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 8. WORKING WITH NOTEBOOKS

8.1. 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 JupyterHub 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 using 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

8.2. 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.

> **Note**
>
> You can also install packages directly, but Red Hat recommends using a **`requirements.txt`** file so that it is easier to deploy your model later.

Prerequisites

- Log in to JupyterHub 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
```

   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 installs the package on your notebook server, but you still need to 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.

8.3. 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 JupyterHub.

Procedure

1. Click **File** → **Hub Control Panel**.

   The control panel opens in a new tab.

2. Click the **Stop my server** button.

   This button disappears when the server stops.

3. Click **My Server** to restart the server and select new settings.

Verification

- The notebook server launcher opens when the server restarts.

Additional resources

- Launching JupyterHub and starting a notebook server