

Machine Learning 1.5.1

Projects in Cloudera Machine Learning

Date published: 2020-07-16

Date modified: 2023-01-31

The Cloudera logo is displayed in a bold, orange, sans-serif font. The word "CLOUDERA" is written in all caps, with a stylized 'E' that has a horizontal bar extending to the right.

<https://docs.cloudera.com/>

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.

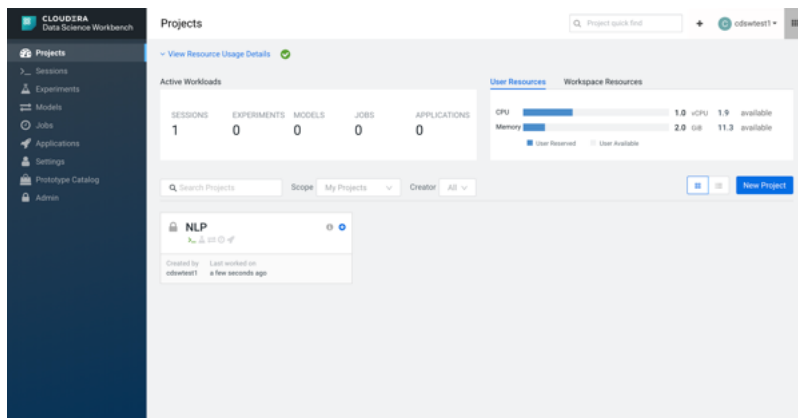
Contents

Projects in Cloudera Machine Learning.....	5
Creating a Project with Legacy Engine Variants.....	5
Creating a project from a password-protected Git repo.....	6
Configuring Project-level Runtimes.....	7
Adding Project Collaborators.....	7
Modifying Project Settings.....	8
Managing Project Files.....	9
Custom Template Projects.....	10
Deleting a Project.....	10
Native Workbench Console and Editor.....	11
Launch a Session.....	11
Run Code.....	12
Access the Terminal.....	13
Stop a Session.....	13
Workbench editor file types.....	14
Environmental Variables.....	14
Third-Party Editors.....	15
Modes of Configuring Third-Party Editors.....	16
Configure a Browser IDE as an Editor.....	17
Test a Browser IDE in a Session Before Installation.....	17
Configure a Browser IDE at the Project Level.....	18
Configure a Browser IDE at the Legacy Engine Level.....	19
Configure a Local IDE using an SSH Gateway.....	21
Configure PyCharm as a Local IDE.....	21
Add Cloudera Machine Learning as an Interpreter for PyCharm.....	22
Configure PyCharm to use Cloudera Machine Learning as the Remote Console.....	23
(Optional) Configure the Sync Between Cloudera Machine Learning and PyCharm.....	23
Configure VS Code as a Local IDE.....	23
Download cdsctl and Add an SSH Key.....	24
Initialize an SSH Connection to Cloudera Machine Learning for VS Code.....	24
Setting up VS Code.....	26
(Optional) Using VS Code with Python.....	31
(Optional) Using VS Code with R.....	36
(Optional) Using VS Code with Jupyter.....	37
(Optional) Using VS Code with Git integration.....	43
Limiting files in Explorer view.....	44
Git for Collaboration.....	45
Linking an Existing Project to a Git Remote.....	46
Web Applications Embedded in Sessions.....	46

Projects in Cloudera Machine Learning

Projects form the heart of Cloudera Machine Learning. They hold all the code, configuration, and libraries needed to reproducibly run analyses. Each project is independent, ensuring users can work freely without interfering with one another or breaking existing workloads.

Access the Projects page by clicking Projects in the navigation panel. The Projects page gives you a quick summary of project information.



- Resource Usage Details - A collapsible section that displays resource usage.
 - Active Workloads - If there are active workloads running, this section describes the number of Sessions, Experiments, Models, Jobs, and Applications that are running.
 - User Resources and Workspace Resources
 - Click on the User Resources tab to see the CPU and memory resource usage for the user. The maximum usage of the vCPU and GB is calculated based on whether or not you have a quota. If you have a quota, the maximum usage will be based on your quota. If you don't have a quota, the maximum usage will be what is available on the cluster. If you have a GPU, you'll also see the GPU usage.
 - Click on the Workspace Resources tab to see usage overall.
- Search Projects - Enter a term for keyword search across Project names.
- Scope - An additional filter only viewable by Administrators.
 - Selecting My Projects displays only the Projects that you have created or are a Collaborator of.
 - Selecting All Projects displays all Projects on the ML Workspace.
- Creator - An additional filter to only display Projects created by a specified user.
- Projects View Selector - A setting that enables you to display Projects in a summary card-based view or a detailed table-based view.

The following topics describe how to create and manage projects in Cloudera Machine Learning.

Creating a Project with Legacy Engine Variants

Projects create an independent working environment to hold your code, configuration, and libraries for your analysis. This topic describes how to create a project with Legacy Engine variants in Cloudera Machine Learning.

Procedure

1. Go to Cloudera Machine Learning and on the left sidebar, click Projects.
2. Click New Project.

3. If you are a member of a team, from the drop-down menu, select the Account under which you want to create this project. If there is only one account on the deployment, you will not see this option.
4. Enter a Project Name.
5. Select Project Visibility from one of the following options.
 - Private - Only project collaborators can view or edit the project.
 - Team - If the project is created under a team account, all members of the team can view the project. Only explicitly-added collaborators can edit the project.
 - Public - All authenticated users of Cloudera Machine Learning will be able to view the project. Collaborators will be able to edit the project.
6. Under Initial Setup, you can either create a blank project, or select one of the following sources for your project files.
 - Built-in Templates - Template projects contain example code that can help you get started with Cloudera Machine Learning. They are available in R, Python, PySpark, and Scala. Using a template project is not required, but it helps you start using Cloudera Machine Learning right away.
 - Custom Templates - Site administrators can add template projects that are customized for their organization's use-cases. For details, see *Custom Template Projects*.
 - Local - If you have an existing project on your local disk, use this option to upload compressed files or folders to Cloudera Machine Learning.
 - Git - If you already use Git for version control and collaboration, you can continue to do so with Cloudera Machine Learning. Specifying a Git URL will clone the project into Cloudera Machine Learning. To use a password-protected Git repository, see *Creating a project from a password-protected Git repo*.
7. Click Create Project. After the project is created, you can see your project files and the list of jobs defined in your project.

Note that as part of the project filesystem, Cloudera Machine Learning also creates the following `.gitignore` file.

```
R
node_modules
*.pyc
.*
!.gitignore
```

8. (Optional) To work with team members on a project, add them as collaborators to the project.

Related Information

[Custom Template Projects](#)

[Adding a new SSH key to your GitHub account](#)

[Creating a project from a password-protected Git repo](#)

Creating a project from a password-protected Git repo

You can create projects in CML by replicating the project files from a Git repo. The Git repo can be public, or it can be private, accessed by SSH or HTTPS authentication.

When you create a project, you can choose to create it from a Git repository. In the New Project page, under Initial Setup, choose the Git tab. Paste the Git URL in Git URL of Project.

There are two ways to authenticate a password-protected repo: SSH and HTTPS. The SSH key is automatically generated by CML for each user or team account.

To clone a repo with SSH:

- Make sure you have a public SSH key added to your Github account. For more information, see *Adding an SSH Key to GitHub*.
- In the Git interface, select `Code SSH` and copy the URL.
- Paste the URL into Git URL of Project. It should appear similar to `git@github.com:someuser/somerepo.git`.

To clone a repo with HTTPS:

- In the Git interface, select `Code SSH` and copy the URL.
- Paste the URL into Git URL of Project.
- Insert the repo username and password into the URL, like so: `https://<username>:<password>@github.com/someuser/somerepo.git`

Continue with creating the project, and the files will be imported from the Git repository.

Configuring Project-level Runtimes

If you've specified project-level Runtimes, you can view your chosen Runtime configuration by clicking `Project Settings Runtime/Engine`. Your chosen Runtimes are listed under `Available Runtimes`.

- If the `Available Runtimes` table is empty, users can select from all Runtimes available in the deployment to start new sessions or workloads.
- The filtering options in `Project Settings` only affect the user interface. The filtering options do not apply when projects are accessed using `API v2`.
- You can remove an available Runtime by clicking the corresponding "x" in the right most column. Runtimes can only be removed if there are no active workloads using them.
- You can add additional Runtimes by clicking `Add Runtime` and choosing additional Runtimes.
- If there is a newer version of a Runtime, a warning icon displays next to the appropriate Runtimes. You can apply the latest version by clicking `Add Latest`. The older Runtime version is not removed from the table. However, you can remove it by clicking the "x" in the right most column.
- The `Available Runtimes` table shows related counters for non-interactive workloads.

Adding Project Collaborators

This topic shows you how to invite colleagues to collaborate on a project.

About this task

For a project created under your personal account, anyone who belongs to your organization can be added as a collaborator. For a project created under a team account, you can only add collaborators that already belong to the team. If you want to work on a project that requires collaborators from different teams, create a new team with the required members, and then create a project under that account. If your project was created from a Git repository, each collaborator must create the project from the same central Git repository.

You can grant project collaborators one of three levels of access:

- **Viewer** - Read-only access to code, data, and results.
- **Operator** - Read-only access to code, data, and results. Additionally, Operators can start and stop existing jobs in the projects that they have access to.
- **Contributor** - Can view, edit, create, and delete files and environmental variables, run sessions/experiments/jobs/models and run code in running jobs. Additionally, Contributors can set the default engine for the project.
- **Admin** - Has complete access to all aspects of the project. This includes the ability to add new collaborators, and delete the entire project.

Adding Teams as collaborators

Any administrator, owner, contributor or operator can add a team as a collaborator on a project. They can be added to team or individual projects. When a team is added, if there are members of the team who are already members of the project, their permissions can be upgraded to the permissions they have on the team (`Inherit mode`). Alternatively, the permissions applied to the team are applied equally to each member on the team, even if they have different permissions independently on the project (`Override mode`).

**Note:****Collaborating Securely on Projects**

Before adding project collaborators, you must remember that assigning the Contributor or Admin role to a project collaborator is the same as giving them write access to your data in CDH. This is because project contributors and project administrators have write access to all your project code (including any library code that you might not be actively inspecting). For example, a contributor/admin could modify project file(s) to insert code that deletes some data on the CDH cluster. The next time you launch a session and run the same code, it will appear as though you deleted the data yourself.

Additionally, project collaborators also have access to all actively running sessions and jobs. This means that a malicious user can easily impersonate you by accessing one of your active sessions. Therefore, it is extremely important to restrict project access to trusted collaborators only. Note that site administrators can restrict this ability by allowing only session creators to run commands within their own active sessions.

For these reasons, Cloudera recommends using Git to collaborate securely on shared projects. This will also help avoid file modification conflicts when your team is working on more elaborate projects.

Procedure

1. In Cloudera Machine Learning, navigate to the project overview page.
2. Click Collaborators.
3. Search for collaborators by either name, team name, or email address and click Add.

Modifying Project Settings

Project contributors and administrators can modify aspects of the project environment such as the engine used to launch sessions, the environment variables, and to create SSH tunnels to access external resources.

Procedure

1. Switch context to the account where the project was created.
2. Click Projects.
3. From the list of projects, select the one to modify.

4. Click Project Settings to open the Project Settings dashboard.

General

Modify the project name, description, visibility (privacy settings), and owner on this page. For Project Owner, only individual users (not Teams) can be assigned, and only a Administrator can change the owner.



Note: After the owner is changed, the previous owner will not be able to see the project, unless they are added as a Collaborator.

Engine

Cloudera Machine Learning ensures that your code is always run with the specific engine version you selected. You can also select the engine version and add third-party editors here.

Advanced

- Environment Variables - If there are any environmental variables that should be injected into all the engines running this project, you can add them to this page. For more details, see *Engine Environment Variables*.
- Ephemeral Storage Settings - Ephemeral storage space is scratch space a CML session, job, application or model can use. For more information, see *Ephemeral storage*.
- Shared Memory Limit - You can specify additional shared memory available to sessions running with the project.



Note: You can specify additional shared memory available to sessions running with the project. The maximum size of this volume is the half of your physical RAM in the node, not including memory used for swap.

SSH Tunnels

In some environments, external databases and data sources reside behind restrictive firewalls. Cloudera Machine Learning provides a convenient way to connect to such resources using your SSH key. For instructions, see *SSH Keys*.

Data Connections

Data Connections connect projects to data sources. For more information, see *Data connection management*.

Delete Project

This page can only be accessed by project administrators. Remember that deleting a project is irreversible. All files, data, sessions, and jobs are removed.

Related Information

[Managing Engines](#)

[Engine Environment Variables](#)

[SSH Keys](#)

[Ephemeral storage](#)

Managing Project Files

Cloudera Machine Learning allows you to move, rename, copy, and delete files within the scope of the project where they live. You can also upload new files to a project, or download project files. For use cases beyond simple projects, Cloudera strongly recommends using *Git for Collaboration* to manage your projects using version control.

Procedure

1. Switch context to the account where the project was created.
2. Click Projects.
3. From the list of projects, click on the project you want to modify. This will take you to the project overview.

4. Click Files.

Upload Files to a Project

Files can only be uploaded within the scope of a single project. Therefore, to access a script or data file from multiple projects, you will need to manually upload it to all the relevant projects.

Click Upload. Select Files or Folder from the dropdown, and choose the files or folder you want to upload from your local filesystem.

In addition to uploading files or a folder, you can upload a .tar file of multiple files and folders. After you select and upload the .tar file, you can use a terminal session to extract the contents:

- a. On the project overview page, click Open Workbench and select a running session or create a new one.
- b. Click Terminal access.
- c. In the terminal window, extract the contents of the .tar file:

```
tar -xvf <file_name>.tar.gz
```

The extracted files are now available for the project.

Download Project Files

Click Download to download the entire project in a .zip file. To download only a specific file, select the checkbox next to the file(s) to be download and click Download.

5. You can also use the checkboxes to Move, Rename, or Delete files within the scope of this project.

Related Information

[Git for Collaboration](#)

Custom Template Projects

Site administrators can add template projects that have been customized for their organization's use-cases. These custom project templates can be added in the form of a Git repository.

Required Role: See User Role Authorization.

To add a new template project, go to Admin Settings . Under the Project Templates section, provide a template name, the URL to the project's Git repository, and click Add.

The added templates will become available in the Template tab on the Create Project page. Site administrators can add, edit, or delete custom templates, but not the built-in ones. However, individual built-in templates can be disabled using a checkbox in the Project Templates table at Admin Settings .

Deleting a Project

This topic demonstrates how to delete a project.

About this task



Important: Deleting a project is an irreversible action. All files, data, and history related to the project will be lost. This includes any jobs, sessions or models you created within the project.

Procedure

1. Go to the project Overview page.
2. On the left sidebar, click Settings.
3. Go to the Delete Project.

- Click Delete Project and click OK to confirm.

Native Workbench Console and Editor

The workbench console provides an interactive environment tailored for data science, supporting R, Python and Scala. It currently supports R, Python, and Scala engines. You can use these engines in isolation, as you would on your laptop, or connect to your CDH cluster.

The workbench UI includes four primary components:

- An editor where you can edit your scripts.
- A console where you can track the results of your analysis.
- A command prompt where you can enter commands interactively.
- A terminal where you can use a Bash shell.

The screenshot illustrates the Native Workbench Console and Editor interface. It features a file explorer on the left, a code editor in the center, and a console/terminal on the right. The code editor shows Python code for data analysis and visualization using Seaborn. The console displays the execution of the code, including the output of a linear regression model and a scatter plot titled "Tips Regression" with a fitted line and marginal histograms. Annotations with blue arrows point to the "Project File System" (file explorer), "Terminal access to running engine" (console), and "Interactive command prompt" (terminal input area).

Typically, you would use the following steps to run a project in the workbench:

Related Information

[Managing Engines](#)

Launch a Session

Sessions allow you to perform actions such as run R or Python code. They also provide access to an interactive command prompt and terminal. This topic demonstrates how to launch a new session.

Procedure

- Navigate to your project's Overview page.
- Click New Session.

3. Check the settings for your session:

You see the following settings:

Editor

Selects the Editor; currently only Workbench is supported and therefore the selector is static.

Kernel

Selects the Kernel. Initially only Python Runtimes are supported.

Engine Image

Displays the Advanced tab in Project Settings and allows you to set environment variables and the shared memory limit.

4. You can modify the engine image used by this session:

a) By Engine Image, click Configure.

Cloudera Machine Learning displays the Project Settings page.

b) Select the Runtime/Engine tab.

c) Next to Default Engine, select ML Runtime or Legacy Engine.

d) Click Save Engine.

5. Specify your Resource Profile.

This attribute will define how many vCPUs and how much memory will be reserved to run the workload (for example, session including the runtime itself). The minimum configuration is 1vCPU and 2 GB memory.

6. Click Start Session.

The command prompt at the bottom right of your browser window will turn green when the engine is ready. Sessions typically take between 10 and 20 seconds to start.

Run Code

This topic shows you how to enter and run code in the interactive Workbench command prompt or the editor after you launch a session.

The editor is best for code you want to keep, while the command prompt is best for quick interactive exploration.

Command Prompt - The command prompt functions largely like any other. Enter a command and press Enter to run it. If you want to enter more than one line of code, use Shift+Enter to move to the next line. The output of your code, including plots, appears in the console.

```
> ls
```

```
analysis.ipynb  entry.py      predict.py      seaborn-data/
analysis.py     fit.py        predict_with_metrics.py  use_model_metrics.py
cdsw-build.sh* lineage.yaml  README.md
config.yml     pi.py         requirements.txt
```

```
> !pip install beautifulsoup4
```

**Enter a command and
press Enter**

If you created your project from a template, you should see project files in the editor. You can open a file in the editor by clicking the file name in the file navigation bar on the left.

Editor - To run code from the editor:

1. Select a script from the project files on the left sidebar.

When you are done with the session, click Stop in the menu bar above the console, or use code to exit by typing the following command:

R

```
quit()
```

Python

```
exit
```

Scala

```
quit()
```

Sessions automatically stop after an hour of inactivity.

Workbench editor file types

The default workbench editor supports the following file types:

- Text
- CSS
- HTML
- JavaScript
- JSON
- PHP
- Scala
- C++
- C#
- CLike
- Java
- CoffeeScript
- R
- Julia
- Ruby
- Clojure
- Perl
- Python
- SASS
- Lua
- SQL
- Diff
- Markdown
- YAML
- Haxe

Environmental Variables

Environmental variables help you customize engine environments, both globally and for individual projects/jobs.

For example, if you need to configure a particular timezone for a project or increase the length of the session or job timeout windows, you can use environmental variables to do so.

For a list of the environmental variables you can configure and instructions on how to configure them, see *Engine Environment Variables*.

You can create new environment variables in the following pages:

- Project Settings Advanced
- New Job Job Settings
- New Application Application Settings
- User Settings Environment Variables
- Site Administration Runtime

To add an environmental variable:

1. Enter the name and value.
2. Click Add
3. Optionally, click Hide/Show to hide the value from viewing by users. This can help with hiding sensitive values stored in environment variables, such as passwords. The value is set to Hide by default.

Third-Party Editors

In addition to the built-in Cloudera Machine Learning editor, you can configure Cloudera Machine Learning 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: Custom editors run inside CML sessions. If the CML session is stopped, this may cause unexpected behavior in the editor UI and, in some cases, may result in data loss. You should, therefore, use the custom editor's UI to shut the editor down first. This will automatically end the CML session too.

In JupyterLab you do that by clicking "Shut Down" in the JupyterLab "File" menu." This applies to both engines and Runtimes, and all versions of CML.

When you bring your own editor, you still get many of the benefits Cloudera Machine Learning 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 Machine Learning
- Reuse code in other Cloudera Machine Learning 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 Machine Learning deployment, not the local machine. Browser IDEs run within a Cloudera Machine Learning 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.

In the Cloudera Machine Learning documentation, browser-based IDEs like Jupyter will be referred to as "browser IDEs". IDEs such as PyCharm that run on your machine outside of your browser will be referred to as "local IDEs" because they run on your local machine. You can use the browser or local IDE of your choice to edit and run code interactively.

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 Machine Learning web UI and cannot be completed through an editor.

Modes of Configuring Third-Party Editors

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

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

Workbench editor

The Workbench editor is the built-in editor for Cloudera Machine Learning. 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 Machine Learning project. Base Engine Image v8 and higher ships with Jupyter preconfigured as a browser IDE. You can select it when you start a session or add a different browser IDE. For more information, see *Configure a Browser IDE as an Editor*.

Keep the following in mind when using browser IDEs:

- Engine Version Requirements
 - Browser-based IDEs require Base Engine Image v8 or higher.
- 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 Machine Learning session. Doing so will cause you to lose your session state. For example, if you want RStudio to save your state, including variables, to `~/RData`, exit the RStudio workspace using the power button in the top right of the RStudio UI.
- Depending on the behavior of the browser IDE, multiple users within a project may overwrite each other's state. For example, RStudio state is persisted in `/home/cdsd/RData` that is shared by all users within a project.
- 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 Machine Learning 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 Cloudera Machine Learning 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 Machine Learning. 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 always 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 Machine Learning. Verify that users understand the requirements and adhere to them when configuring their file sync behavior.

- Users should ensure that any IDEs 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."

Related Information

[Configure a Browser IDE as an Editor](#)

[Configure a Local IDE using an SSH Gateway](#)

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 Machine Learning project.

About this task

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 Machine Learning. When you run the code from the IDE, execution is pushed from the IDE to Cloudera Machine Learning.

Base Engine Image v8 and higher for Cloudera Machine Learning comes preconfigured with Jupyter, and any browser IDEs you want to add must be added to Base Engine Image v8 or higher. Jupyter can be selected in place of the built-in Workbench editor when you launch a session, and no additional configuration is required. 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 preinstalled 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 Machine Learning web UI.
2. Open the Cloudera Machine Learning 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.

- 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 3

The following example command installs Jupyter Lab for Python 3:

```
!pip3 install jupyterlab
```

- 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
```

- 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

The following steps are only required if you want to use an editor that is not included in the default engine image that ships with Cloudera Machine Learning.

Before you begin

Before you start, verify that you have installed the IDE of your choice to the project. For information about how to install additional packages to a project, see *Installing Additional Packages*.

About this task

Perform the following steps to add an editor to a project:

Procedure

- Open the Cloudera Machine Learning web UI.
- Go to the project you want to configure an editor for.
- Go to **Settings Editors** and click **New Editor**.
- 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 Machine Learning 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.

- 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 Machine Learning web UI.

Related Information

[Installing Additional Packages](#)

Configure a Browser IDE at the Legacy Engine Level

You can make a browser IDE available to any project within a Cloudera Machine Learning deployment by creating a customized legacy engine image, installing the editor to it, and adding it to the trusted list for a project. Additionally, browser IDEs that require root permission to install, such as RStudio, can only be used as part of a customized legacy engine image.

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 make 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, and you must use Base Engine Image v9 or higher.

The following sample Dockerfile is for RStudio:

```
#Dockerfile
FROM docker.repository.cloudera.com/cdsw/engine:9-cml1.1

WORKDIR /tmp

#Delete the Cloudera repository that is inaccessible because of the payw
all
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 dist-upgrade -y && \
    apt-get install -y --no-install-recommends \
        libapparmor1 \
        libclang-dev \
        lsb-release \
        psmisc \
        sudo

#The command that follows RUN is the same command you used to install the
IDE to test it in a the session.
RUN wget https://download2.rstudio.org/server/trusty/amd64/rstudio-server-
1.2.1335-amd64.deb && \
    dpkg -i rstudio-server-1.2.1335-amd64.deb

COPY rserver.conf /etc/rstudio/rserver.conf

COPY rstudio-cdsw /usr/local/bin/rstudio-cdsw

RUN chmod +x /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/.rstudio/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 necessary if
# you are installing RStudio Professional. See
# https://docs.rstudio.com/ide/server-pro/r-sessions.html#customizing-session-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 Machine Learning workspace, 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.
- 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>
```

6. Add the image to the trusted list in Cloudera Machine Learning:

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

7. Add the new legacy engine to the trusted list for a project:
 - a) Go to the project Settings page.
 - b) Click Engines.
 - c) Select the new customized legacy engine from the dropdown list of available Docker images. Sessions and jobs you run in your project will have access to this engine.
8. Configure RStudio for the project. When this is done, you will be able to select RStudio from the dropdown list of editors on the Launch New Session page.
 - a) Go to Settings > Editors and click New Editor.
 - b) 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
```
 - c) Save the changes.

Related Information

[Docker push](#)

[Limitations](#)

Configure a Local IDE using an SSH Gateway

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

Cloudera Machine Learning relies on the SSH functionality of the editors 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 editor on your machine and the Cloudera Machine Learning deployment.

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

1. Establish an SSH endpoint with the CML CLI client. See [Initialize an SSH Endpoint](#).
2. Configure the local IDE to use Cloudera Machine Learning as the remote interpreter.
3. Optionally, sync files with tools (like `mutagen`, `SSHFS`, or the functionality built into your IDE) from Cloudera Machine Learning 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 Machine Learning.
5. Sync the files you edited locally to Cloudera Machine Learning.
6. Use the Cloudera Machine Learning web UI to perform actions such as deploying a model that uses the code you edited.

You can see an end-to-end example for PyCharm configuration in the *Configure Pycharm as a Local IDE*.

Related Information

[Configure PyCharm as a Local IDE](#)

Configure PyCharm as a Local IDE

Cloudera Machine Learning supports using editors on your machine that allow remote execution and/or file sync over SSH, such as PyCharm.

About this task

This topic describes the tasks you need to perform to configure Cloudera Machine Learning to act as a remote SSH interpreter for PyCharm. Once finished, you can use PyCharm to edit and sync the changes to Cloudera Machine Learning. To perform actions such as deploying a model, use the Cloudera Machine Learning 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.
- You have Contributor permissions for an existing Cloudera Machine Learning project. Alternatively, create a new project you have access to.

Add Cloudera Machine Learning as an Interpreter for PyCharm

In PyCharm, you can configure an SSH interpreter. Cloudera Machine Learning 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 Machine Learning 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. Open PyCharm.
2. Create a new project.
3. Expand Project Interpreter and select Existing interpreter.
4. Click on ... and select SSH Interpreter
5. Select New server configuration and complete the fields:
 - Host: localhost
 - Port: 2222
 - Username: cdsw
6. 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 Machine Learning 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>
```

7. Complete the wizard. Based on the Python version you want to use, enter one of the following parameters:
 - /usr/local/bin/python2
 - /usr/local/bin/python3

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

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

```
/home/cdsw
```

9. Create the project.

Configure PyCharm to use Cloudera Machine Learning as the Remote Console

Procedure

1. In your project, go to Settings and search for Project Interpreter.
Depending on your operating system, Settings may be called Preferences.
2. Click the gear icon and select Show All.
3. Select the Remote Python editor that you added, which is connected to the Cloudera Machine Learning deployment.
4. Add the following interpreter path by clicking on the folder icon:

```
/usr/local/bin/python2.7/site-packages
```

(Optional) Configure the Sync Between Cloudera Machine Learning 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 Machine Learning, 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 Machine Learning deployment but allow you to download and edit code.

Procedure

1. In your project, go to Settings and search for Project Interpreter.
Depending on your operating system, Settings may be called Preferences.
2. Search for Deployment.
3. On the Connection tab, add the following path to the Root path field:

```
/home/cdsw
```

4. Optionally, add a Deployment path on the Mappings tab if the code for your Cloudera Machine Learning project lives in a subdirectory of the root path.
5. 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.
6. Sync for the project file(s) to your machine and begin editing.

Configure VS Code as a Local IDE

About this task

Cloudera Machine Learning 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 Machine Learning 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 Machine Learning. To perform actions such as deploying a model, use the Cloudera Machine Learning 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 Machine Learning 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 Machine Learning 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 AAAAB3NzaC1yc2EAAAADAQABAAQCAQCha2J5mW3i3BgtZ25/FOsxywpLVkx1RgmZunI`
5. In SSH public keys for session access, paste the key.

What to do next

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

Initialize an SSH Connection to Cloudera Machine Learning for VS Code

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

Procedure

1. Log in to Cloudera Machine Learning 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 Machine Learning.

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 Machine Learning web UI. The public key could be rejected when the new ssh key pair is generated with a special name such as `id_rsa_system`. 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 Machine Learning 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 Machine Learning 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

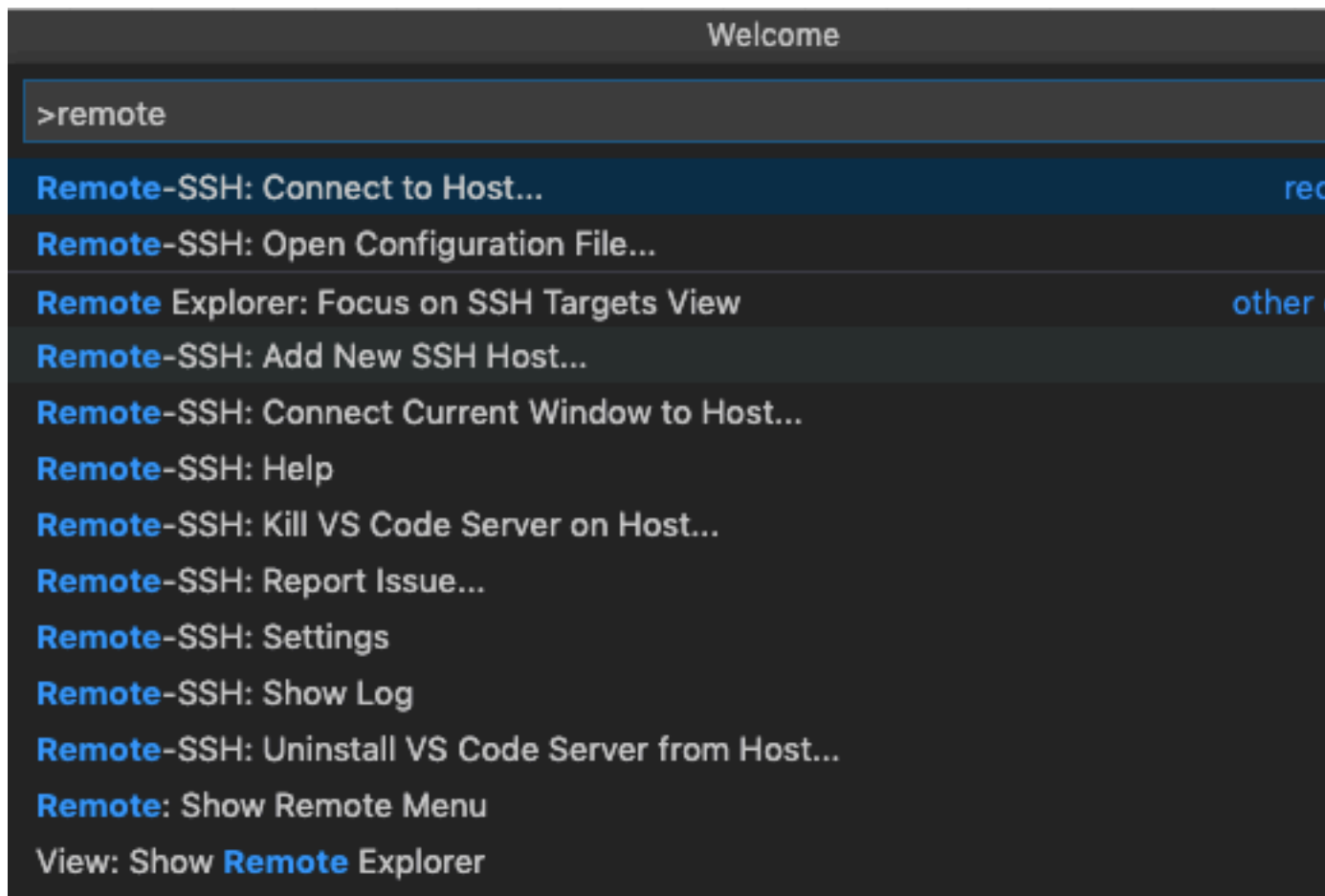
About this task

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

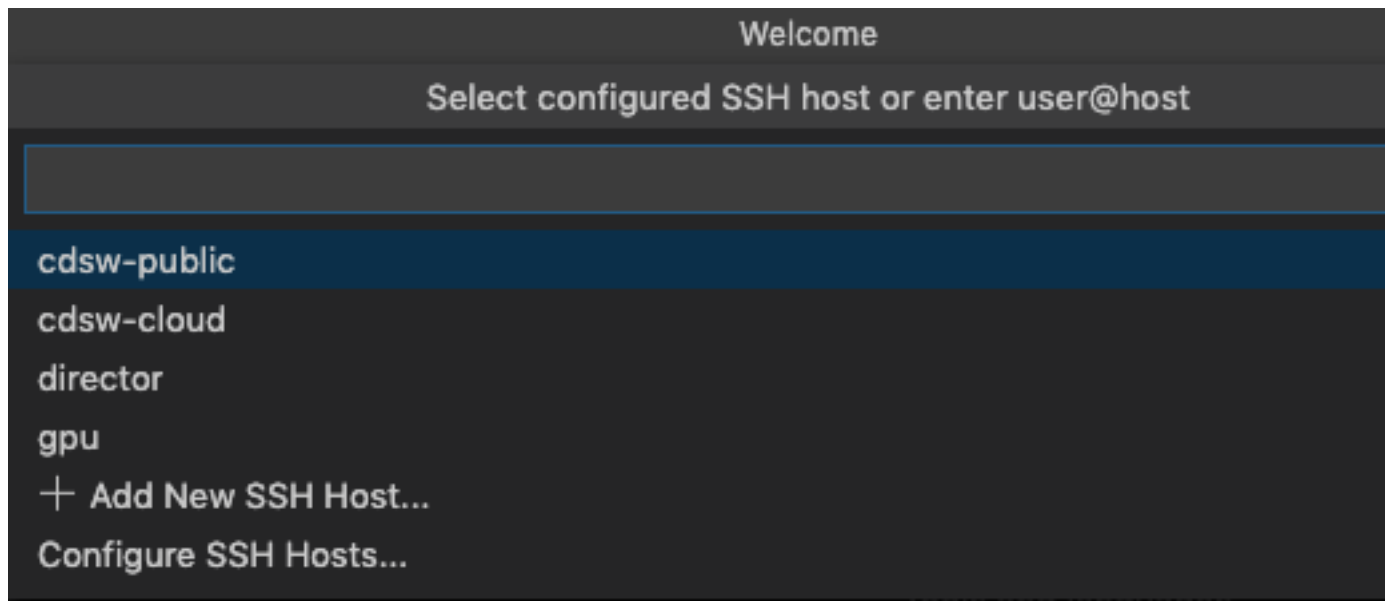
Procedure

1. Verify that the SSH endpoint for Cloudera Machine Learning 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.

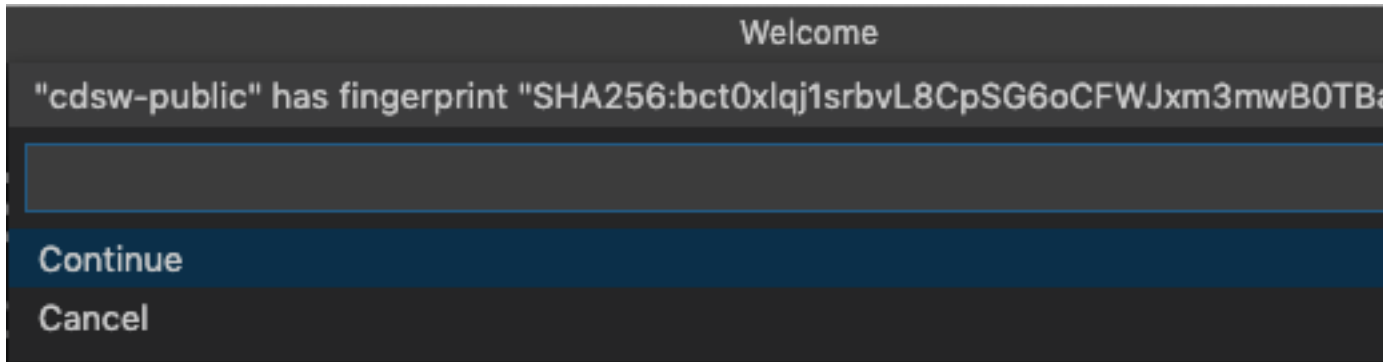


4. Connect to the host you added previously.



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.

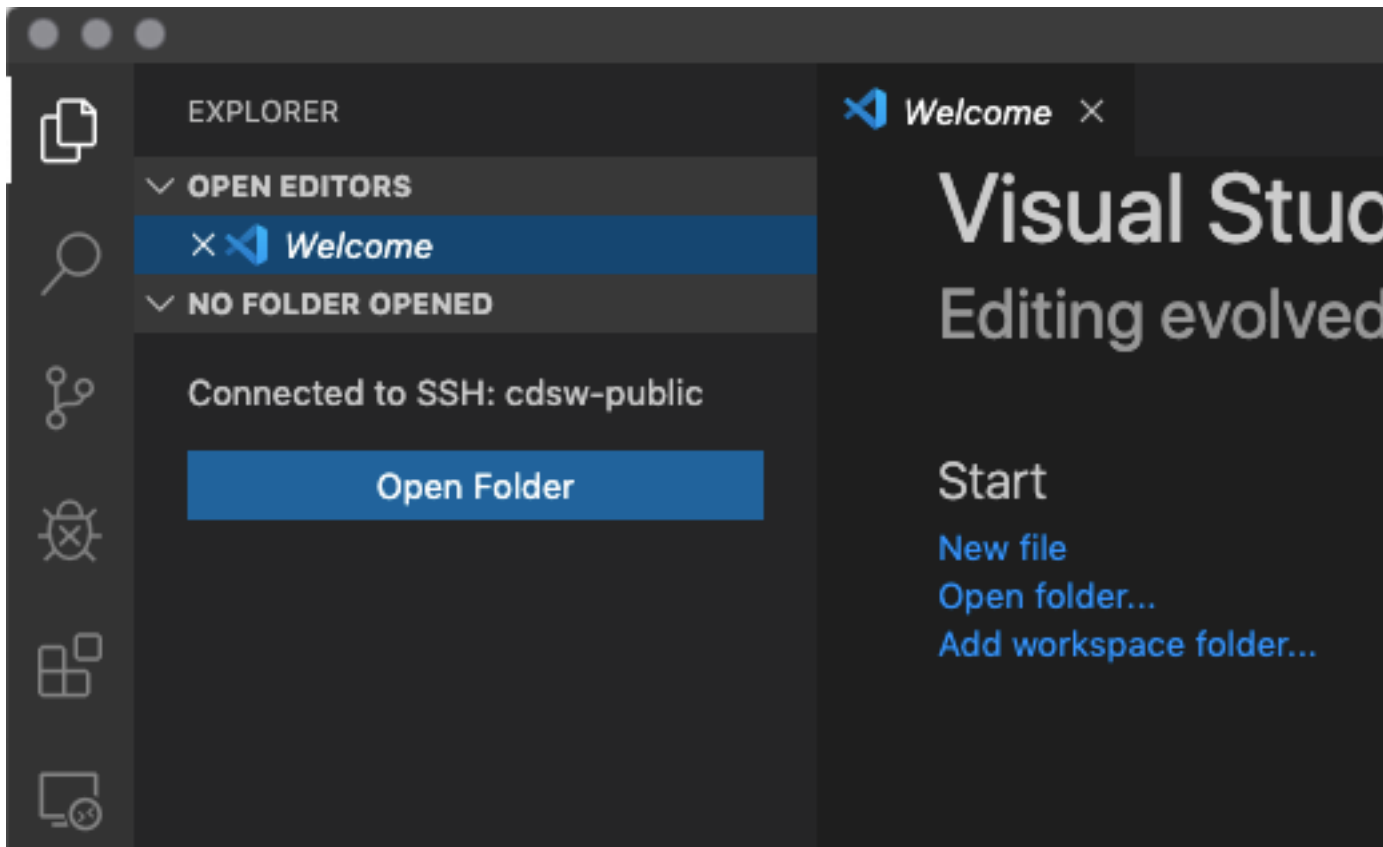


While VS Code connects and sets up the remote connection, it installs some helper applications on the Cloudera Machine Learning server. Sometimes the remote session dies. Click Retry or if it's taking a long time, restart the remote session and it will recover.



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.



7. From the Explorer view, you can edit any of the files on your Cloudera Machine Learning server.

The image shows a code editor interface with a dark theme. On the left is the Explorer sidebar, and on the right is the code editor.

EXPLORER

- 1 OPEN EDITORS 1 UNSAVED
 - als.py /home/cdsw
 - test.R /home/cdsw
 - sql.py 1
 - pi.py /home/cdsw
- CDSW [SSH: CDSW-CLOUD]
 - > c9sdk
 - > code-server2
 - > resources
 - als.py
 - avro_inputformat.py
 - exportedCardTmp_4.ipynb
 - kmeans.py
 - LICENSE.txt
 - log4j.properties
 - logistic_regression.py
 - pagerank.py
 - pi.py
 - sort.py
 - spark-defaults.conf
 - sql.py 1
 - test.ipynb
 - test.R
 - transitive_closure.py
 - wordcount.py

Code Editor (sql.py)

```

1  # # Spark-SQL f
2  #
3  # This example
4
5  from __future__
6  import os
7  import sys
8  from pyspark.sql
9  from pyspark.sql
10
11  spark = SparkSe
12  .builder\
13  .appName("P
14  .getOrCreat
15
16  # A list of Row
17  rows = FRow(

```

PROBLEMS 1 OUTPUT DEB

cdsw@s0zmhi87fa58u7ya

Using the Explorer view, you remotely edit and modify your Cloudera Machine Learning 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

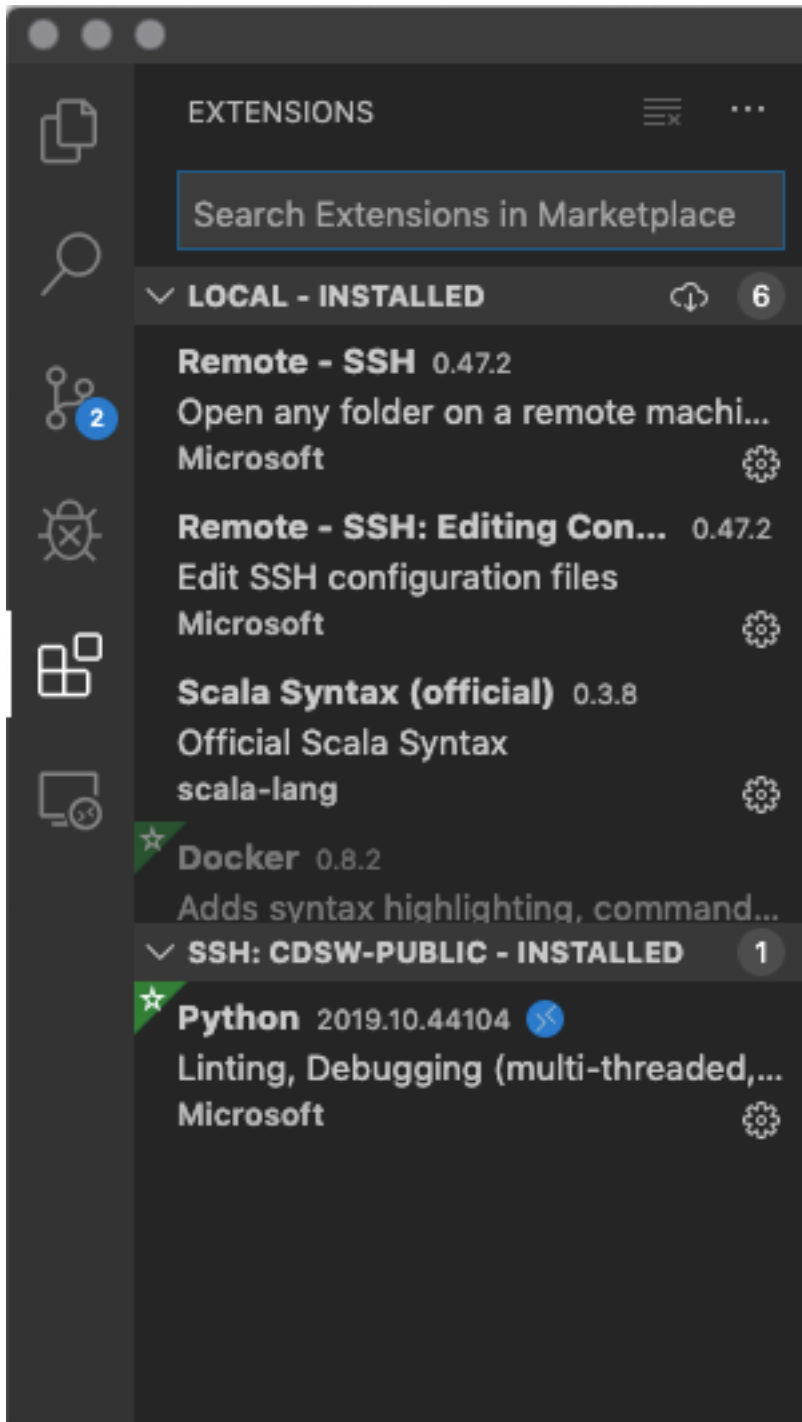
You can use VS Code with Python.

About this task

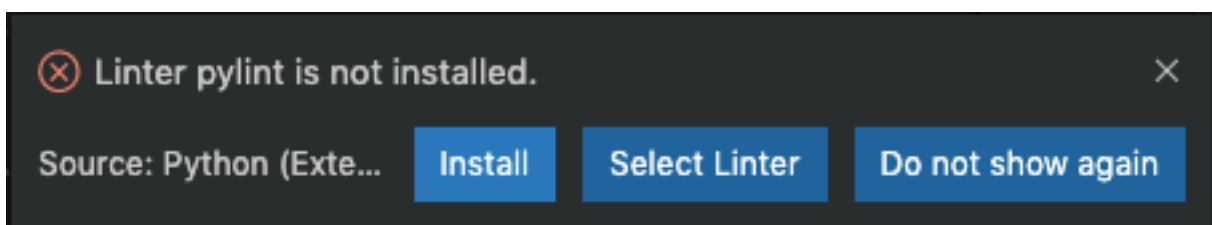
To take full advantage of VS Code Python tools, you must install the Python extension into the remote ssh session. You must install the extension the first time you connect a newly configured remote session.

Procedure

1. Install the Python extension.



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



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 Machine Learning. 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.

The image shows a code editor interface with a dark theme. On the left is the Explorer sidebar, and on the right is the code editor. The Explorer sidebar shows a project structure under 'CDSW [SSH: CDSW-PUBLIC]'. The code editor shows Python code for Spark, including 'flight_df = spark...' and 'df.select(...)'. The bottom status bar shows '68' problems.

EXPLORER

- OPEN EDITORS 1 UNSAVED
 - Part_1_Data_Exploration.R M
 - Part_2_Data_Engineeri... 9+, M
- CDSW [SSH: CDSW-PUBLIC]
 - 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

Part_2_Data_Engineering.py

```
56 flight_df = spa
57 path="s3a://m
58 header=True,
59 schema=schema
60 )
61
62 from pyspark.sql
63 from pyspark.sql
64
65 # pad_time = ud
66
67 # df.select("CR
```

PROBLEMS 68 OUTPUT DE

```
... )
>>>
>>> flight_df = spark
... path="s3a://ml-t
... header=True,
... schema=schema
... )
>>> flight_df.show()
+-----+
-----+-----+
-----+
-----+-----+
| FL_DATE |
XI_OUT | WHEELS_OFF | WHEE
DIVERTED | CRS_ELAPSED_T
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.

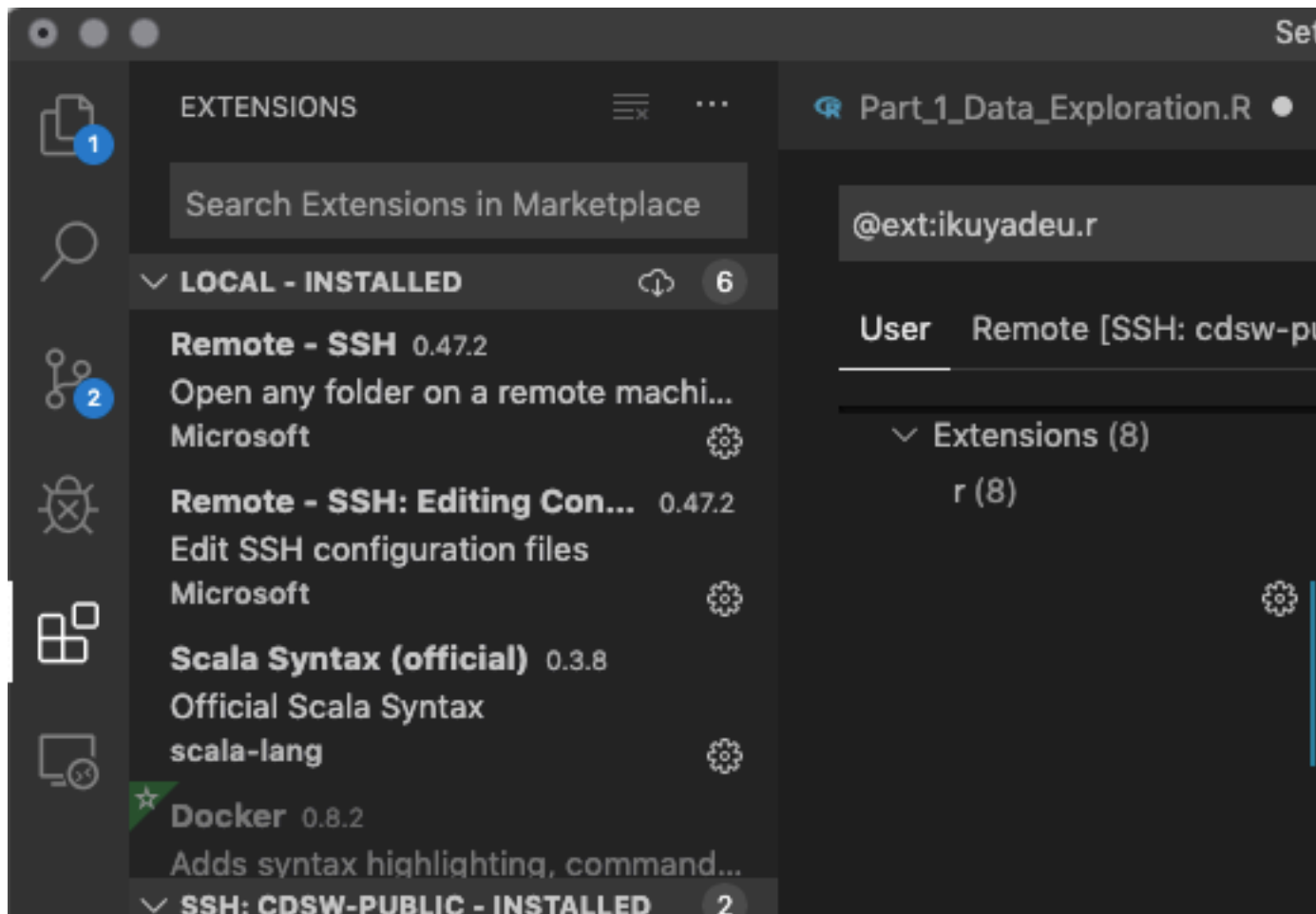
About this task

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.

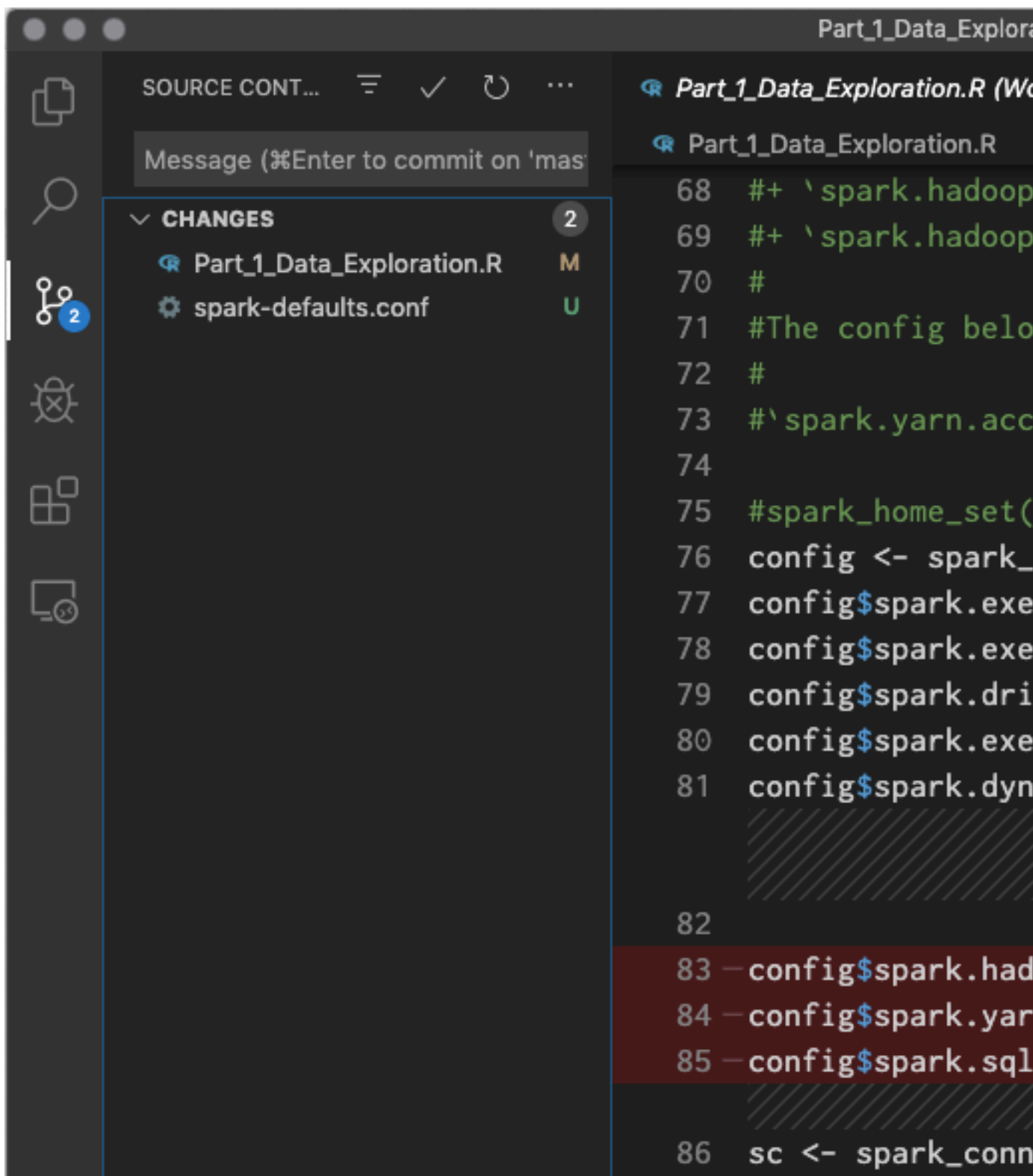
Procedure

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.



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



(Optional) Using VS Code with Jupyter

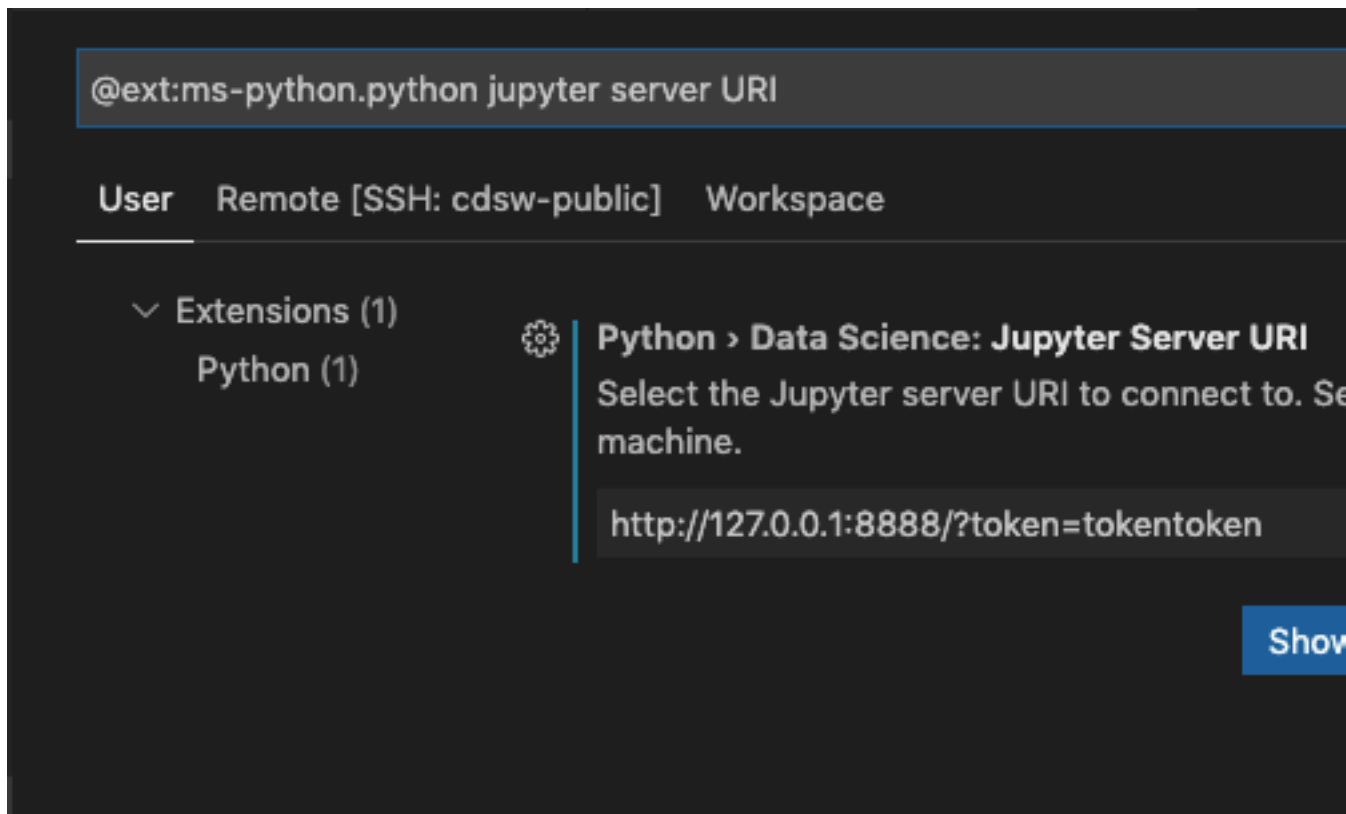
You can use VS Code with Jupyter Notebooks.

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. Because of the way CML uses the internal networking and port forwarding of Kubernetes, when VS Code launches a Jupyter Server it binds to the wrong address and access is blocked. You therefore have to launch your own Jupyter Server and tell VS code to connect to that.
 - a) The first setting you need to set is the Python > Data Science: Jupyter Server URI setting. Set this to `http://127.0.0.1:8888/?token=[some-token]`.



- b) Then you need to open a terminal to launch a Jupiter Notebook server. You can launch it using: `/usr/local/bin/jupyter-notebook --no-browser --ip=127.0.0.1 --NotebookApp.token=[some-token] --NotebookApp.allow_remote_access=True`. This creates a Jupyter server that any new Notebooks you launch will run in.

PROBLEMS 68

OUTPUT

DEBUG CONSOLE

TERMINAL

```
^C Serving notebooks from local directory: /home/cds  
0 active kernels  
The Jupyter Notebook is running at:  
http://127.0.0.1:8888/?token=...  
or http://127.0.0.1:8888/?token=...  
Shutdown this notebook server (y/[n])? y  
[C 16:51:39.751 NotebookApp] Shutdown confirmed  
cdsw@c6cmjsscniinp324:~$ /usr/local/bin/jupyter-notebo  
pp.token='token' --NotebookApp.allow_remote_acces  
[I 16:51:45.043 NotebookApp] Serving notebooks from lo  
[I 16:51:45.043 NotebookApp] The Jupyter Notebook is r  
[I 16:51:45.043 NotebookApp] http://127.0.0.1:8888/?to  
[I 16:51:45.043 NotebookApp] or http://127.0.0.1:8888  
[I 16:51:45.043 NotebookApp] Use Control-C to stop thi  
to skip confirmation).
```


2. 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* ×



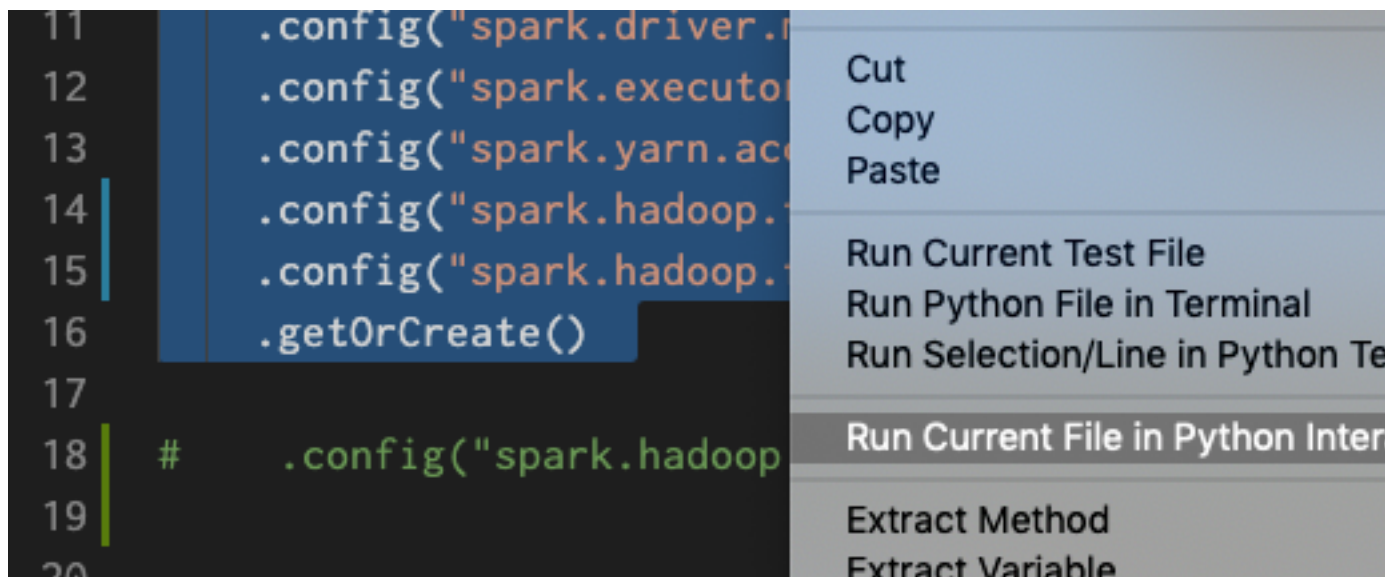
Plot this using a Tufte-like layout :)

```
^ [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='-', linewidth=1)
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)
```



Percentage Cancelled

- 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.



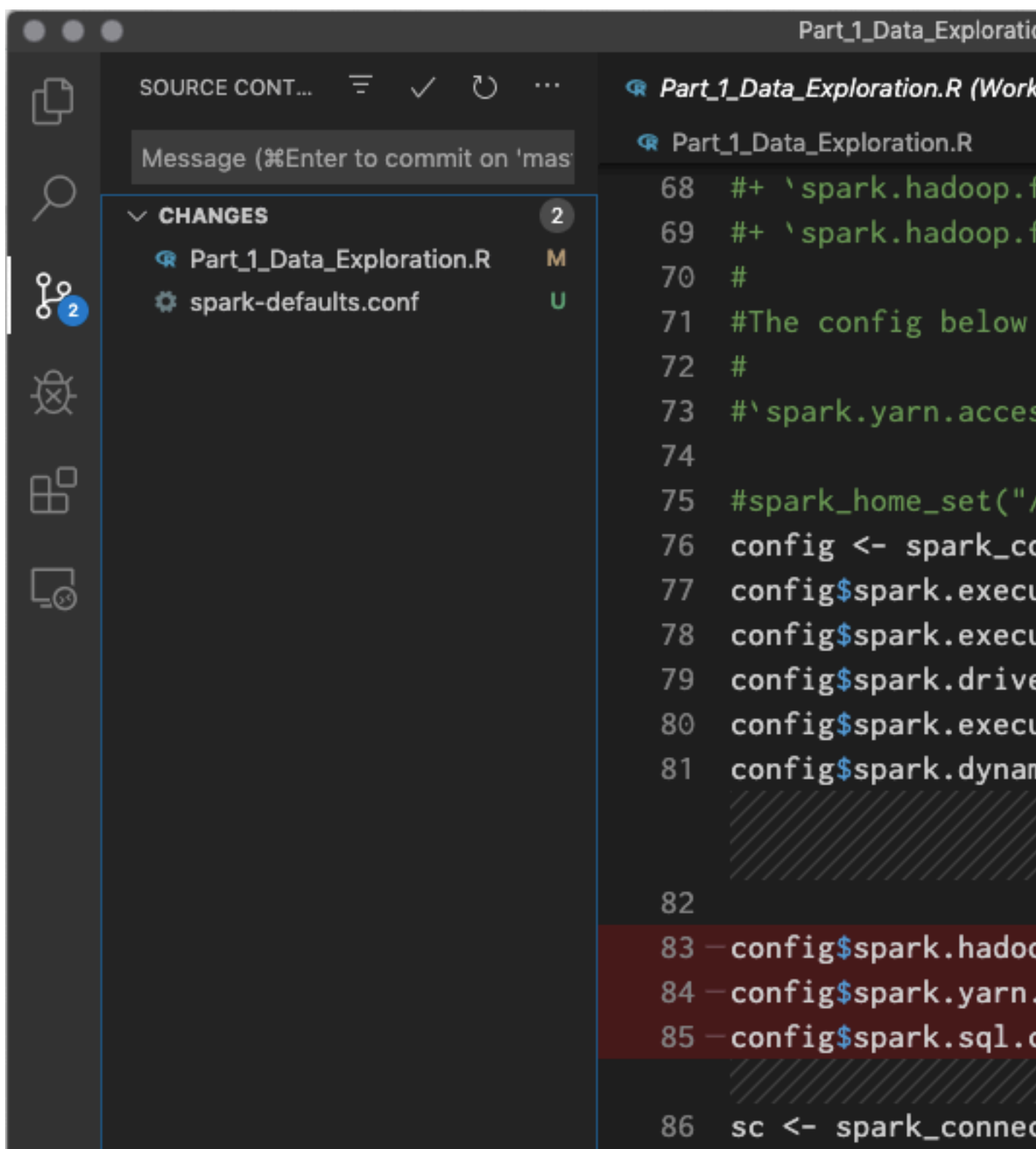
The image shows a screenshot of the Visual Studio Code editor. The code editor is dark-themed and contains Python code. Lines 11 through 17 show a series of `.config()` calls for Spark properties: `spark.driver.extraJavaOptions`, `spark.executor.extraJavaOptions`, `spark.yarn.auxiliaryJars`, `spark.hadoop`, and `spark.hadoop`. Line 16 shows `.getOrCreate()`. Line 18 shows a comment `# .config("spark.hadoop`. A context menu is open over the code, with options: 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, and Extract Variable. The option 'Run Current File in Python Inter' is highlighted.

(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.



Limiting files in Explorer view

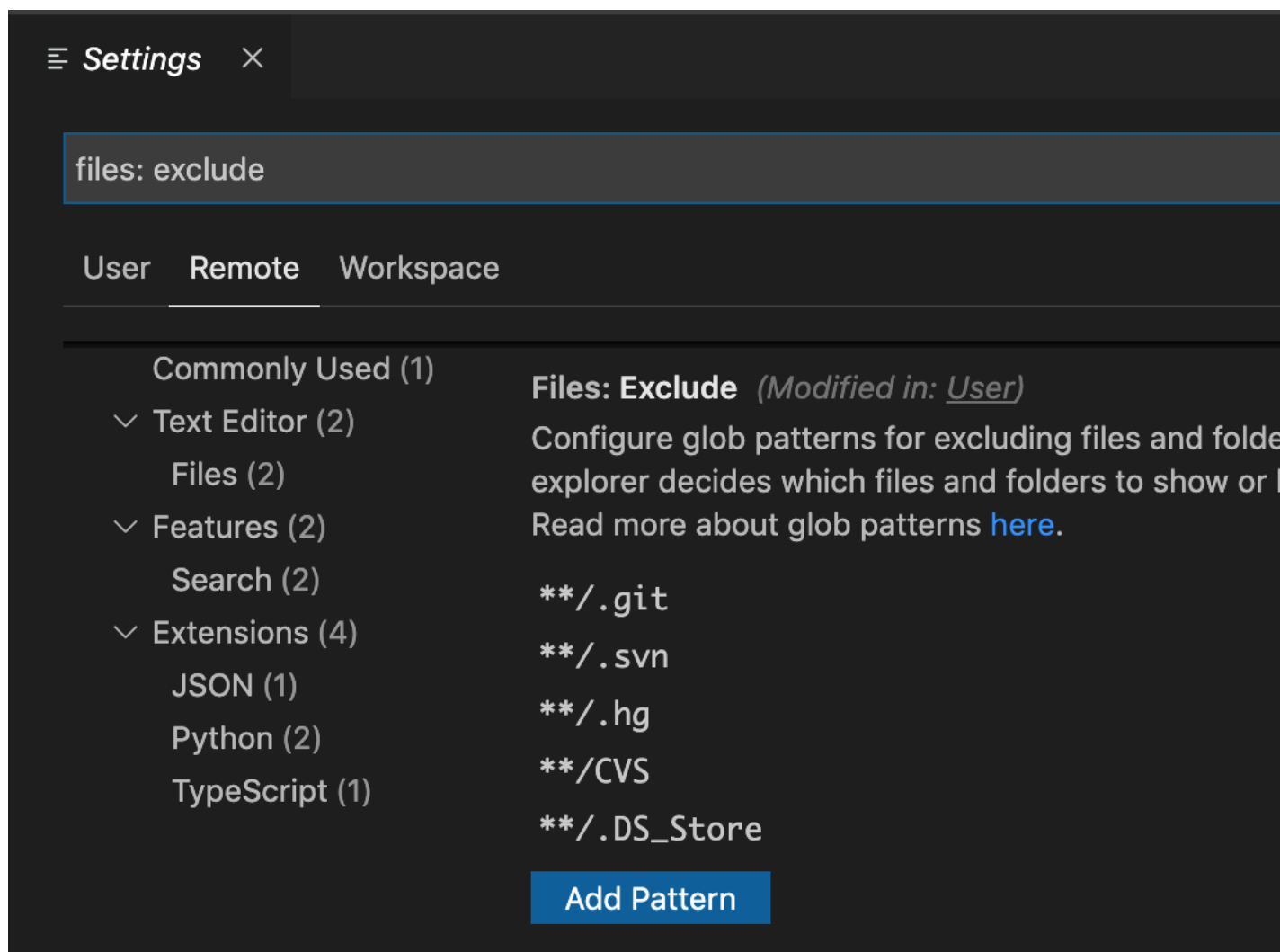
You can limit the number of files shown in the Explorer view.

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.



Git for Collaboration

Cloudera Machine Learning provides seamless access to Git projects. Whether you are working independently, or as part of a team, you can leverage all of benefits of version control and collaboration with Git from within Cloudera Machine Learning.

Teams that already use Git for collaboration can continue to do so. Each team member will need to create a separate Cloudera Machine Learning project from the central Git repository. For anything but simple projects, Cloudera recommends using Git for version control. You should work on Cloudera Machine Learning the same way you would work locally, and for most data scientists and developers that means using Git.

Cloudera Machine Learning does not include significant UI support for Git, but instead allows you to use the full power of the command line. If you launch a session and open a Terminal, you can run any Git command, including `init`, `add`, `commit`, `branch`, `merge` and `rebase`. Everything should work exactly as it does locally.

When you create a project, you can optionally supply an HTTPS or SSH Git URL that points to a remote repository. The new project is a clone of that remote repository. You can commit, push and pull your code by running a console and opening a Terminal. Note that if you want to use SSH to clone the repo, you will need to first add your personal Cloudera Machine Learning SSH key to your GitHub account. For instructions, see *Adding SSH Key to GitHub*.

If you see Git commands hanging indefinitely, check with your cluster administrators to make sure that the SSH ports on the Cloudera Machine Learning hosts are not blocked.

Related Information

[Adding an SSH Key to GitHub](#)

[Creating a Project](#)

Linking an Existing Project to a Git Remote

If you did not create your project from a Git repository, you can link an existing project to a Git remote (for example, `git@github.com:username/repo.git`) so that you can push and pull your code.

Procedure

1. Launch a new session.
2. Open a terminal.
3. Enter the following commands:

Shell

```
git init
git add *
git commit -a -m 'Initial commit'
git remote add origin git@github.com:username/repo.git
```

You can run `git status` after `git init` to make sure your `.gitignore` includes a folder for libraries and other non-code artifacts.

Web Applications Embedded in Sessions

This topic describes how Cloudera Machine Learning allows you to embed web applications for frameworks such as Spark 2, TensorFlow, Shiny, and so on within sessions and jobs.

Many data science libraries and processing frameworks include user interfaces to help track progress of your jobs and break down workflows. These are instrumental in debugging and using the platforms themselves. For example, TensorFlow visualizations can be run on TensorBoard. Other web application frameworks such as Shiny and Flask are popular ways for data scientists to display additional interactive analysis in the languages they already know.

Cloudera Machine Learning allows you to access these web UIs directly from sessions and jobs. This feature is particularly helpful when you want to monitor and track progress for batch jobs. Even though jobs don't give you access to the interactive workbench console, you can still track long running jobs through the UI. However, note that the UI is only active so long as the job or session is active. If your session times out after 60 minutes (default timeout value), so will the UI.



Important: If you want to share your web application as a long-running standalone application that other business users can access, Cloudera recommends you now use the feature to support long-running web applications on ML workspaces.

If you are only running a server-backed visualization as part of your own analysis, then you can continue to keep embedding web applications in sessions as described in this topic. Note that running web applications in sessions is also the recommended way to develop, test, and debug analytical apps before deployment.

HTTP services running in containers that bind to `CDSW_APP_PORT`, `CDSW_READONLY_PORT`, and `CDSW_PUBLIC_PORT` are available in browsers at the following urls:

- CDSW_APP_PORT is available at

```
https://<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>
```

- CDSW_READONLY_PORT is available at

```
https://read-only-<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>
```

- CDSW_PUBLIC_PORT is available at

```
https://public-<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>
```

Therefore, TensorBoard, Shiny, Flask or any other web framework accompanying a project can be accessed directly from within a session or job, as long as it is run on CDSW_APP_PORT or CDSW_READONLY_PORT.

CDSW_APP_PORT is meant for applications that grant some level of control to the project, such as access to the active session or terminal. CDSW_READONLY_PORT must be used for applications that grant read-only access to project results.

To access the UI while you are in an active session, click the grid icon in the upper right hand corner of the Cloudera Machine Learning web application, and select the UI from the dropdown. For a job, navigate to the job overview page and click the History tab. Click on a job run to open the session output for the job. You can now click the grid icon in the upper right hand corner of the Cloudera Machine Learning web application to access the UI for this session.

Limitations with port availability

Cloudera Machine Learning exposes only one port per-access level. This means, in version 1.6.0, you can run a maximum of 3 web applications simultaneously:

- one on CDSW_APP_PORT, which can be used for applications that grant some level of control over the project to Contributors and Admins,
- one on CDSW_READONLY_PORT, which can be used for applications that only need to give read-only access to project collaborators,
- and, one on the now-deprecated CDSW_PUBLIC_PORT, which is accessible by all users.

However, by default the editors feature runs third-party browser-based editors on CDSW_APP_PORT. Therefore, for projects that are already using browser-based third-party editors, you are left with only 2 other ports to run applications on: CDSW_READONLY_PORT and CDSW_PUBLIC_PORT. Keep in mind the level of access you want to grant users when you are selecting one of these ports for a web application.

Example: A Shiny Application

This example demonstrates how to create and run a Shiny application and view the associated UI while in an active session.

Create a new, blank project and run an R console. Create the files, ui.R and server.R, in the project, and copy the contents of the following example files provided by [Shiny by RStudio](#):

R

```
# ui.R

library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Hello Shiny!"),
```

```
# Sidebar with a slider input for the number of bins
sidebarLayout(
  sidebarPanel(
    sliderInput("bins",
               "Number of bins:",
               min = 1,
               max = 50,
               value = 30)
  ),

  # Show a plot of the generated distribution
  mainPanel(
    plotOutput("distPlot")
  )
)
))
```

R

```
# server.R
library(shiny)
# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  # 1) It is "reactive" and therefore should re-execute automatically
  #    when inputs change
  # 2) Its output type is a plot

  output$distPlot <- renderPlot({
    x <- faithful[, 2] # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

Run the following code in the interactive workbench prompt to install the Shiny package, load the library into the engine, and run the Shiny application.

R

```
install.packages('shiny')

library('shiny')

runApp(port=as.numeric(Sys.getenv("CDSW_READONLY_PORT")), host="127.0.0.1",
       launch.browser="FALSE")
```

Finally, to access the web application, either:

- Click the grid icon in the upper right hand corner of the Cloudera Data Science Workbench web application, and select the Shiny UI, Hello Shiny!, from the dropdown.
- Access the web application directly by visiting the URL: [https://public-\[session-id\].\[CML host\]/](https://public-[session-id].[CML host]/)

The UI will be active as long as the session is still running.