. If you have files that contain changes, in the **Changes tab**, hover the cursor over the **Changed** section bar and click **Stage all changes** ( ).

    d. In the **Required** field, enter a summary of your changes.

    e. In the **Description** field, enter a description of your changes.

    f. Click **Commit**.

    g. In the JupyterHub interface, click **Git → Push to Remote** to push your changes to the remote repository.

       The **Git credentials required** dialog opens.

    h. Enter your credentials to access the remote repository.

    i. Click **OK**.

14. Build the model.

    If you have configured a webhook, your model builds automatically when the repository is updated.

Otherwise, build manually:

a. In the OpenShift Dedicated web console, change into the **Developer** perspective.

b. Click **Build**.

c. Click your model's build config.

d. Click **Build**.

Verification

- You can view your deployed model in **Topology** view in the **Developer** perspective in Red Hat OpenShift Dedicated.
- You have tested the deployed model using its application endpoint URL.

Additional resources

- Creating an OpenShift application from a Git repository

8.6. VIEWING YOUR PUBLISHED MODELS

You can view data models that you have deployed with Red Hat OpenShift Data Science to analyze the results of your work.

Prerequisites

- You have logged in to the OpenShift Dedicated web console.
- You have successfully deployed a working data model.

Procedure

1. In the OpenShift Dedicated web console, change into the **Developer** perspective.
2. Click **Topology**.

   The **Graph view** opens where you can see the status of the applications in your Red Hat OpenShift Data Science deployment, and access JupyterHub.
3. On the JupyterHub application pod, click **Open URL** (↗).

Verification

- The most recent build of your model opens in a new browser tab

LEGAL NOTICE