

## Editors

Date published: 2020-02-28

Date modified: 2021-02-25

# CLOUDERA

# Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

<b>Editors.....</b>	<b>4</b>
Configure a Browser IDE as an Editor.....	5
Test a Browser IDE in a Session Before Installation.....	6
Configure a Browser IDE at the Project Level.....	6
Configure a Browser IDE at the Engine Level.....	7
Configure Jupyter Notebook in a Customized Engine Image.....	10
Configure a SSH Gateway to Use Local IDEs.....	11
Configure and Use a Local IDE.....	11
Configure PyCharm as a Local IDE.....	11

## Editors

In addition to the native Cloudera Data Science Workbench editor, you can configure Cloudera Data Science Workbench to work with third-party, browser-based IDEs such as Jupyter and also certain local IDEs that run on your machine, such as PyCharm.



**Note:** ML Runtimes do not support third-party editors.

In the Cloudera Data Science Workbench documentation, browser-based IDEs such as Jupyter and RStudio are referred to as browser IDEs, whereas IDEs such as PyCharm that run on your local machine outside the browser are referred to as local IDEs.

You can use the browser or local IDE of your choice to edit and run code interactively. When you bring your own editor, you still get many of the benefits of Cloudera Data Science Workbench behind an editor interface you are familiar with:

- Dependency management that lets you share code with confidence
- CDH client configurations
- Automatic Kerberos authentication through Cloudera Data Science Workbench
- Reuse code in other Cloudera Data Science Workbench features such as experiments and jobs
- Collaboration features such as teams
- Compliance with IT rules for where compute, data, and/or code must reside. For example, compute occurs within the Cloudera Data Science Workbench deployment, not the local machine. Browser IDEs run within a Cloudera Data Science Workbench session and follow all the same compliance rules. Local IDEs, on the other hand, can bring data or code to a user's machine. Therefore, Site Administrators can opt to disable local IDEs to balance user productivity with compliance concerns.

Note that you can only edit and run code interactively with the IDEs. Tasks such as creating a project or deploying a model require the Cloudera Data Science Workbench web UI and cannot be completed through an editor.

The configuration for an IDE depends on which type of editor you want to use:

### Workbench editor

The Workbench editor is the built-in editor for Cloudera Data Science Workbench. No additional configuration is required to use it. When you launch a session, select the Workbench editor.

### Third-party, browser-based IDEs

Browser IDEs are editors such as Jupyter or RStudio. When you use a browser IDE, it runs within a session and allows you to edit and run code interactively. Changes that you make in the editor are propagated to the Cloudera Data Science Workbench project. Base Engine Image v8 ships with Jupyter preconfigured as a browser IDE. You can select it when you start a session or add a different browser IDE.

Keep the following in mind when using browser IDEs:

- Engine Version Requirements
  - Browser-based IDEs that are configured using custom engines require Base Engine Image v8 or higher.
  - Browser-based IDEs that are configured directly within individual projects do not require a specific engine image. However, Cloudera recommends you use the latest engine image.
- When you are finished using a browser IDE, you must exit the IDE properly, including saving your work if necessary. Do not just stop the Cloudera Data Science Workbench session. Doing so will cause you to lose your session state.
- Depending on the behavior of the browser IDE, multiple users within a project may overwrite each other's state.

- Browser IDEs do not adhere to the timeout set in `IDLE_MAXIMUM_MINUTES`. Instead, they use the timeout set in `SESSION_MAXIMUM_MINUTES`, which is 7 days by default. Cloudera recommends that users stop their session manually after using a browser-based editor. Running sessions continue to consume resources and may impact other users.
- Logs for browser IDEs are available on the Logs tab of the session window. This includes information that the IDE may generate, such as error messages, in addition to any Cloudera Data Science Workbench logs.

### Local IDE Editors on your machine that can use SSH-based remote editing

These editors, referred to as Local IDEs in the documentation, are editors such as PyCharm that run on your local machine. They connect to the Cloudera Data Science Workbench with an SSH endpoint and allow you to edit and run code interactively. You must manually configure some sort of file sync and ignore list between your local machine and Cloudera Data Science Workbench. You can use functionality within the local IDE, such as PyCharm's sync, or external tools that can sync via the SSH endpoint, such as mutagen.

Keep the following in mind before setting up local IDEs:

- Local IDEs do not require a specific engine image, but Cloudera recommends you use the latest engine image.
- Site Administrators should work with IT to determine the data access policies for your organization. For example, your data policy may not allow users to sync certain files to their machines from Cloudera Data Science Workbench. Verify that users understand the requirements and adhere to them when configuring their file sync behavior.
- Users should ensure that the IDEs they want to use support SSH. For example, VS Code supports "remote development over SSH," and PyCharm supports using a "remote interpreter over SSH."

For more information, see [AWS Account Requirements](#).

## Configure a Browser IDE as an Editor

When you use a browser IDE, changes that you make in the editor are propagated to the Cloudera Data Science Workbench project. For example, if you create a new .py file or modify an existing one with the third-party editor, the changes are propagated to Cloudera Data Science Workbench. When you run the code from the notebook, execution is pushed from the notebook to Cloudera Data Science Workbench.

Base Engine Image v8 (and later) comes preconfigured with Jupyter. Jupyter can be selected in place of the built-in Workbench editor when you launch a session, and no additional configuration is required.



**Note:** If you create a customized engine image by extending the CDSW base image, Jupyter Notebook will still be installed on this customized engine image. However, CDSW will not automatically list Jupyter Notebook in the dropdown list of editors on the Start A New Session page in projects that are configured to use this customized engine image. You must configure the custom engine image to use Jupyter Notebook. For details, see [Configure Jupyter Notebook in a Customized Engine Image](#).

You can configure additional IDEs to be available from the dropdown. You have two configuration options:

- **Project Level:** You can configure an editor at the project level so that any session launched within that project can use the editor configured. Other projects across the deployment will not be able to use any editors configured in such a manner. For steps, see [Configure a Browser IDE at the Project Level](#).
- **Engine Level:** You can create a custom engine configured with the editor so that any project across the deployment that uses this custom engine can also use the editor configured. This might be the only option in case of certain browser IDEs (such as RStudio) that require root permission to install and therefore cannot be directly installed within the project. For steps, see [Configure a Browser IDE at the Engine Level](#).

Cloudera recommends you first test the browser IDE you intend to install in a session before you install it to the project or build a custom engine with it. For steps, see [Test a Browser IDE in a Session Before Installation](#).

## Test a Browser IDE in a Session Before Installation

This process can be used to ensure that a browser IDE works as expected before you install it to a project or to a customized engine image. This process is not meant for browser IDEs that require root permission to install, such as RStudio.

### About this task

These steps are only required if you want to use an editor that does not come pre-installed as part of the default engine image. Perform the following steps to configure an editor for your session:

### Procedure

1. Ensure that your browser accepts pop-up windows and cookies from Cloudera Data Science Workbench web UI.
2. Open the Cloudera Data Science Workbench web UI.
3. Go to your project and launch a session with the kernel of your choice and the Workbench editor. Alternatively, open an existing session.
4. In the interactive command prompt or terminal for the session, install the editor you want to use. See the documentation for your editor for specific instructions.

For example:

#### Jupyter Lab

##### Python 2

The following example command installs Jupyter Lab for Python 2:

```
!pip install jupyterlab
```

##### Python 3

The following example command installs Jupyter Lab for Python 3:

```
!pip3 install jupyterlab
```

5. After the installation completes, enter the command to start the server for the notebook on the port specified in the CDSW\_APP\_PORT environment variable on IP address 127.0.0.1.

For example, the following command starts the server for Jupyter Lab on the port specified in the CDSW\_APP\_PORT environment variable:

```
!/home/cdswh/.local/bin/jupyter-lab --no-browser --ip=127.0.0.1 --port=${CDSW_APP_PORT} --NotebookApp.token= --NotebookApp.allow_remote_access=True --log-level=ERROR
```

6. Click on the grid icon in the top right.  
You should see the editor in the drop-down menu. If you select the editor, it opens in a new browser tab.

## Configure a Browser IDE at the Project Level

Perform the following steps to configure an editor at the project level.

### Before you begin



**Note:** The following steps are only required if you want to use an editor that does not come pre-installed as part of the default engine image that Cloudera Data Science Workbench ships with.

### About this task

Perform the following steps to configure an editor at the project level:

## Procedure

1. (Recommended) Test a Browser IDE in a Session Before Installation.
2. Install the IDE of your choice to the project. For information about how to install additional packages to a project, see [AWS Account Requirements](#).
3. Open the Cloudera Data Science Workbench web UI.
4. Go to the project you want to configure an editor for.
5. Go to **Settings Editors** and click **New Editor**.
6. Complete the fields:
  - **Name:** Provide a name for the editor. This is the name that appears in the dropdown menu for Editors when you start a new session.
  - **Command:** Enter the command to start the server for the editor on the Cloudera Data Science Workbench public port specified in the CDSW\_APP\_PORT environment variable (default 8081).

For example, the following command starts Jupyter Lab on the port specified by the CDSW\_APP\_PORT environment variable:

```
/home/cdsw/.local/bin/jupyter-lab --no-browser --ip=127.0.0.1 --port=${CDSW_APP_PORT} --NotebookApp.token= --NotebookApp.allow_remote_access=True --log-level=ERROR
```

This is the same command you used to start the IDE to test it in a session.

7. Save the changes.  
When a user starts a new session, the editor you added is available in the list of editors. Browsers must be configured to accept cookies and allow pop-up windows from the Cloudera Data Science Workbench web UI.

## Configure a Browser IDE at the Engine Level

You can make a browser IDE available to any project within a Cloudera Data Science Workbench deployment by creating a customized engine image, installing the editor to it, and then whitelisting the custom image for projects as needed. Additionally, browser IDEs that require root permission to install, such as RStudio, can only be used as part of a customized engine image.

### Before you begin



**Note:** The following steps are only required if you want to use an editor that does not come pre-installed as part of the default engine image that Cloudera Data Science Workbench ships with.

### About this task

When a user launches a session, they can select the customized engine with the editors available. The following steps describe how to build a customized engine image for RStudio:

## Procedure

1. Create a Dockerfile for the new custom image. Note that the base engine image uses Ubuntu.

The following sample Dockerfile is for RStudio:

```
FROM docker.repository.cloudera.com/cdsw/engine:10

WORKDIR /tmp

#Delete the Cloudera repository that is inaccessible because of the paywal
l

RUN rm /etc/apt/sources.list.d/*
#The RUN commands that install an editor
```

```
#For example: RUN apt-get install myeditor

RUN apt-get update && \
  apt-get install -y --no-install-recommends \
    libapparmor1 \
    libclang-dev \
    lsb-release \
    psmisc \
    sudo && \
  apt-get clean && \
  apt-get autoremove && \
  rm -rf /var/lib/apt/lists/*

RUN wget --quiet https://download2.rstudio.org/server/bionic/amd64/rstudio-
server-1.2.5033-amd64.deb && \
  dpkg -i rstudio-server-1.2.5033-amd64.deb && \
  rm rstudio-server-1.2.5033-amd64.deb
COPY rserver.conf /etc/rstudio/rserver.conf

COPY rstudio-cdsd /usr/local/bin/rstudio-cdsd
RUN chmod +x /usr/local/bin/rstudio-cdsd
```

## 2. Create rserver.conf:

```
# Must match CDSW_APP_PORT
www-port=8090
server-app-armor-enabled=0
server-daemonize=0
www-address=127.0.0.1
auth-none=1
auth-validate-users=0
```

Make sure that the `www-port` property matches the port set in the `CDSW_APP_PORT` environment variable (default 8090).

## 3. Create rstudio-cdsd:

```
#!/bin/bash
# This saves RStudio's user runtime information to /tmp, which ensures
several
# RStudio sessions can run in the same project simultaneously
mkdir -p /tmp/rstudio/sessions/active
mkdir -p /home/cdsd/.rstudio/sessions
if [ -d /home/cdsd/.rstudio/sessions/active ]; then rm -rf /home/cdsd/.rst
udio/sessions/active; fi
ln -s /tmp/rstudio/sessions/active /home/cdsd/.rstudio/sessions/active
# This ensures RStudio picks up the environment. This may not be necess
ary if
# you are installing RStudio Professional. See
# https://docs.rstudio.com/ide/server-pro/r-sessions.html#customizing-sess
ion-launches.
# SPARK_DIST_CLASSPATH is treated as a special case to workaround a bug in
R
# with very long environment variables.
env | grep -v ^SPARK_DIST_CLASSPATH >> /usr/local/lib/R/etc/Renviron.site
echo "Sys.setenv(\"SPARK_DIST_CLASSPATH\"=\"${SPARK_DIST_CLASSPATH}\")"
>> /usr/local/lib/R/etc/Rprofile.site

# Now start RStudio
/usr/sbin/rstudio-server start
```



#### 4. Build the Dockerfile:

```
docker build -t <image-name>:<tag> . -f Dockerfile
```

If you want to build your image on a Cloudera Data Science Workbench gateway host, you must add the `--network=host` option to the build command:

```
docker build --network=host -t <image-name>:<tag> . -f Dockerfile
```

#### 5. Distribute the image:

- Push the image to a public registry such as DockerHub.

For instructions, refer the Docker documentation: [docker push](#).

- Push the image to your company's Docker registry.

When using this method, make sure to tag your image with the following schema:

```
docker tag <image-name> <company-registry>/<user-name>/<image-name>:<tag>
```

Once the image has been tagged properly, use the following command to push the image:

```
docker push <company-registry>/<user-name>/<image-name>:<tag>
```

- Distribute the image manually:

- a. Save the docker image as a tarball on the host where it was built

```
docker image save -o ./<new_customized_engine>.tar <image-name>
```

- b. Distribute the image to all the Cloudera Data Science Workbench gateway hosts.

```
scp ./<new_customized_engine>.tar root@<cdsw.your_company.com>:/tmp/
```

- c. Load the image on all the Cloudera Data Science Workbench gateway hosts.

```
docker load --input /tmp/./<new_customized_engine>.tar
```

- d. To verify that the image was successfully distributed and loaded, run:

```
docker images
```

#### 6. Whitelist the image in Cloudera Data Science Workbench:

- a) Log in to the Cloudera Data Science Workbench web UI as a site administrator.
- b) Click Admin Engines .
- c) Add `<company-registry>/<user-name>/<image-name>:<tag>` to the list of whitelisted engine images.

#### 7. Whitelist the new engine for a project:

- a) Go to the project Settings page.
- b) Click Engines.
- c) Select the new engine from the dropdown list of available Docker images. This engine will now be used to launch sessions within this project.

8. Configure project(s) to use RStudio. When this is done, you will be able to select RStudio from the dropdown list of editors on the Launch New Session page. There are two ways to do this: for an individual project, or for all projects that use this engine.

Configure RStudio for an individual project

- a) Go to the project Settings Editors .
- b) Click New Editor.
- c) Complete the fields:
  - Name: Provide a name for the editor. For example, RStudio. This is the name that appears in the dropdown menu for Editors when you start a new session.
  - Command: Enter the command to start the server for the editor.

For example, the following command will start RStudio:

```
/usr/local/bin/rstudio-cdsw
```

- d) Click Save.

Configure RStudio for all projects that use this engine

- a) Log in to the Cloudera Data Science Workbench web UI as a site administrator.
- b) Click Admin Engines .
- c) Under Engine Images, click the Edit button for the engine image that you whitelisted here in a previous step.
- d) Click New Editor.
  - Name: Provide a name for the editor. For example, RStudio. This is the name that appears in the dropdown menu for Editors when you start a new session.
  - Command: Enter the command to start the server for the editor.

For example, the following command will start RStudio:

```
/usr/local/bin/rstudio-cdsw
```

- e) Click Save, then click Save again.

### What to do next

For more information about how to create a customized engine image and limitations, see [AWS Account Requirements](#)

## Configure Jupyter Notebook in a Customized Engine Image

CDSW's base Image v8 (and later) come with Jupyter Notebook pre-installed on them. If you create a customized engine image by extending this base image, Jupyter Notebook will still be installed on this customized engine image. However, CDSW will not automatically list Jupyter Notebook in the dropdown list of editors on the Start A New Session page in projects that are configured to use this customized engine image.

### About this task

You must use the following steps to configure the custom engine image to use Jupyter Notebook.

### Procedure

1. Log in to the Cloudera Data Science Workbench web UI as a site administrator.
2. Click Admin Engines.
3. Under Engine Images, click the Edit button for the customized engine image that you want to configure for Jupyter Notebook.

#### 4. Click New Editor.

- Name: Enter Jupyter Notebook. This is the name that appears in the dropdown menu for Editors when you start a new session.
- Command: Enter the command to start Jupyter Notebook.

```
/usr/local/bin/jupyter-notebook --no-browser --ip=127.0.0.1 --port=${CDS_W_APP_PORT} --NotebookApp.token= --NotebookApp.allow_remote_access=True --log-level=ERROR
```

#### 5. Click Save, then click Save again.

## Configure a SSH Gateway to Use Local IDEs

Cloudera Data Science Workbench relies on the SSH functionality of the local IDEs to connect to the SSH endpoint on your local machine created with the `cdswctl` client.

Users establish an SSH endpoint on their machine with the `cdswctl` client. This endpoint acts as the bridge that connects the IDE on your machine and the Cloudera Data Science Workbench deployment.

## Configure and Use a Local IDE

The specifics for how to configure a local IDE to work with Cloudera Data Science Workbench are dependent on the local IDE you want to use.

### About this task

The following steps are a high-level description of the steps a user must complete:

### Procedure

1. Establish an SSH endpoint with the [cdswctl Command Interface Client](#).
2. Configure the local IDE to use Cloudera Data Science Workbench as the remote interpreter.
3. Optionally, sync files with tools (like mutagen, SSHFS, or the functionality built into your IDE) from Cloudera Data Science Workbench to your local machine. Ensure that you adhere to IT policies.
4. Edit the code in the local IDE and run the code interactively on Cloudera Data Science Workbench.
5. Sync the files you edited locally to Cloudera Data Science Workbench.
6. Use the Cloudera Data Science Workbench web UI to perform actions such as deploying a model that uses the code you edited.

### What to do next

You can see an end-to-end example for PyCharm configuration here: [Configure Pycharm as Local IDE](#).

## Configure PyCharm as a Local IDE

### About this task

Cloudera Data Science Workbench supports using local IDEs on your machine that allow remote execution and/or file sync over SSH, such as PyCharm. This topic describes the tasks you need to perform to configure Cloudera Data Science Workbench to act as a remote SSH interpreter for PyCharm. Once finished, you can use PyCharm to edit and sync the changes to Cloudera Data Science Workbench. To perform actions such as deploying a model, use the Cloudera Data Science Workbench web UI.



**Note:** These instructions were written for the Professional Edition of PyCharm Version 2019.1. See the documentation for your version of PyCharm for specific instructions.

Before you begin, ensure that the following prerequisites are met:

- You have an edition of PyCharm that supports SSH, such as the Professional Edition.
- You have an SSH public/private key pair for your local machine that is compatible with PyCharm. If you use OpenSSH to generate the key, include the `-m PEM` option because PyCharm does not support modern (RFC 4716) OpenSSH keys.
- You have Contributor permissions for an existing Cloudera Data Science project. Alternatively, create a new project you have access to.

### Download `cdswctl` and Add an SSH Key

The first step to configure PyCharm as a local IDE is to download `cdswctl` and add an SSH key.

#### Procedure

1. Open the Cloudera Data Science Workbench web UI and go to **Settings Remote Editing** for your user account.
2. Download `cdswctl` client for your operating system.
3. In the terminal, run `cat ~/.ssh/id_rsa`. If you used a different filename above when generating the key, use that filename instead. This command prints the key as a string.
4. Copy the key. It should resemble the following: `ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQCha2J5mW3i3BgtZ25/FOsxywpLVkx1RgmZunI`
5. In SSH public keys for session access, paste the key.

#### What to do next

Cloudera Data Science Workbench uses the SSH public key to authenticate your CLI client session, including the SSH endpoint connection to the Cloudera Data Science Workbench deployment. Any SSH endpoints that are running when you add an SSH public key must also be restarted.

### Initialize an SSH Connection to Cloudera Data Science Workbench

The following task describes how to establish an SSH endpoint for Cloudera Data Science Workbench. Creating an SSH endpoint is the first step to configuring a remote editor for Cloudera Data Science Workbench.

#### Procedure

1. Log in to Cloudera Data Science Workbench with the CLI client. Depending on your deployment, make sure you add `http` or `https` to the URL as shown below:

```
cdswctl login -n <username> -u http(s)://cdsw.your_domain.com -y <api_key>
```

For example, the following command logs the user `sample_user` into the `https://cdsw.your_domain.com` deployment:

```
cdswctl login -n sample_user -u https://cdsw.your_domain.com -y <api_key>
```

2. Create a local SSH endpoint to Cloudera Data Science Workbench. Run the following command:

```
cdswctl ssh-endpoint -p <username>/<project_name> [-c <CPU_cores>] [-m
<memory_in_GB>] [-g <number_of_GPUs>]
```

The command uses the following defaults for optional parameters:

- CPU cores: 1
- Memory: 1 GB
- GPUs: 0

For example, the following command starts a session for the logged-in user `sample_user` under the `customerchurn` project with .5 cores, .75 GB of memory, 0 GPUs, and the Python3 kernel:

```
cdswctl ssh-endpoint -p customerchurn -c 0.5 -m 0.75
```

To create an SSH endpoint in a project owned by another user or a team, for example `finance`, prepend the username to the project and separate them with a forward slash:

```
cdswctl ssh-endpoint -p finance/customerchurn -c 0.5 -m 0.75
```

This command creates session in the project `customerchurn` that belongs to the team `finance`.

Information for the SSH endpoint appears in the output:

```
...
You can SSH to it using
  ssh -p <some_port> cdsw@localhost
...
```

3. Open a new command prompt and run the outputted command from the previous step:

```
ssh -p <some_port> cdsw@localhost
```

For example:

```
ssh -p 9750 cdsw@localhost
```

You will be prompted for the passphrase for the SSH key you entered in the Cloudera Data Science web UI.

Once you are connected to the endpoint, you are logged in as the `cdsw` user and can perform actions as though you are accessing the terminal through the Cloudera Data Science Workbench web UI.

4. Test the connection.

If you run `ls`, the project files associated with the session you created are shown. If you run `whoami`, the command returns the `cdsw` user.

### Add Cloudera Data Science Workbench as an Interpreter for PyCharm

In PyCharm, you can configure an SSH interpreter. Cloudera Data Science Workbench uses this method to connect to PyCharm and act as its interpreter.

#### About this task

Before you begin, ensure that the SSH endpoint for Cloudera Data Science Workbench is running on your local machine. These instructions were written for the Professional Edition of PyCharm Version 2019.1 and are meant as a starting point. If additional information is required, see the documentation for your version of PyCharm for specific instructions.

## Procedure

1. Verify that the SSH endpoint for Cloudera Data Science Workbench is running with `cdswctl`. If the endpoint is not running, start it.
2. Open PyCharm.
3. Create a new project.
4. Expand Project Interpreter and select Existing interpreter.
5. Click on ... and select SSH Interpreter
6. Select New server configuration and complete the fields:
  - Host: localhost
  - Port: `<port_number>`  
This is the port number provided by `cdswctl`.
  - Username: `cdsw`

7. Select Key pair and complete the fields using the RSA private key that corresponds to the public key you added to the Remote Editing tab in the Cloudera Data Science Workbench web UI..

For macOS users, you must add your RSA private key to your keychain. In a terminal window, run the following command:

```
ssh-add -K <path to your private key>/<private_key>
```

8. Complete the wizard. Based on the Python version you want to use, enter one of the following parameters:
  - For Python 2: `/usr/local/bin/python`
  - For Python 3: `/usr/local/bin/python3`

You are returned to the New Project window. Existing interpreter is selected, and you should see the connection to Cloudera Data Science Workbench in the Interpreter field.

9. In the Remote project location field, specify the following directory:

```
/home/cdsw
```

10. Create the project.

## (Optional) Configure the Sync Between Cloudera Data Science Workbench and Pycharm (Optional) Configure the Sync Between Cloudera Data Science Workbench and PyCharm

Configuring what files PyCharm ignores can help you adhere to IT policies.

## About this task

Before you configure syncing behavior between the remote editor and Cloudera Data Science Workbench, ensure that you understand the policies set forth by IT and the Site Administrator. For example, a policy might require that data remains within the Cloudera Data Science Workbench deployment but allow you to download and edit code.

## Procedure

1. In your project, go to Preferences.  
Depending on your operating system, Preferences may be called Settings.
2. Go to Build, Execution, Deployment and select Deployment.
3. On the Connection tab, add the following path to the Root path field:

```
/home/cdsw
```

4. On the Excluded Paths tab, add any paths you want to exclude.

Cloudera recommends excluding the following paths at a minimum:

- /home/cdsw/.local
- /home/cdsw/.cache
- /home/cdsw/.ipython
- /home/cdsw/.ipython
- /home/cdsw/.oracle\_jre\_usage
- /home/cdsw/.pip
- /home/cdsw/.pycharm\_helpers

5. Optionally, add a Deployment path on the Mappings tab if the code for your Cloudera Data Science Workbench project lives in a subdirectory of the root path.
6. Expand Deployment in the left navigation and go to OptionsUpload changed files automatically to the default server and set the behavior to adhere to the policies set forth by IT and the Site Administrator.

Cloudera recommends setting the behavior to Automatic upload because the data remains on the cluster while your changes get uploaded.

7. Sync for the project file(s) to your machine and begin editing.