Cloudera Data Science Workbench

# Editors

**Date published: 2020-02-28**
**Date modified:**

CLOUDERA

**https://docs.cloudera.com/**

# Legal Notice

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.

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

# Editors

In addition to the built-in 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 and VS Code.

**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 and VS Codethat 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.

- 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/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
   ```
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=${C
     DSW_APP_PORT} --NotebookApp.token= --NotebookApp.allow_remote_access=Tru
     e --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 Legacy Engine Level

You can make a browser IDE available to any project within a Cloudera Data Science Workbench deployment by creating a customized legacy 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 legacy 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 legacy engine with the editors available. The following steps describe how to build a customized legacy 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-cdsw /usr/local/bin/rstudio-cdsw
RUN chmod 777 /usr/local/bin/rstudio-cdsw
```

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-cdsw:

```
#!/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/cdsw/.rstudio/sessions
if [ -d /home/cdsw/.rstudio/sessions/active ]; then rm -rf /home/cdsw/.rst
udio/sessions/active; fi
ln -s /tmp/rstudio/sessions/active /home/cdsw/.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. Add th enew legacy engine to the trusted list for a project:

   a) Go to the project Settings page.
   b) Click Engines.
   c) Select the new legacy 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 or VS Code 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.pub. 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 AAAAB3NzaC1yc2EAAAADAQABAAACAQCha2J5 mW3i3BgtZ25/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 for Pycharm

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 <legacy_
   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 <legacy_
   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>] [-r <runtime ID> ]
   ```

   If the project is configured to use ML runtimes, the -r parameter must be specified, otherwise it must be omitted. See *Using ML runtimes with cdswctl* documentation page for more information.

   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 VS Code

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

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  Options Upload 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.

## Configure VS Code 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 VS Code. This topic describes the tasks you need to perform to configure Cloudera Data Science Workbench to act as a remote SSH interpreter for VS Code. Once finished, you can use VS Code 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.

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

- You have an edition of VS Code that supports SSH.
- You have an SSH public/private key pair for your local machine that is compatible with VS Code.
- You have Contributor permissions for an existing Cloudera Data Science Workbench project. Alternatively, create a new project you have access to.

### Download cdswctl and Add an SSH Key

The first step to configure VS Code 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.pub. 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    AAAAB3NzaC1yc2EAAAADAQABAAACAQCha2J5 mW3i3BgtZ25/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 for VS Code

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.

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

   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
   ```

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>] [-r <runtime ID> ]
   ```

   If the project is configured to use ML runtimes, the -r parameter must be specified, otherwise it must be omitted. To retrieve the Runtime ID, use the following command:

   cdswctl runtimes list

   See *Using ML runtimes with cdswctl* documentation page for more information.

   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 7847 cdsw@localhost
   ```

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

   The public key could be rejected when the new ssh key pair is generated with a special name such as id_rsa_systest. If the public key is rejected, you must add the following information to the ~/.ssh/config  file:

   ```
   Host *
             AddKeysToAgent yes
             StrictHostKeyChecking no
             IdentityFile ~/.ssh/id_rsa_cdswctl
   ```

   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.

   Once you are connected, you should see something like this:

   ```
   $ cdswctl ssh-endpoint -p ml-at-scale -m 4 -c 2
   Forwarding local port 7847 to port 2222 on session bhsb7k4eqmonap62 in p
   roject finance/customerchurn.
   You can SSH to the session using

       ssh -p 7847 cdsw@localhost
   ```

5. Add an entry into your SSH config file.

   For example:

   ```
   $ cat ~/.ssh/config
   Host cdsw-public
       HostName localhost
       IdentityFile ~/.ssh/id_rsa
       User cdsw
       Port 7847
   ```

   HostName is always localhost and User is always cdsw. You get the Port number from Step 2.

## Setting up VS Code

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

## Before you begin

Ensure that you have installed the following:

- Remote SSH extension

  Remove Development using SSH
- [Optional] Python extension
- [Optional] R extension

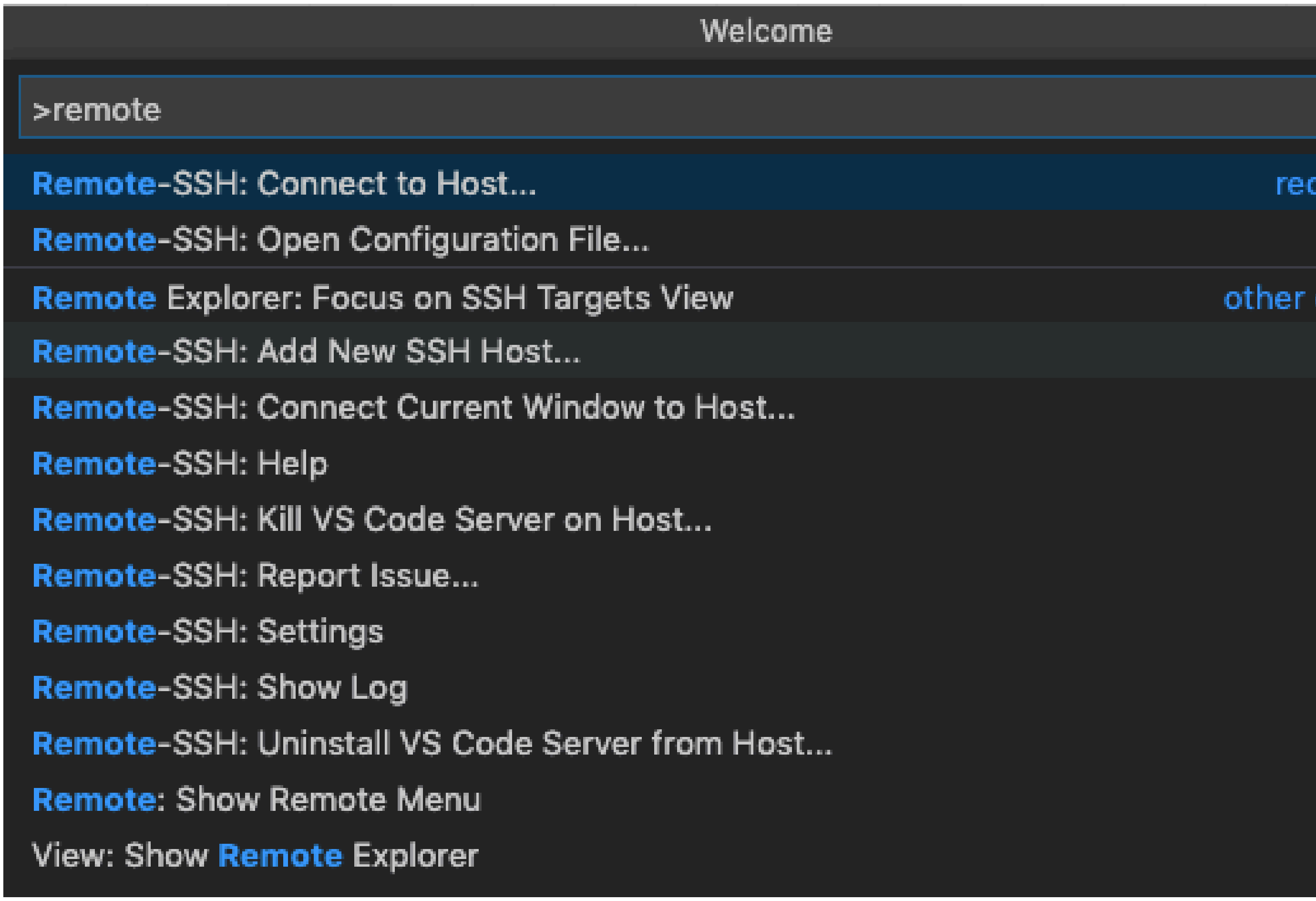
### About this task

Before you begin, ensure that the SSH endpoint for Cloudera Data Science Workbench is running on your local machine. If additional information is required, see the documentation for your version of VS Code for specific instructions.

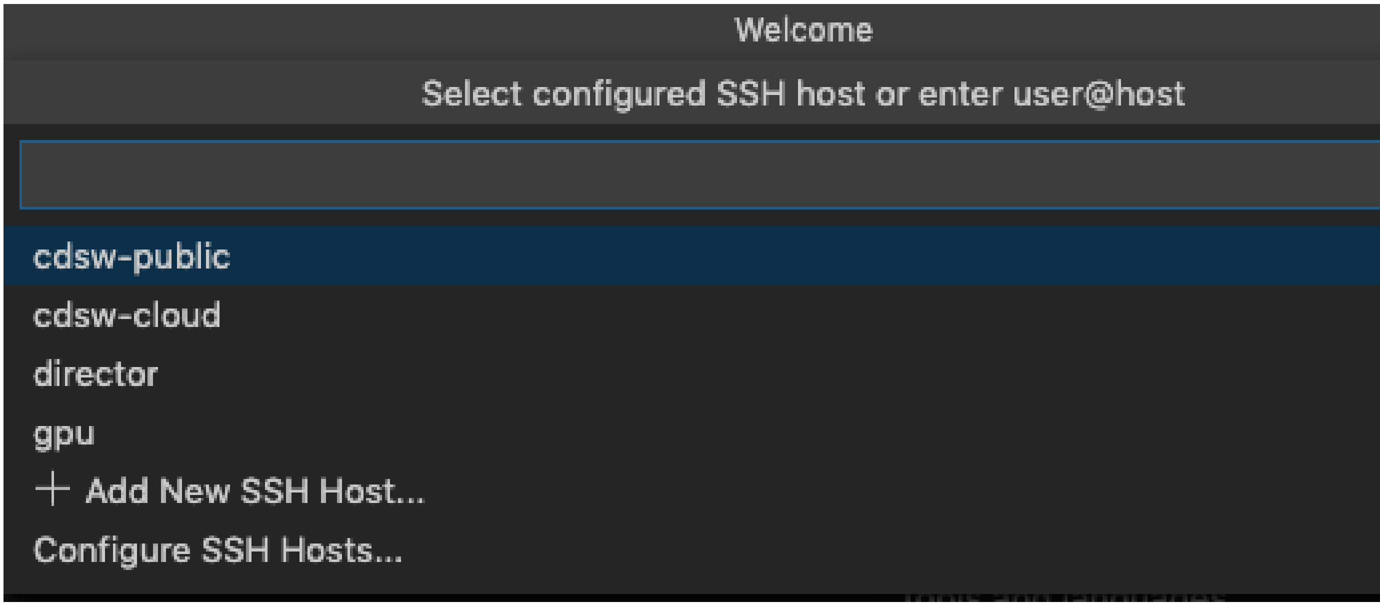
### Procedure

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 VS Code.
3. Open the command pallet and connect to a remote host.


Welcome
>remote
Remote-SSH: Connect to Host...
rec
Remote-SSH: Open Configuration File...
Remote Explorer: Focus on SSH Targets View
other
Remote-SSH: Add New SSH Host...
Remote-SSH: Connect Current Window to Host...
Remote-SSH: Help
Remote-SSH: Kill VS Code Server on Host...
Remote-SSH: Report Issue...
Remote-SSH: Settings
Remote-SSH: Show Log
Remote-SSH: Uninstall VS Code Server from Host...
Remote: Show Remote Menu
View: Show Remote Explorer

4. Connect to the host you added previously.

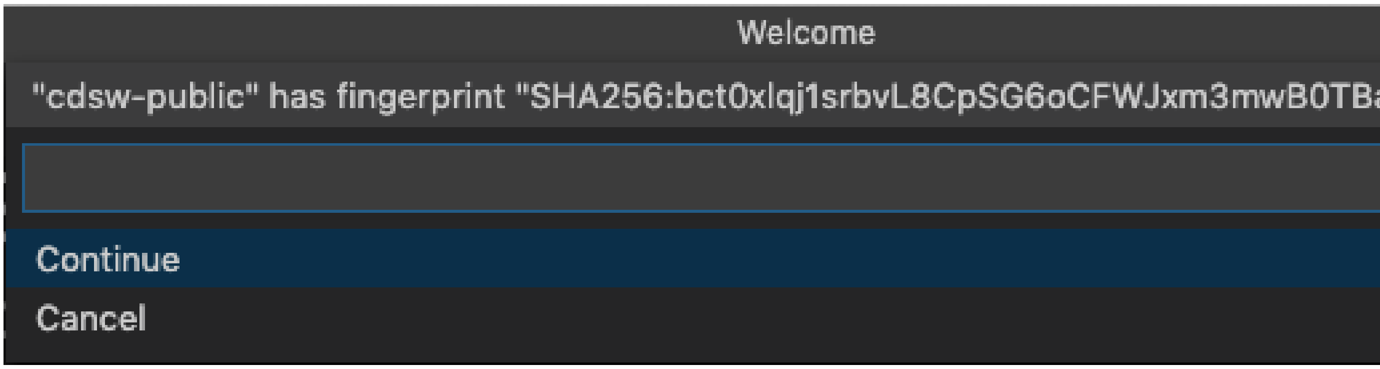


Welcome

Select configured SSH host or enter user@host

5. For the first connection, you must accept the fingerprint.

   You might not see a pop up, so pay attention to VS Code. If it's the first time your are connecting to a new session, or the port number changed, you will need to accept the fingerprint.



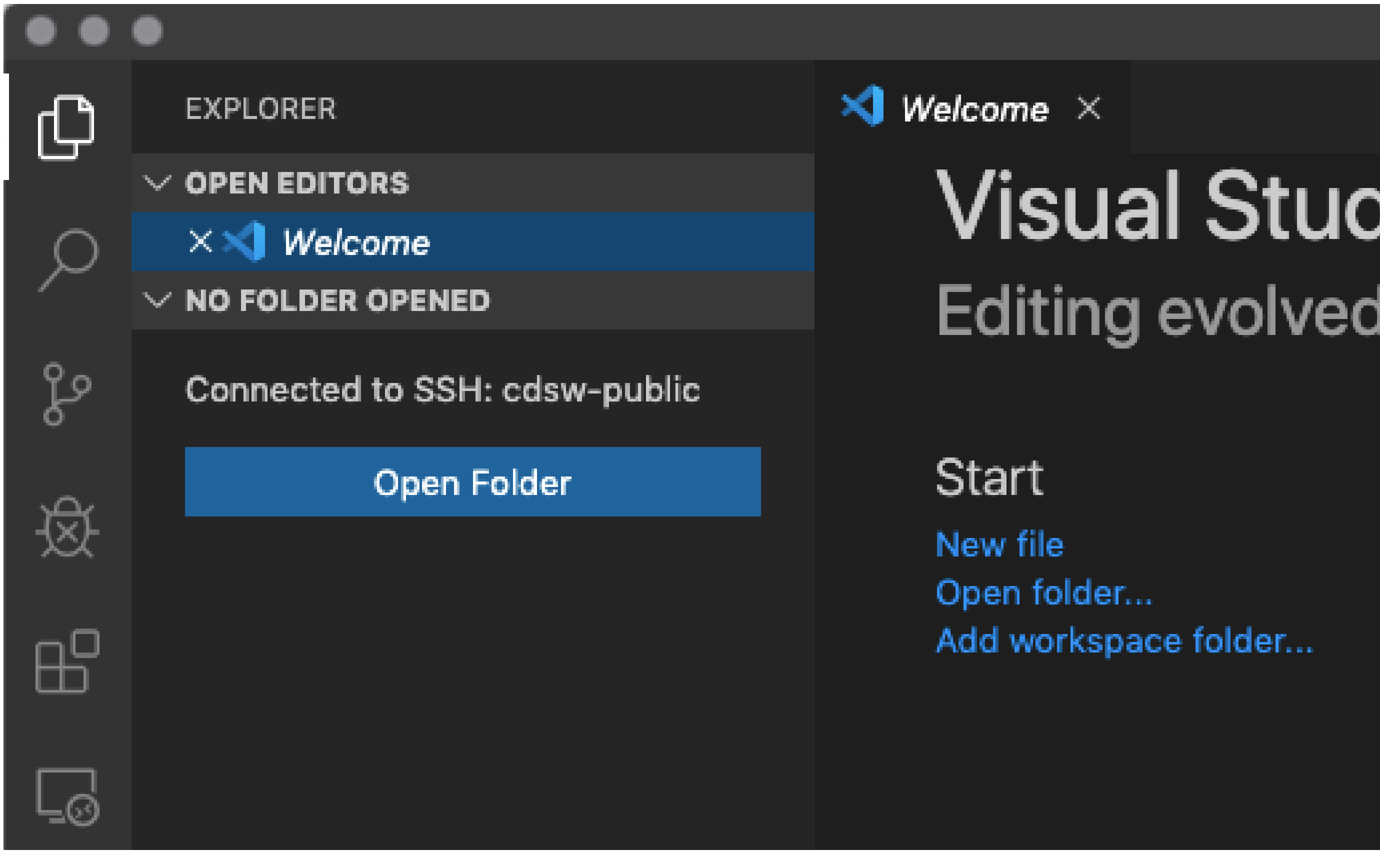
Welcome

"cdsw-public" has fingerprint "SHA256:bct0xlqj1srbvL8CpSG6oCFWJxm3mwB0TB

While VS Code connects and sets up the remote connection, it installs some helper applications on the Cloudera Data Science Workbench server. Sometimes the remote session dies. Click Retry or if it's taking a long time, restart the remote session and it will recover.

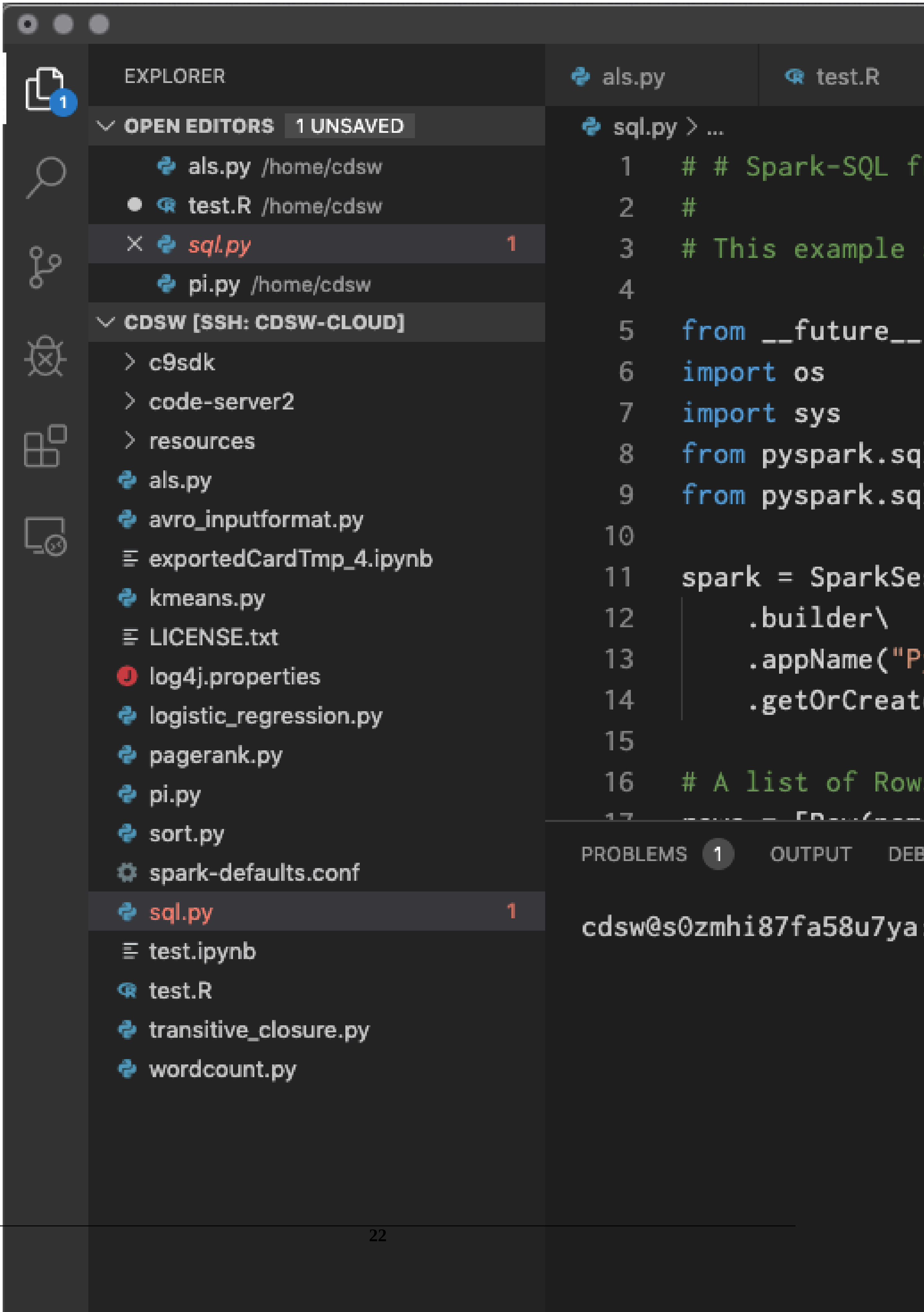
**Note:** If you get stuck in a loop during setup with VS Code reconnecting every 30 secs or so, the issue is with the lock file that VS Code creates during the install. Close VS Code and in CML terminal, delete the /home/cdsw/.vscode-server/ directory and start again.

6. After you are connected, you can open the Explorer and view and edit the files in the /home/cdsw directory.



Connected to SSH: cdsw-public

7. From the Explorer view, you can edit any of the files on your Cloudera Data Science Workbench server.



```
# # Spark-SQL f
#
# This example

from __future__
import os
import sys
from pyspark.sq
from pyspark.sq

spark = SparkSe
    .builder\
    .appName("P
    .getOrCreat

# A list of Row
```

```
cdsw@s0zmhi87fa58u7ya
```

Using the Explorer view, you remotely edit and modify your Cloudera Data Science Workbench files. VS Code also supports Python and R which you offer has some powerful coding tools that you can take advantage of over the remote connection.

## (Optional) Using VS Code with Python

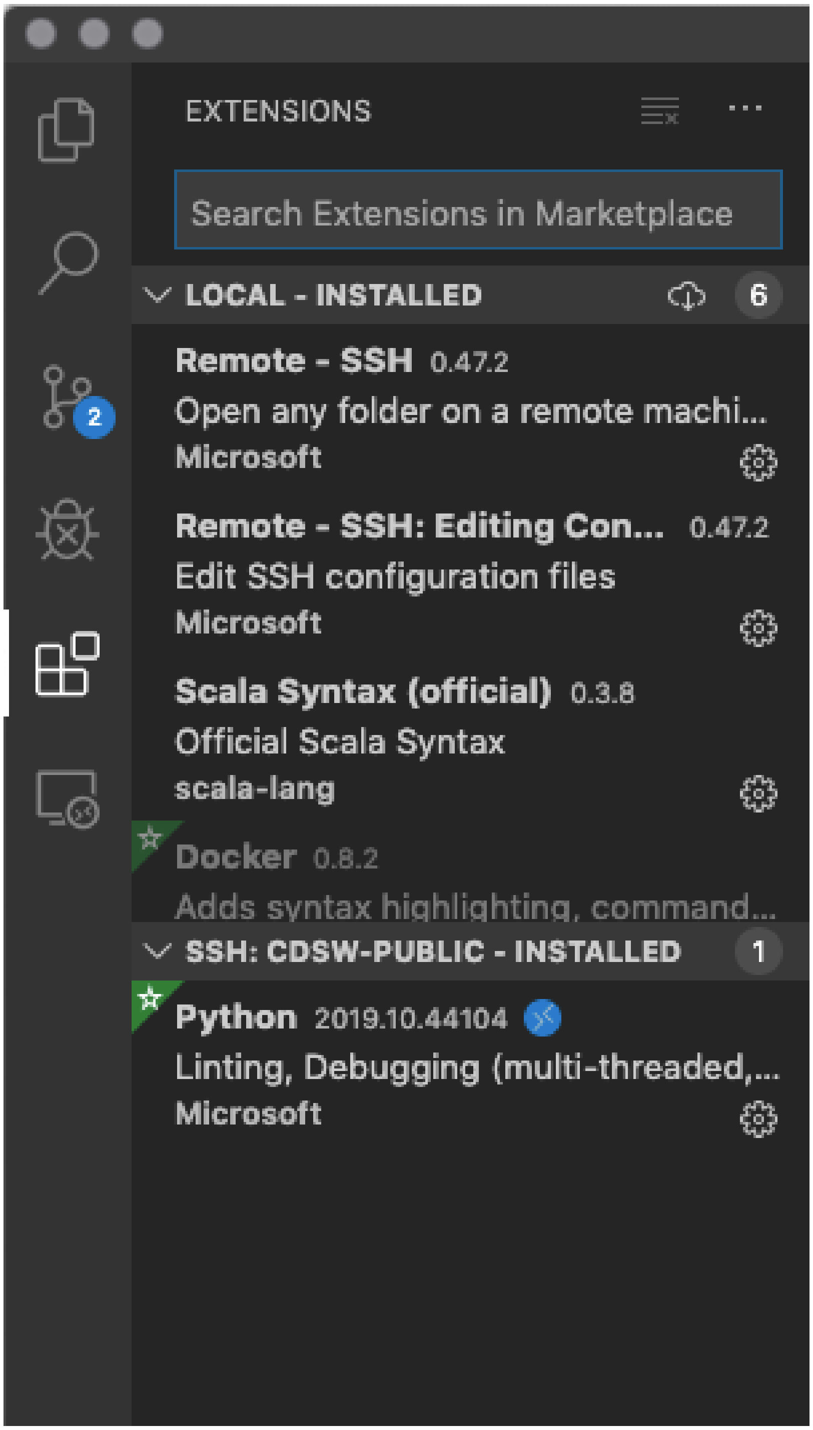
You can use VS Code with Python.

### About this task

To take full advantage of VS Code Python tools, you must install the Python extension into the remote ssh session. You must install the extension the first time you connect a newly configured remote session.

### Procedure

1. Install the Python extension.



EXTENSIONS

Search Extensions in Marketplace

LOCAL - INSTALLED 6

Remote - SSH 0.47.2
Open any folder on a remote machi...
Microsoft

Remote - SSH: Editing Con... 0.47.2
Edit SSH configuration files
Microsoft

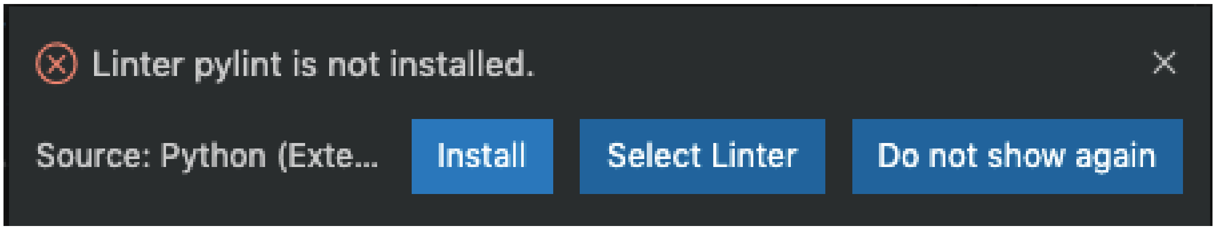
Scala Syntax (official) 0.3.8
Official Scala Syntax
scala-lang

Docker 0.8.2
Adds syntax highlighting, command...

SSH: CDSW-PUBLIC - INSTALLED 1

Python 2019.10.44104
Linting, Debugging (multi-threaded,...
Microsoft

2. With the Extension installed, once you open your first python file, you will be prompted to install pylint Linter.



Linter pylint is not installed.

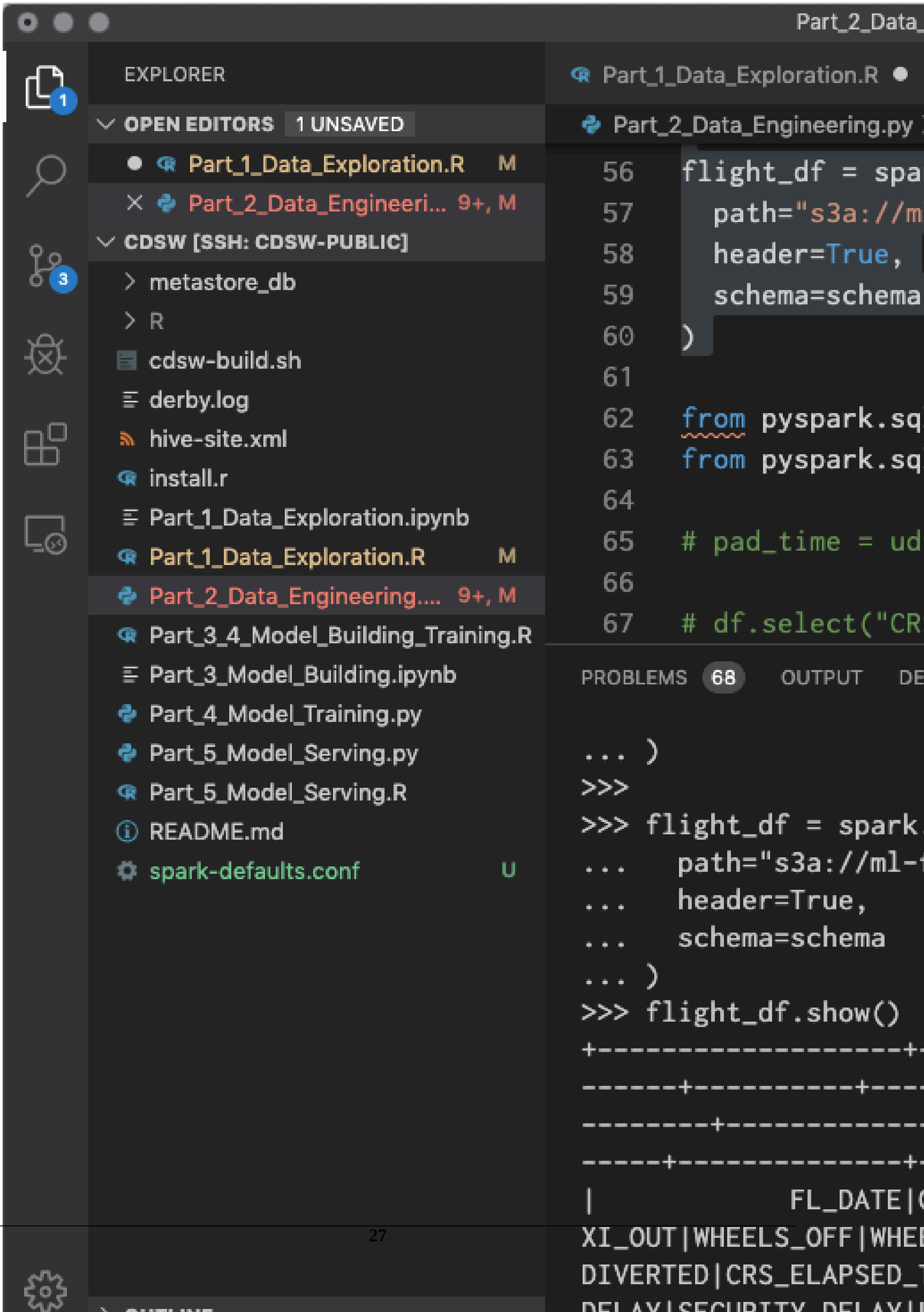
Source: Python (Exte...

3. Click Install.

   VS Code opens a terminal and runs the code needed to install the linter. It's important to note that this is a remote terminal, running on an engine in Cloudera Data Science Workbench. It's the same as if you launched a terminal inside a running workbench.

4. If you want to run arbitrary Python code inside VS Code, open a Python file, select some code, right click, and select Run Selection/Line in Python Terminal.

   You can also just hit Shift-Enter in the code editor window. This will open up a new terminal if there isn’t one and run the selected code. Since this is a remote session, you can run pyspark directly inside VS Code.



Open editors:

- ● Part_1_Data_Exploration.R M
- × Part_2_Data_Engineeri... 9+, M

Files:

- metastore_db
- R
- cdsw-build.sh
- derby.log
- hive-site.xml
- install.r
- Part_1_Data_Exploration.ipynb
- Part_1_Data_Exploration.R M
- Part_2_Data_Engineering.... 9+, M
- Part_3_4_Model_Building_Training.R
- Part_3_Model_Building.ipynb
- Part_4_Model_Training.py
- Part_5_Model_Serving.py
- Part_5_Model_Serving.R
- README.md
- spark-defaults.conf U

Editor tabs: Part_1_Data_Exploration.R ●, Part_2_Data_Engineering.py

```
flight_df = spa
  path="s3a://m
  header=True,
  schema=schema
)

from pyspark.sq
from pyspark.sq

# pad_time = ud

# df.select("CR
```

```
... )
>>>
>>> flight_df = spark
...   path="s3a://ml-
...   header=True,
...   schema=schema
... )
>>> flight_df.show()
+--------------------+-
-------+-----------+---
---------+-------------
-----+-------------+-
|            FL_DATE|C
XI_OUT|WHEELS_OFF|WHE
DIVERTED|CRS_ELAPSED_
DELAY|SECURITY_DELAY|
```

For more complex code requirements, you can also use the Python Debugging feature in VS Code.

## (Optional) Using VS Code with R

You can use VS Code with R.

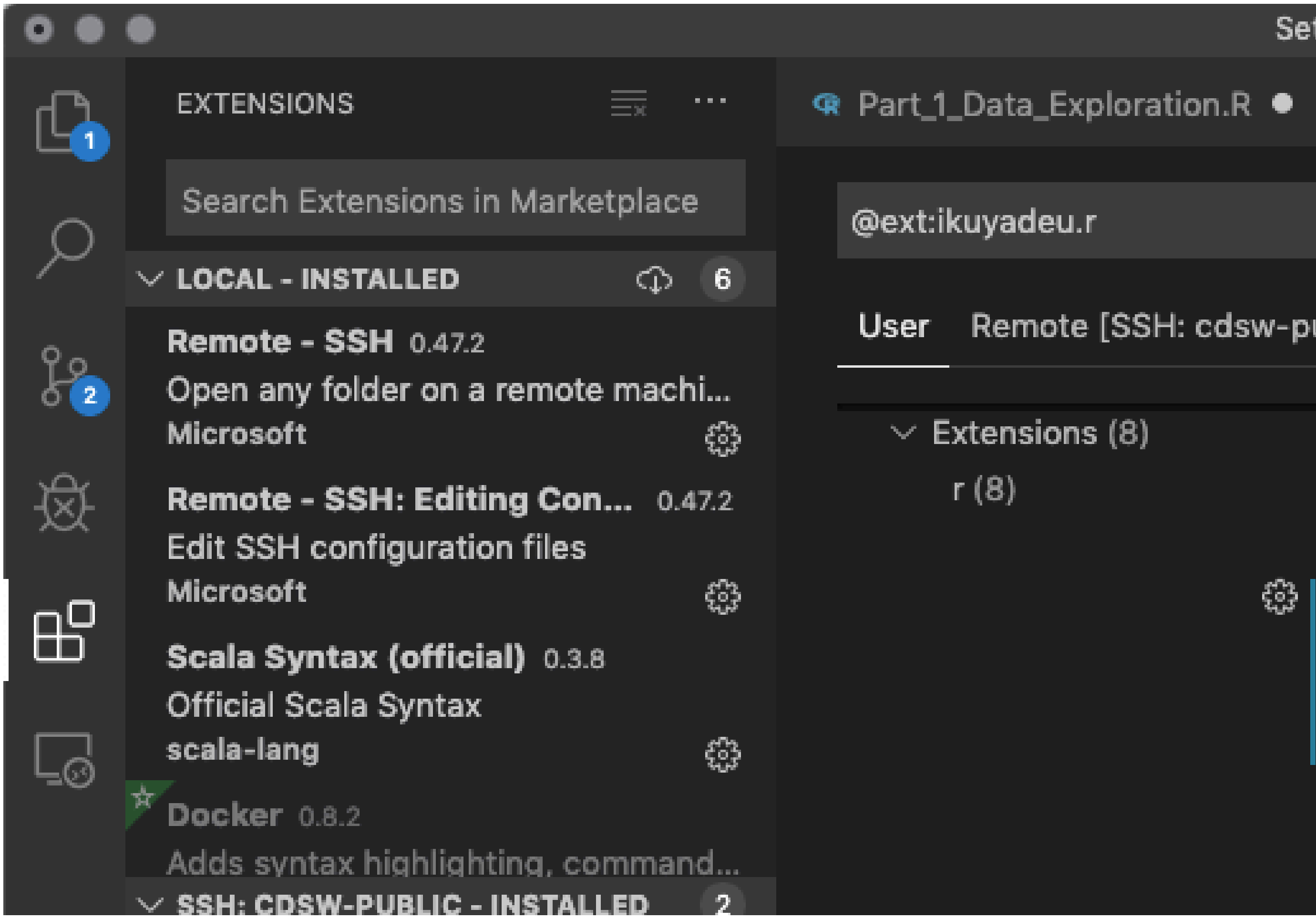
### About this task

The R extension provides similar capabilities as Python. This means you can edit R files with code completion and execute arbitrary code in the terminal. With sparklyr, you can run spark code using R inside VS Code.

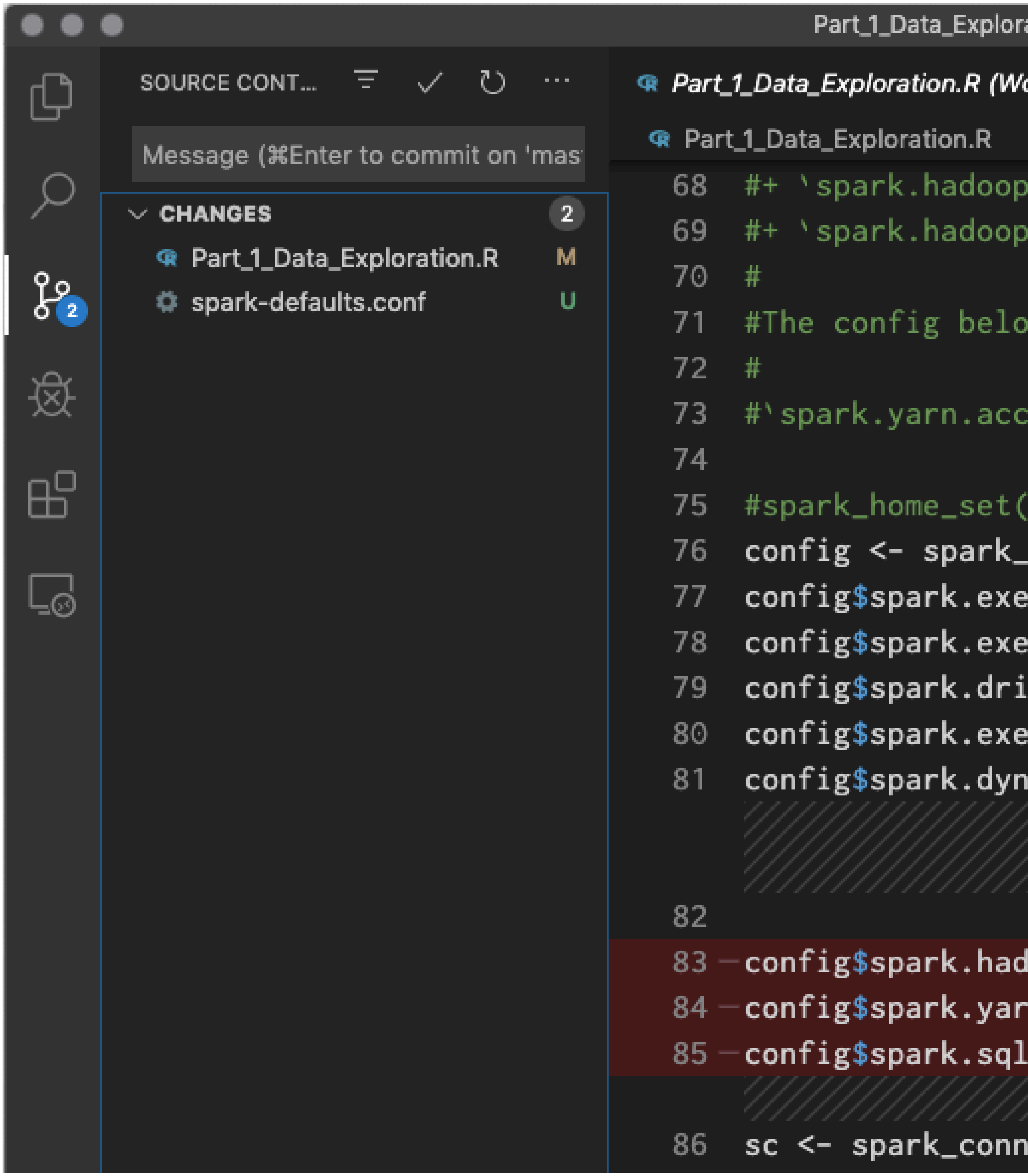
### Procedure

1. Prior to installing the R extension, check where your R binary lives in CDW by running which R and then pasting that into the R > Rterm:   Linux setting in VS Code.

   The R binary is most likely located in /usr/local/bin/R, but its best to check.


Set
EXTENSIONS
Search Extensions in Marketplace
LOCAL - INSTALLED 6
Remote - SSH 0.47.2
Open any folder on a remote machi...
Microsoft
Remote - SSH: Editing Con... 0.47.2
Edit SSH configuration files
Microsoft
Scala Syntax (official) 0.3.8
Official Scala Syntax
scala-lang
Docker 0.8.2
Adds syntax highlighting, command...
SSH: CDSW-PUBLIC - INSTALLED 2
1
2
Part_1_Data_Exploration.R
@ext:ikuyadeu.r
User
Remote [SSH: cdsw-pu
Extensions (8)
r (8)

2. After you install the R extension, you can use sparklyr to run spark code using R inside VS Code.


Part_1_Data_Explora
SOURCE CONT...
Message (⌘Enter to commit on 'mas
CHANGES 2
Part_1_Data_Exploration.R M
spark-defaults.conf U
2
Part_1_Data_Exploration.R (Wo
Part_1_Data_Exploration.R

```
#+ `spark.hadoop
#+ `spark.hadoop
#
#The config belo
#
#`spark.yarn.acc

#spark_home_set(
config <- spark_
config$spark.exe
config$spark.exe
config$spark.dri
config$spark.exe
config$spark.dyn

- config$spark.had
- config$spark.yar
- config$spark.sql

sc <- spark_conn
```

## (Optional) Using VS Code with Jupyter

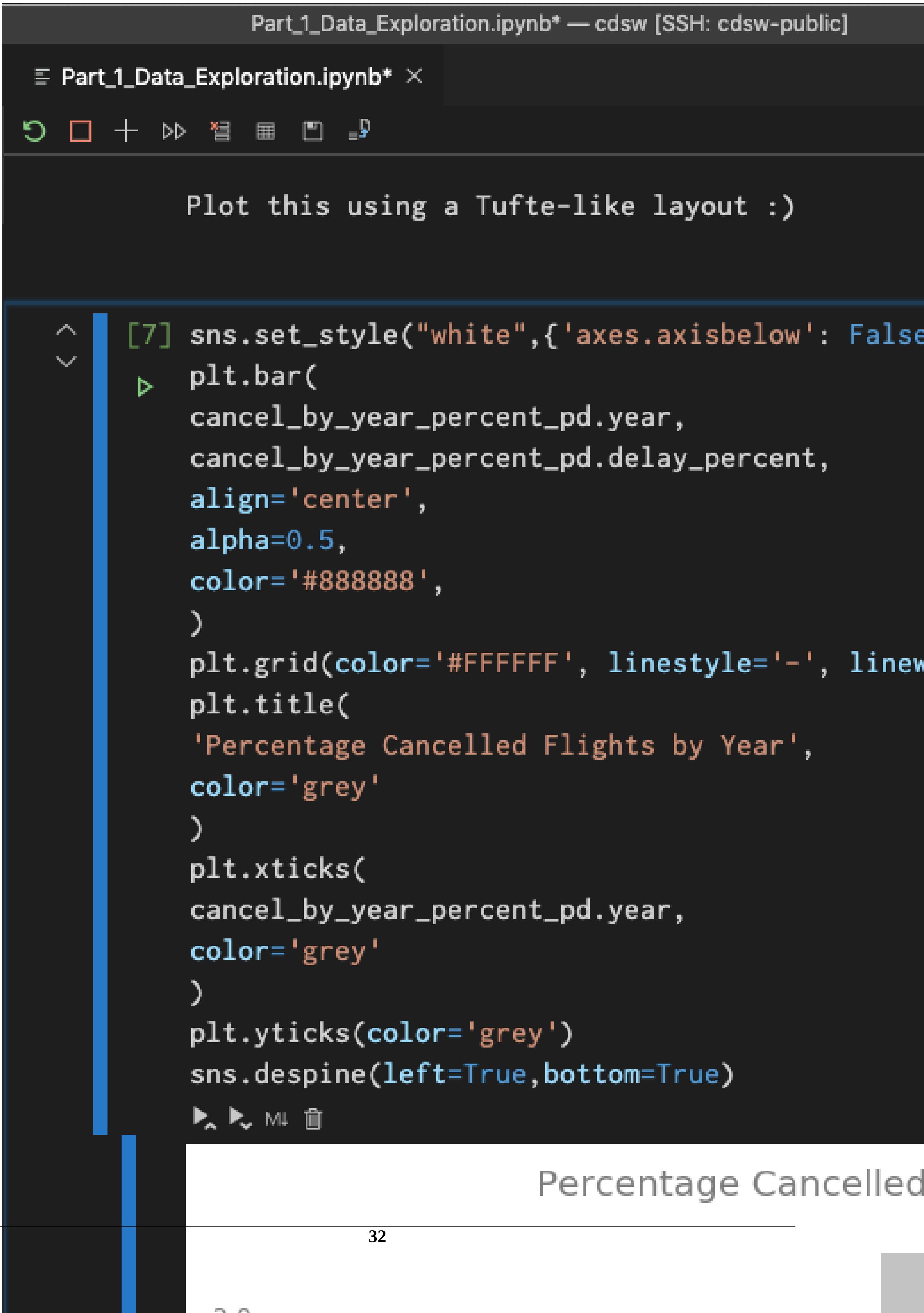
You can use VS Code with Jupyter Notebooks.

### About this task

You can work on Jupyter Notebooks within VS Code. This gives you all the great code completion, syntax highlighting and documentation hints that are part of the VS Code experience and the interactivity of a Jupyter Notebook. Any changes you make to the Notebook will be reflected on the CDSW / CML server and can be viewed online using Jupyter Notebook as a browser based editor.

### Procedure

1. After you install the Jupyter Notebooks, you can use it inside VS Code.

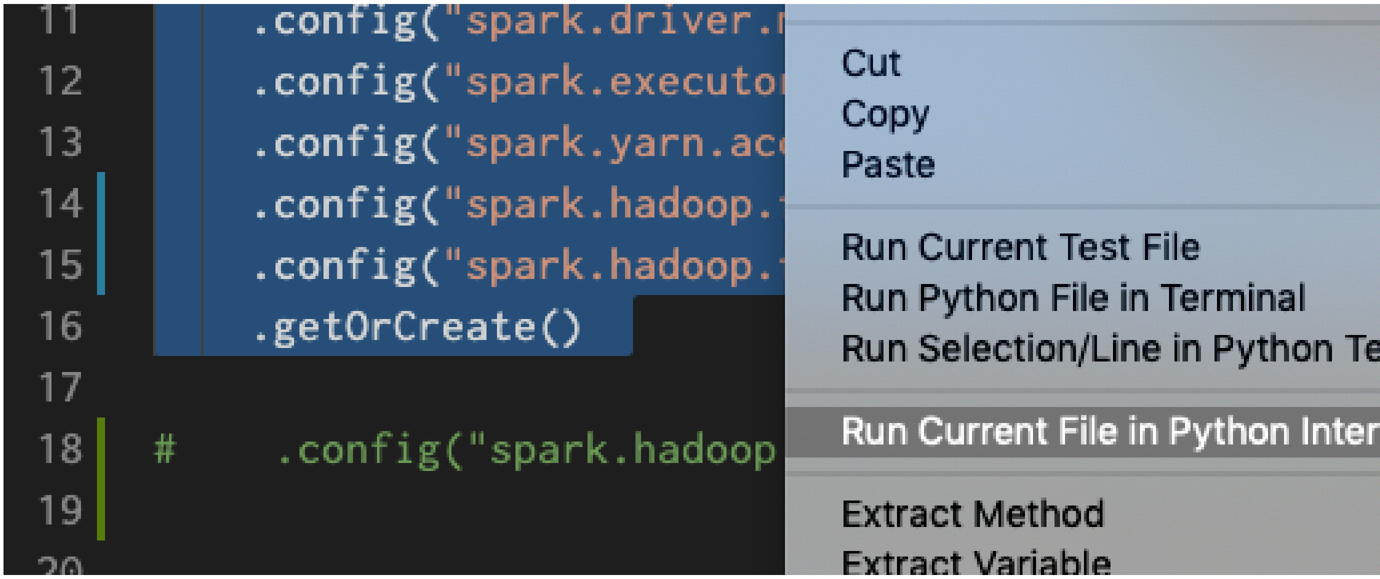

Part_1_Data_Exploration.ipynb* — cdsw [SSH: cdsw-public]
Part_1_Data_Exploration.ipynb*
M↓
Percentage Cancelled

Plot this using a Tufte-like layout :)

```
[7] sns.set_style("white",{'axes.axisbelow': False
    plt.bar(
    cancel_by_year_percent_pd.year,
    cancel_by_year_percent_pd.delay_percent,
    align='center',
    alpha=0.5,
    color='#888888',
    )
    plt.grid(color='#FFFFFF', linestyle='-', linew
    plt.title(
    'Percentage Cancelled Flights by Year',
    color='grey'
    )
    plt.xticks(
    cancel_by_year_percent_pd.year,
    color='grey'
    )
    plt.yticks(color='grey')
    sns.despine(left=True,bottom=True)
```

2. Another feature that you can use with VS Code is running a temporary Notebook for executing random code snippets. Select code you want to run, right click and click Run Current File in Python Interactive Window.

   This is less robust though and will create many Untitled*.ipynb files in your home directory.



```
        .config("spark.driver.r
        .config("spark.executor
        .config("spark.yarn.acc
        .config("spark.hadoop.
        .config("spark.hadoop.
        .getOrCreate()

#        .config("spark.hadoop
```

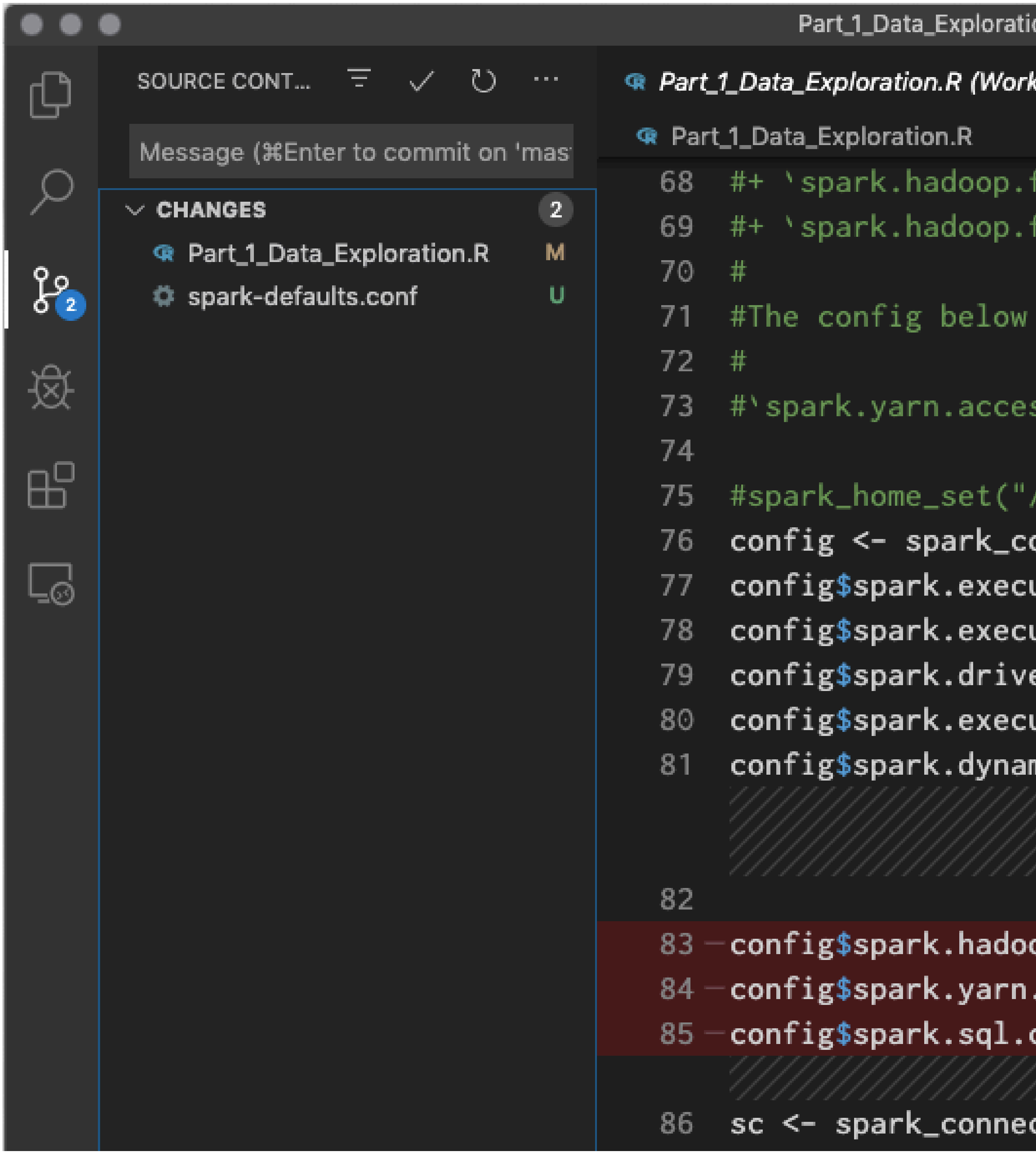
Cut
Copy
Paste
Run Current Test File
Run Python File in Terminal
Run Selection/Line in Python Te
Run Current File in Python Inter
Extract Method
Extract Variable

### (Optional) Using VS Code with Git integration

VS Code has substantial Git integration.

### About this task

If you created your project from a git repo or a custom template, your changes and outside changes made to the repo will automatically appear.


Part_1_Data_Exploratio
SOURCE CONT...
Message (⌘Enter to commit on 'mas
CHANGES 2
Part_1_Data_Exploration.R M
spark-defaults.conf U
2
Part_1_Data_Exploration.R (Work
Part_1_Data_Exploration.R

```
#+ 'spark.hadoop.f
#+ 'spark.hadoop.f
#
#The config below
#
#'spark.yarn.acces

#spark_home_set("/
config <- spark_co
config$spark.execu
config$spark.execu
config$spark.drive
config$spark.execu
config$spark.dynam

config$spark.hadoo
config$spark.yarn.
config$spark.sql.c

sc <- spark_connec
```

## Limiting files in Explorer view

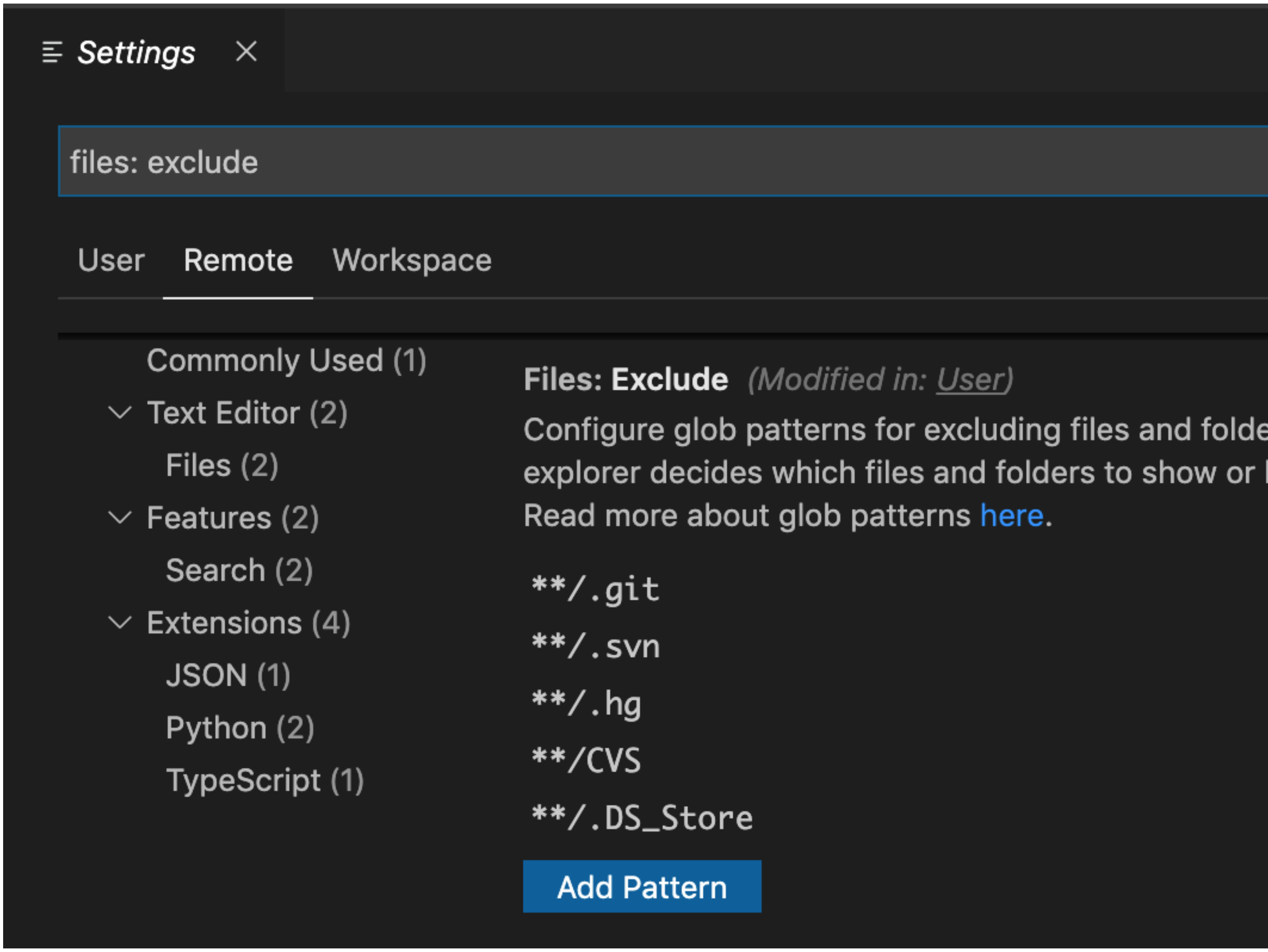
You can limit the number of files shown in the Explorer view.

### About this task

If you end up with many of .[something] directories, in /home/cdsw, it can be difficult to navigate.

### Procedure

If you add the **/.* pattern to the Files: Exclude setting, it will hide all those files and directories.



**Files: Exclude** *(Modified in: User)*

Configure glob patterns for excluding files and folde
explorer decides which files and folders to show or
Read more about glob patterns here.

```
**/.git
**/.svn
**/.hg
**/CVS
**/.DS_Store
```