Cloudera AI

Managing Projects in Cloudera AI

Date published: 2020-07-16
Date modified: 2025-10-31

CLOUDΞRA

Legal Notice

Contents

Projects in Cloudera AI

Projects form the heart of Cloudera AI. 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.
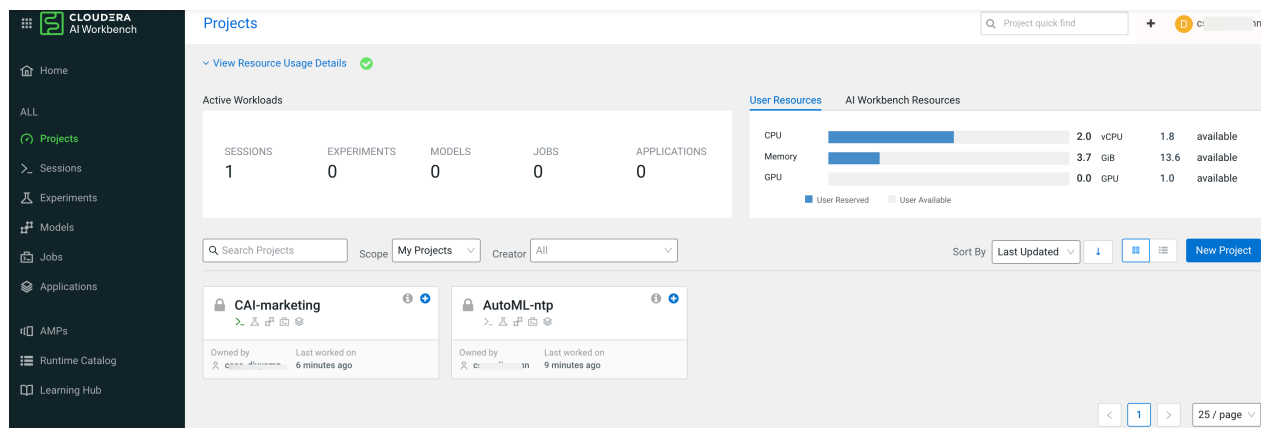
- Resource Usage Details - A collapsible section that displays resource usage.
  - Active Workloads - If there are active workloads running, this section describes the number of Sessions, Experiments, Models, Jobs, and Applications that are running.
  - User Resources and Workbench Resources
    - Click 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 do not have a quota, the maximum usage will be what is available on the cluster. If you have a GPU, you will also see the GPU usage.
    - Click on the Workbench 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 Cloudera AI Workbench.
- 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 AI.

Creating a Project with ML Runtimes 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 ML Runtimes variants in Cloudera AI.

For Cloudera AI UI

1. Go to Cloudera AI and on the left sidebar, click Projects.
2. Click New Project.
3. If you are a member of a team, select the Account under which you want to create this project from the drop-down menu. If there is only one account on the deployment, you do 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 AI 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.

   - Blank - The project will contain no information from a template, local file, or Git.
   - Templates - Template projects contain example code that can help you get started with Cloudera AI. They are available in R, Python, PySpark, and Scala. Using a template project is not required, but it helps to start using Cloudera AI right away.
   - Local - If you have an existing project on your local disk, use this option to upload compressed files or folders to Cloudera AI.
   - Git - If you already use Git for version control and collaboration, you can continue to do so with Cloudera AI. Specifying a Git URL clones the project into Cloudera AI. To use a password-protected Git repository, see *Creating a project from a password-protected Git repo*.

7. If you would like to configure which Runtimes are available for this particular project, complete the following:

   - Scroll down to the Runtimes section and enter the appropriate information:

     - A few recommended Runtimes are configured by default.
     - Use the **Advanced** view to add ML Runtimes based on a more detailed Editor, Kernel, Edition, and Version criteria.
     - Runtimes with the Enabled status and highest maintenance version will be configured as default settings for project creation as follows:

       - Upon new installation, runtimes will be calculated from variants recommended by Cloudera.
       - In cases other than new installations, variants that are set as default on the Runtime Catalog.

8. 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 AI also creates the following .gitignore file.

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

9. Set or verify the ML Runtimes settings for the project.

   a. In the left navigation bar, select Project Settings.
   b. Select the Runtime tab.

   Note: Note that site administrators now have the ability to select ML Runtimes as a default for newly created projects. This setting can be found within the Administration menu below the Runtime tab.

For Cloudera AI APIv2

To create a project with ML Runtimes, follow this example:

```
project_body = cmlapi.CreateProjectRequest(
      name = "project_name",
      description = "project_description",
      default_project_engine_type = "ml_runtime",
      project_body.visibility = "public", # or "private" or "organization"
```

```
        template = "Python")
```

You also need to specify a runtime_identifier if this is used with an ml_runtime project. Obtain a list of runtimes with the following command:

```
client.list_runtimes()
```

For some more examples of commands related to projects, see: *Using the Projects API.*

Related Information

Using the Projects API

Creating a project from a password-protected Git repo

Creating a project from a password-protected Git repo

You can create projects in Cloudera AI 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 Cloudera AI 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/som euser/somerepo.git

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

Configuring Project-level Runtimes

If you 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 selecting 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 chosing 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 selecting the x" in the right most column.
- The Available Runtimes table shows related counters for non-interactive workloads.

Adding project collaborators

Learn how you can add collaborators to a project.

About this task

For projects created under your personal account, you can add anyone from your organization as a collaborator. Howevere, for projects created under a team account, you can only add collaborators who are already members of that team. If your project requires collaborators from multiple teams, create a new team with the necessary members, and then create the project under that team account. Additionally, if your project was created from a Git repository, each collaborator must create the project using the same central Git repository.

You can assign one of the following access levels to project collaborators:

- Viewer - The Viewer has read-only access to code, data, and results.
- Operator - The Operator has 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 - The Contributor can view, edit, create, and delete files and environmental variables, run sessions, experiments, jobs or models and run code in running jobs. Additionally, Contributors can set the default engine for the project.
- Administrator - The Administrator 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

Administrators, owners, contributors or operators can add a team as a collaborator to a project. Teams can be added to both team projects or to individual projects. When a team is added, two permission modes are available:

- Inherit mode - If team members are already part of the project, their permissions are upgraded to match the permissions they have within the team.
- Override mode - The permissions assigned to the team are uniformly applied to all team members, regardless of any individual permissions they may already have on the project.

Before you begin

Collaborating securely on projects

Before adding project collaborators, it is important to note that assigning the Contributor or the Administrator role to a project collaborator effectively grants them write access to your data in Cloudera. This is because project contributors and project administrators have write access to all project code, including any library code that you might not be actively reviewing. For example, a Contributor or Administrator can modify project file(s) to include code that deletes data on the Cloudera cluster. If you later launch a session and run the modified code, it may appear as though you were the one who deleted the data.

Additionally, project collaborators have access to all active sessions and jobs. This means a malicious user can potentially impersonate you by accessing one of your active sessions. Therefore, it is crucial to limit project access to trusted collaborators only. To mitigate this risk, Site Administrators can enforce restrictions by allowing only session creators to execute commands within their own active sessions.

For these reasons, Cloudera recommends using Git for secure collaboration on shared projects. Git not only enhances security, but also helps prevent file modification conflicts, especially when your team is working on complex projects.

Procedure

1. In the Cloudera console, click the Cloudera AI tile.

   The Home page displays.
2. Select the required Workbench.

   The Cloudera AI Workbench page displays.

3. Click Projects in the left navigation pane and select the required project.

4. Go to Collaborators, and enter the user ID in the Search box.

5. Choose the User ID, and click Add. The user or team is added with their role displayed.

Modifying Project settings

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

About this task
Users can view the Spark configurations set at the workbench-level for the Project.

Procedure

1. Switch 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.

   You can see the following configuration tab options in the Project Settings dashboard:

   General

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

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

   ML Runtimes

   > Available Runtimes: You can add, remove and configure the ML Runtimes on this tab. When starting a Cloudera AI session, job, application, or model the ML Runtime variant to be used can

be chosen from this list.Cloudera AI ensures that your code is always run with the specific ML Runtime version you selected.

Spark Config: Users can view the default and custom Spark configurations set at the Cloudera AI Workbench level that will be applied for all Spark workloads running in this project.

Advanced

- **Environment Variables** - If there are any environmental variables that have to be injected into all the ML Runtimes running this project, you can add them to this page. For more details, see *ML Runtimes Environment Variables*.
- **Ephemeral Storage Settings** - Ephemeral storage space is a scratch space that a Cloudera AI 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 AI 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

Engine Environment Variables

SSH Keys

Data connection management

Ephemeral storage

Ephemeral storage

Managing Project Files

Cloudera AI 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

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 Project Settings.
3. Go to the Delete Project.
4. 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.

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

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

   New projects now default to using ML Runtimes. Legacy Engines are deprecated in the current release. However, you can change the default engine later in this task.

3. Check the settings for your session:

If your project is using ML Runtimes, you will see the following settings:

Editor

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

Kernel

Selects the Kernel, for example Python 3.7, R4.0.

Edition

Selects the Runtime Edition. Initially only Standard variants are supported.

Version

Selects the ML Runtimes version.

> **Note:** The selector options only consider the configurations supported by the actual deployments and certain selections will automatically limit others. For example, certain versions are only relevant for Python or certain editors are supported only with certain kernels.

If your project is using Legacy Engines, 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. If your project is using Legacy Engines, you can modify the engine image used by this session:

a) By Engine Image, click Configure.

Cloudera AI 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. Select the desired Resource Group from the dropdown menu. This choice defines the underlying instance type (for example, M5X large) provided by your administrator.

6. Based on the Resource Group you selected in the previous step, the list of available resource profiles will populate. Select the desired CPU and memory.

7. [Optional] Enable the Enable GPU toggle button to allocate the available GPU resources from the selected resource group. By selecting both the Resource Group and CPU, GPU, and memory, you ensure that your workload runs on the particular node type you require while consuming the specific amount of compute resources requested. The minimum configuration is 1vCPU and 2 GB memory.

8. 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 can be best used for codes you want to keep, while the command prompt is an optimal choice for quick interactive exploration.

Command Promt

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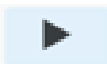
If you created your project from a template, you can see the project files in the editor. You can open a file in the editor by clicking the file name in the file navigation bar on the left.

Editor

To run code from the editor:

1. Select a script from the project files on the left sidebar.
2.

   To run the whole script click ▶ on the top navigation bar, or, highlight the code you want to run and press Ctrl+Enter (Windows/Linux) or cmd+Enter (macOS).

When working on an analysis, writing and executing your code from the editor rather than the command prompt enables to iteratively develop your code and save it along the way.

If you require more space for your editor, you can collapse the file list by double-clicking between the file list pane and the editor pane. You can hide the editor, using the editor's View menu.

Code Autocomplete

The Python and R kernels include support for automatic code completion, both in the editor and the command prompt. Use **single tab** to display suggestions and **double tab** for autocomplete.

Project Code Files

All project files are stored to persistent storage within the respective project directory at /var/lib/cdsw/current/projects. They can be accessed within the project in a typical directory structure. For example, you can import functions from one file to another within the same project.

Access the Terminal

Cloudera AI provides full terminal access to running engines from the web console. This topic show you how to access the Terminal from a running Workbench session.

You can use the terminal to move files around, run Git commands, access the YARN and Hadoop CLIs, or install libraries that cannot be installed directly from the engine. To access the Terminal from a running session, click Terminal Access above the session log pane.

```
Kernel: python2

Project workspace: /home/cdsw

Kerberos principal: ▮▮ ▮▮▮▮ ▮▮▮ ▮ ▮▮▮

Runtimes:
  R: R version 3.4.1 (--) -- "Single Candle"
  Python 2: Python 2.7.11
  Python 3: Python 3.6.1
  Java: java version "1.8.0_144"

cdsw@frczgqdokx4k67un:~$ ▯
```

The terminal's default working directory is /home/cdsw, which is where all your project files are stored. Any modifications you make to this folder will persist across runs, while modifications to other folders are discarded.

If you are using Kerberos authentication, you can run klist to see your Kerberos principal. If you run hdfs dfs -ls you will see the files stored in your HDFS home directory.

Note that the terminal does not provide root or sudo access to the container. To install packages that require root access, see *Customized Engine Images*.

Related Information
Customized engine images

Stop a Session

This topic demonstrates how to stop a session to free up resources for other users when you are finished.

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 AI editor, you can configure Cloudera AI 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 Cloudera AI sessions. If the Cloudera AI 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 Cloudera AI 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 Cloudera AI.

When you bring your own editor, you still get many of the benefits Cloudera AI 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 AI
- Reuse code in other Cloudera AI 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 AI deployment, not the local machine. Browser IDEs run within a Cloudera AI 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 AI 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 AI web UI and cannot be completed through an editor.

Modes of configuring third-party Editors

The configuration for an Integrated Development Environment (IDE) depends on which type of editor you want to use.

In addition to the native Cloudera AI editor, you can configure Cloudera AI 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 AI. 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 AI 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 AI 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 workbench 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/cdsw/.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 AI 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 AI 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 AI. 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 AI. 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-based IDE as an Editor

Configuring a local IDE using an SSH gateway

Configure a browser-based IDE as an Editor

When you use a browser-based Integrated Development Environment (IDE), changes that you make in the editor are propagated to the Cloudera AI 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 AI. When you run the code from the IDE, execution is pushed from the IDE to Cloudera AI.

Base Engine Image v8 and higher for Cloudera AI 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*.

Testing a browser-based IDE in a Session before installation

This process can be used to ensure that a browser-based Integrated Development Environment (IDE) works as expected before you install it to a project or to a customized engine image. This process is not meant for browser-based 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 AI web UI.

2. Open the Cloudera AI 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 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.

Configuring a browser-based 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 AI.

Before you begin

Before you start, verify that you have installed the browser-based Integrated Development Environment (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

1. Open the Cloudera AI web UI.

2. Go to the project you want to configure an editor for.

3. Go to Settings Editors and click New Editor.

4. 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 AI 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.

5. 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 AI web UI.

Related Information
Installing Additional Packages

Configuring a local IDE using an SSH gateway

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

Cloudera AI 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 AI deployment.

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

1. Establish an SSH endpoint with the Cloudera AI CLI client. See Initialize an SSH Endpoint.
2. Configure the local IDE to use Cloudera AI as the remote interpreter.
3. Optionally, sync files with tools (like mutagen, SSHFS, or the functionality built into your IDE) from Cloudera AI 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 AI.
5. Sync the files you edited locally to Cloudera AI.
6. Use the Cloudera AI 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 AI 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 AI to act as a remote SSH interpreter for PyCharm. Once finished, you can use PyCharm to edit and sync the changes to Cloudera AI. To perform actions such as deploying a model, use the Cloudera AIg 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 AI project. Alternatively, create a new project you have access to.

Add Cloudera AI as an Interpreter for PyCharm

In PyCharm, you can configure an SSH interpreter. Cloudera AI 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 AI 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 AI 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 prviate 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 AI 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 AI 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 AI 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 AI 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 AI, 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 AI 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 AI 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

Follow the guidelines for configuring VS Code as a local Integrated Development Environment (IDE).

About this task
Cloudera AI 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 AI to act as a remote SSH interpreter for VS Code. Once finished, you can use VS Code to edit and syncronise the changes to Cloudera AI. To perform actions such as deploying a model, use the Cloudera AI 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 AI 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 AI web UI and go to User Settings Remote Editing for your user account.

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

For more information on logging in with `cdswctl`, see *Log into cdswctl*.

Related Information

Log into cdswctl

Initialize an SSH connection to Cloudera AI for VS code

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

Procedure

1. Log in to Cloudera AI with the CLI client using the `cdswctl login` command.

```
cdswctl login -n [***USERNAME***] -u [***http(s)://your_cloudera_ai_work
bench_url***]
```

- Replace [***USERNAME***] with your Cloudera AI username.
- Replace http(s)://your_cloudera_ai_workbench_url with the actual URL of your Cloudera AI Workbench deployment.

For example, to log in the user sample_user to the deployment located at https://ml-dd7ab8-e7b.exu2-8y8x.cdp. cldr.com/home, you would use

```
cdswctl login -n SAMPLE_USER -u https://ml-dd7ab8-e7b.exu2-8y8x.cdp.cldr
.com/home
```

If your workspace uses Single Sign-On (SSO), then add your API key using the `-y` flag. For example:

```
cdswctl login -n SAMPLE_USER -y api_key -u https://ml-dd7ab8-e7b.exu2-8y
8x.cdp.cldr.com/home
```

> **Note:** The `-y` flag and API key are only required for SSO-enabled workspaces. If your workspace is not SSO-enabled, omit the `-y` flag. For creating an API key, see Generating an API key.

2. Create a local SSH endpoint to Cloudera AI.

Run the following command:

```
cdswctl ssh-endpoint -p [***USERNAME***]/[***PROJECT NAME***] [-c [***CP
U 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 [***PORT***] cdsw@localhost
...
```

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

```
ssh -p [***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 AI web UI.

The public key could be rejected when the new ssh key pair is generated with a special name such as id_rsa_syste st. 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 AI 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 AI uses this method to connect to VS Code and act as its interpreter.

Before you begin

Ensure that you have installed the following VS Code extensions:

- Remote SSH extension

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

About this task

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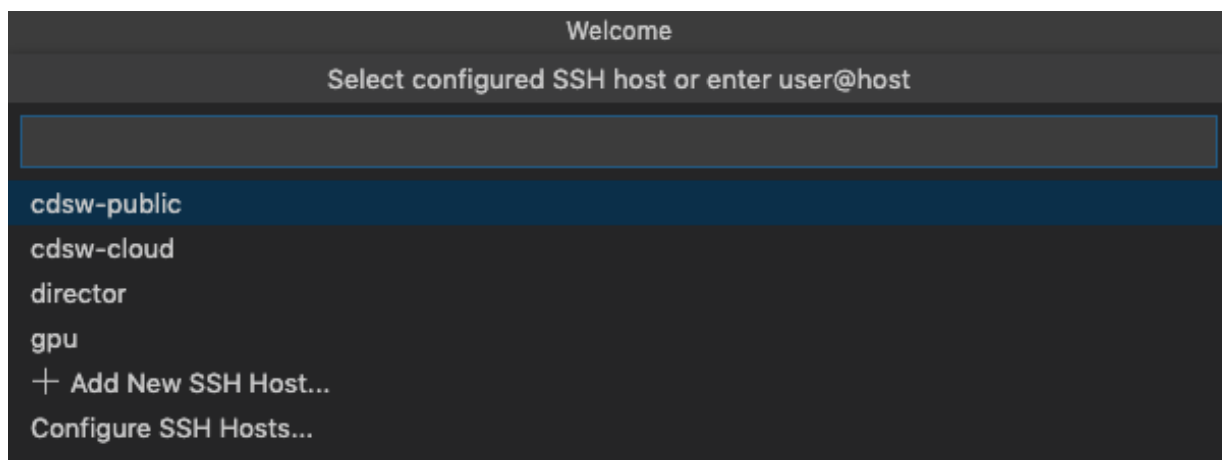
Procedure

1. Verify that the SSH endpoint for Cloudera AI 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.
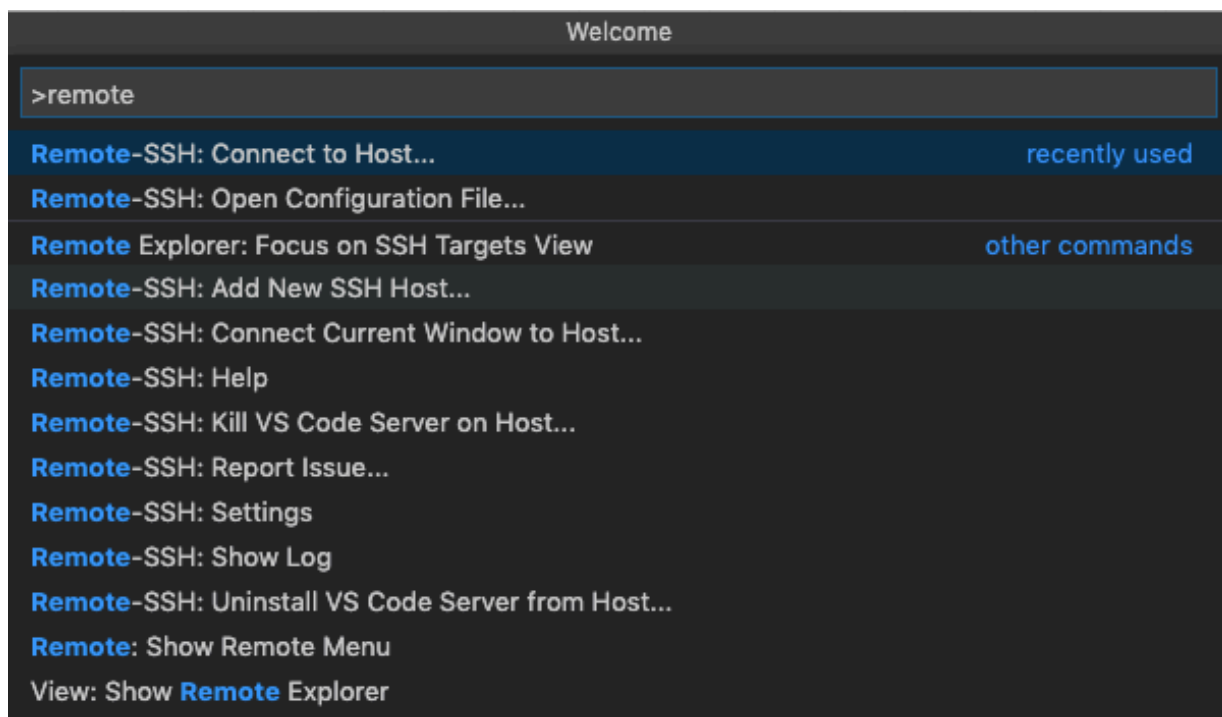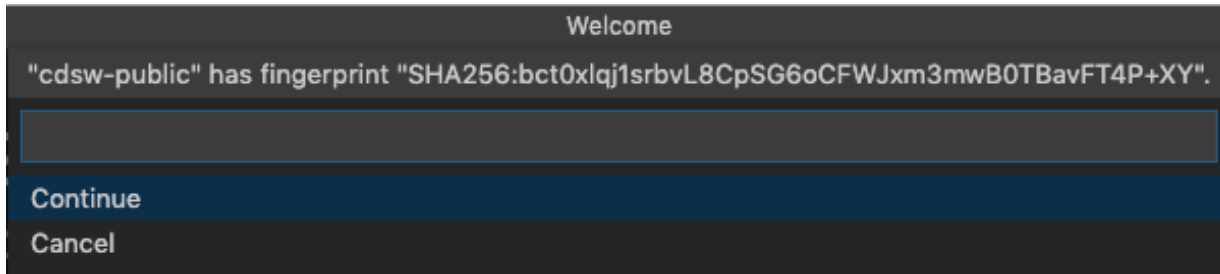

4. Connect to the host you added previously.

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 is 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 AI 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 seconds or so, the issue is with the lock file that VS Code creates during the install. Close VS Code and in Cloudera AI 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.
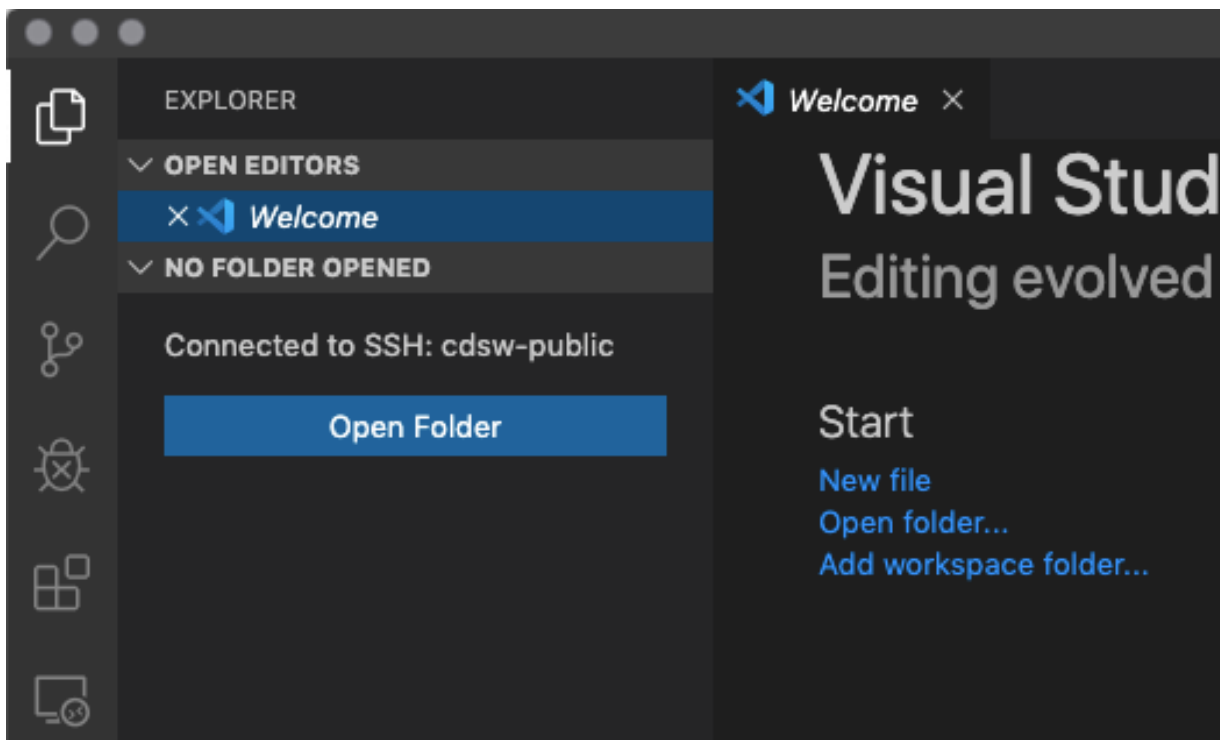
7. From the Explorer view, you can edit any of the files on your Cloudera AI server.
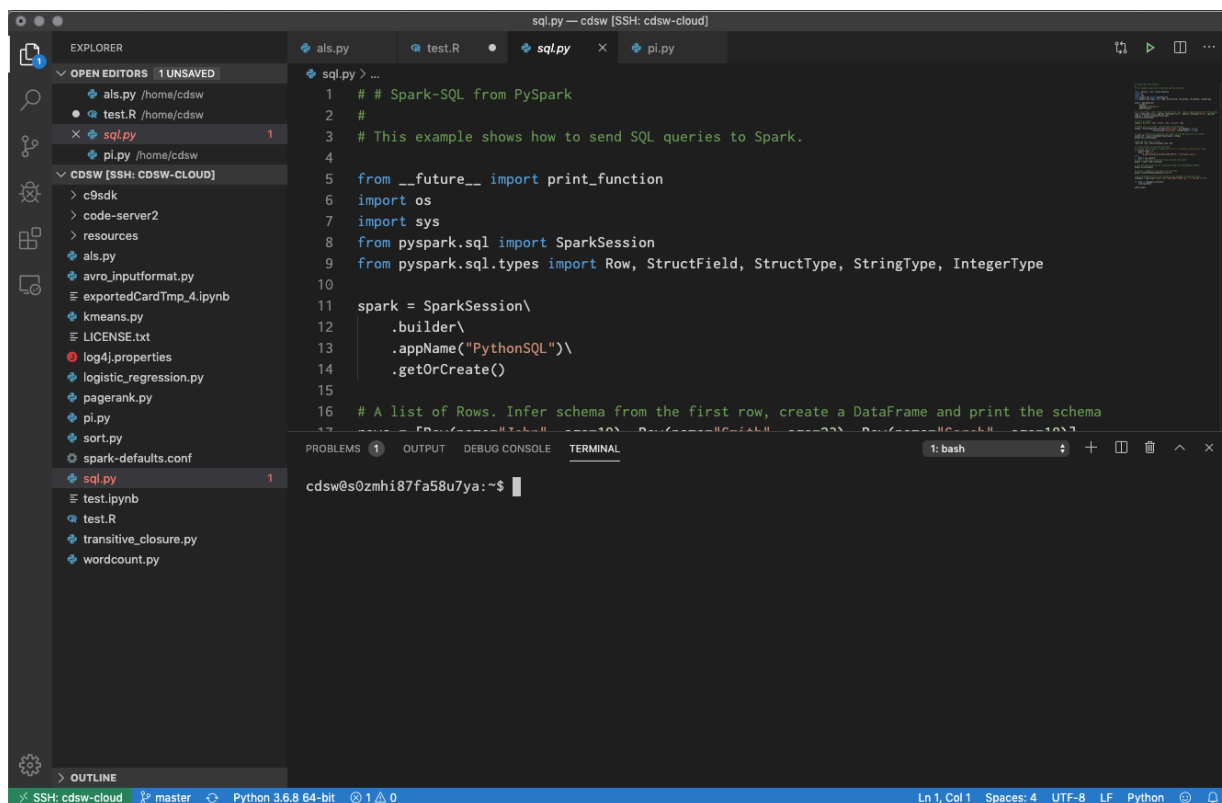
Using the Explorer view, you remotely edit and modify your Cloudera AI files. VS Code also supports Python and R, some powerful coding tools, that you can take advantage of over the remote connection.
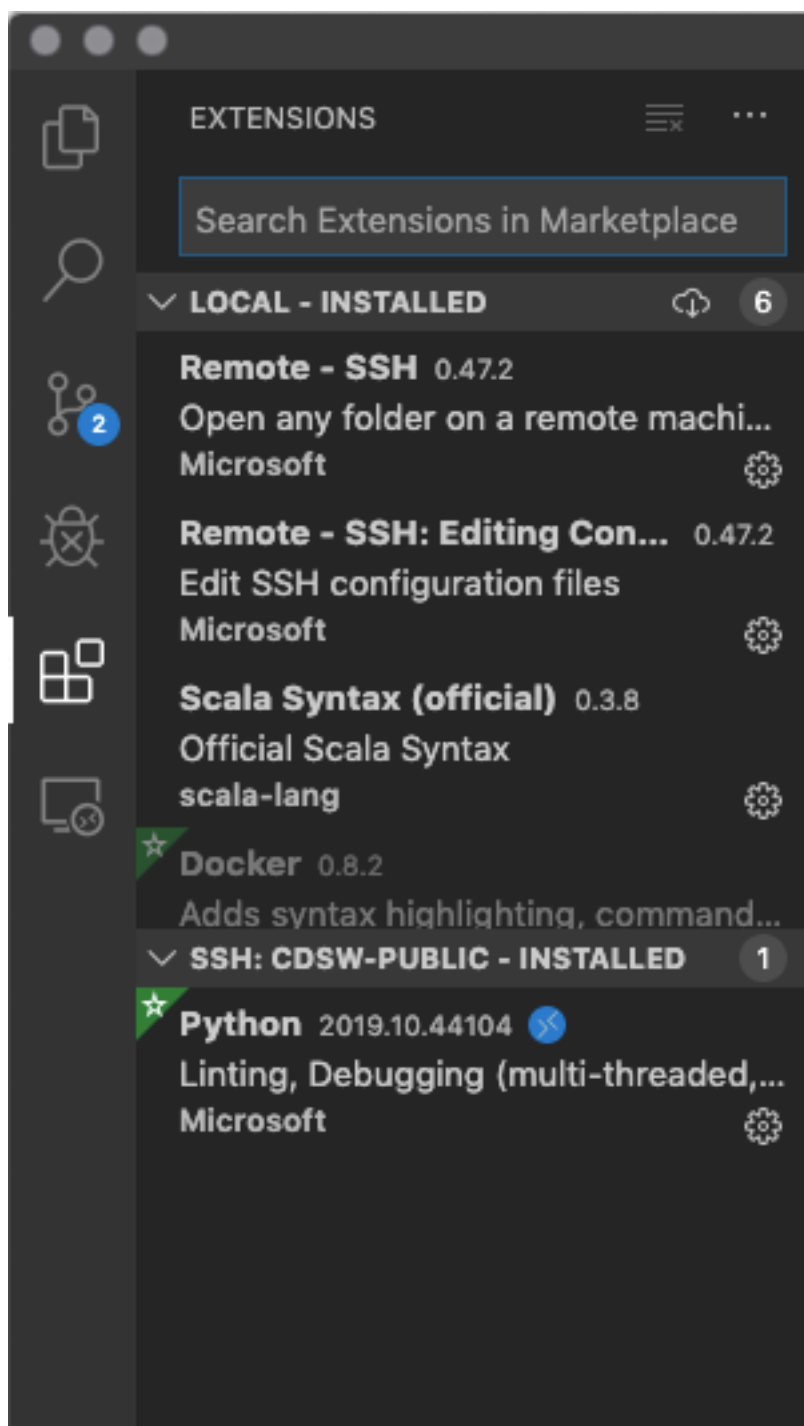
(Optional) Using VS Code with Python

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.
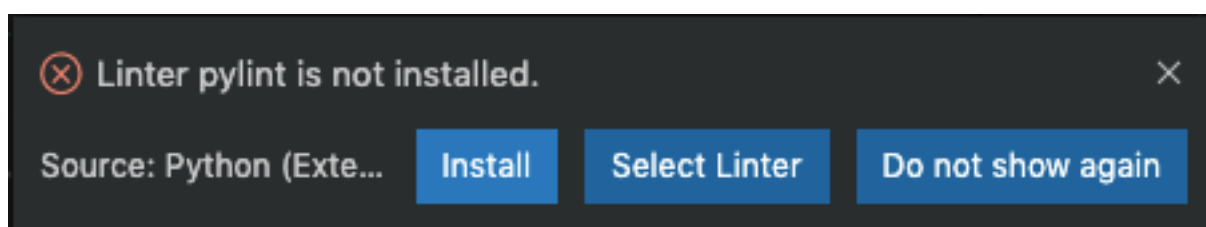
3. Click Install.

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 AI. It is 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, run the selected code. Since this is a remote session, you can run pyspark directly inside VS Code.
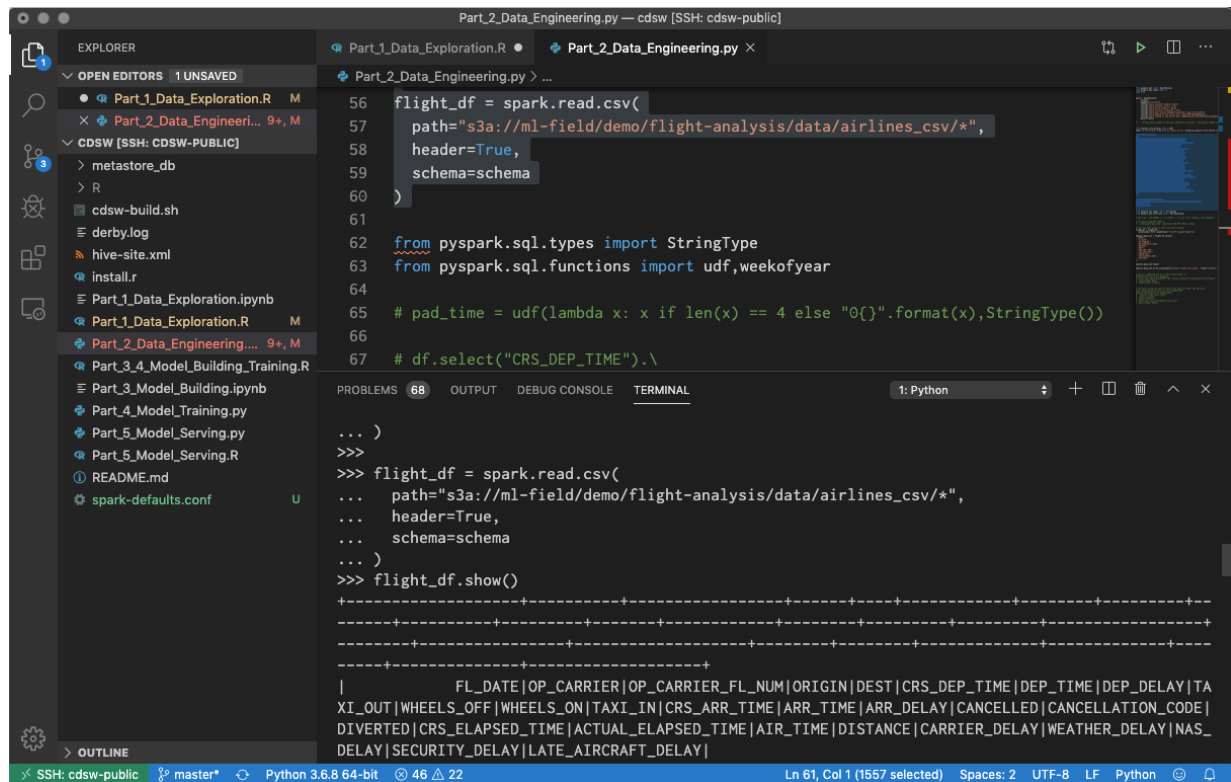
For more complex code requirements, you can also use the Python Debugging feature in VS Code.

(Optional) Using VS Code with R
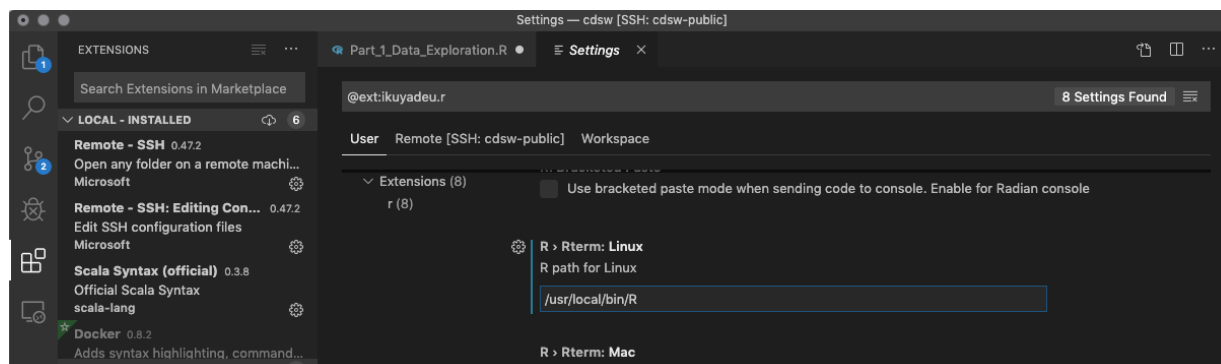
You can use VS Code with R.

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.
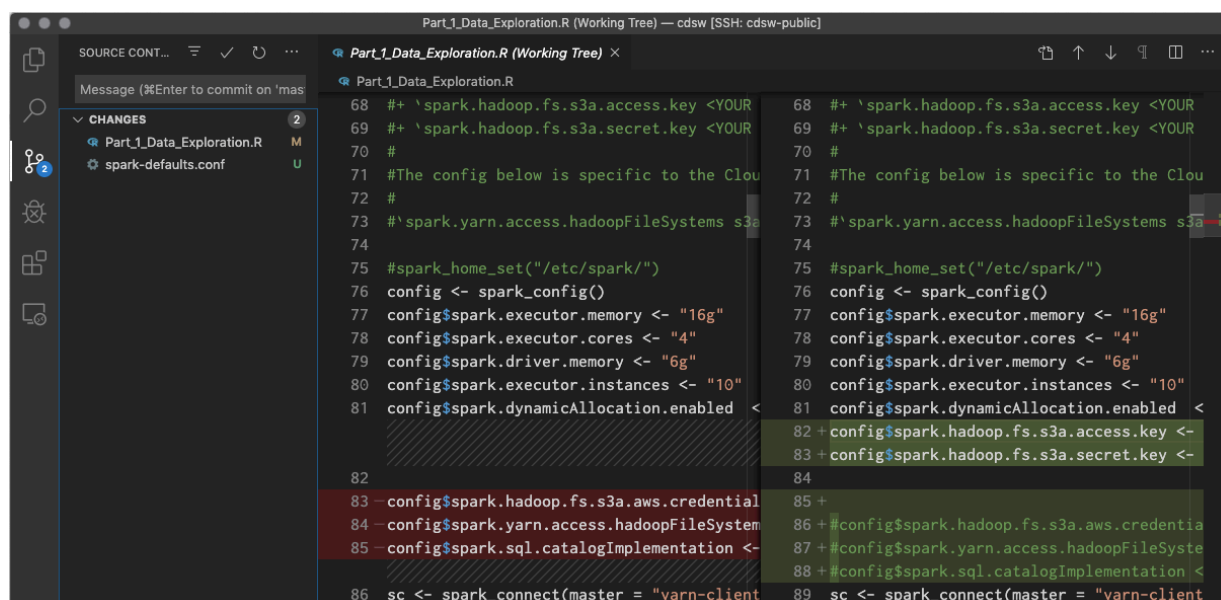
Procedure

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 / Cloudera AI server and can be viewed online using Jupyter Notebook as a browser based editor.

Procedure

1. Because of the way Cloudera AI 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://1 27.0.0.1:8888/?token=[some-token].
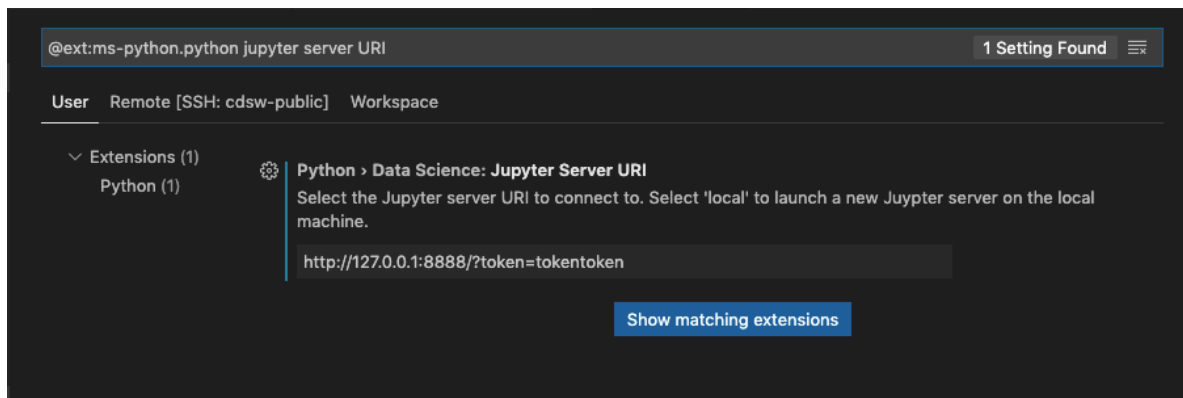
    b) Then you need to open a terminal to launch a Jupiter Notebook server.

       You can launch it using: /usr/local/bin/jupyter-notebook —no-browser   —ip=127.0.0.1 —NotebookApp.t oken=[some-token] —NotebookApp.allow_remote_access=True   .
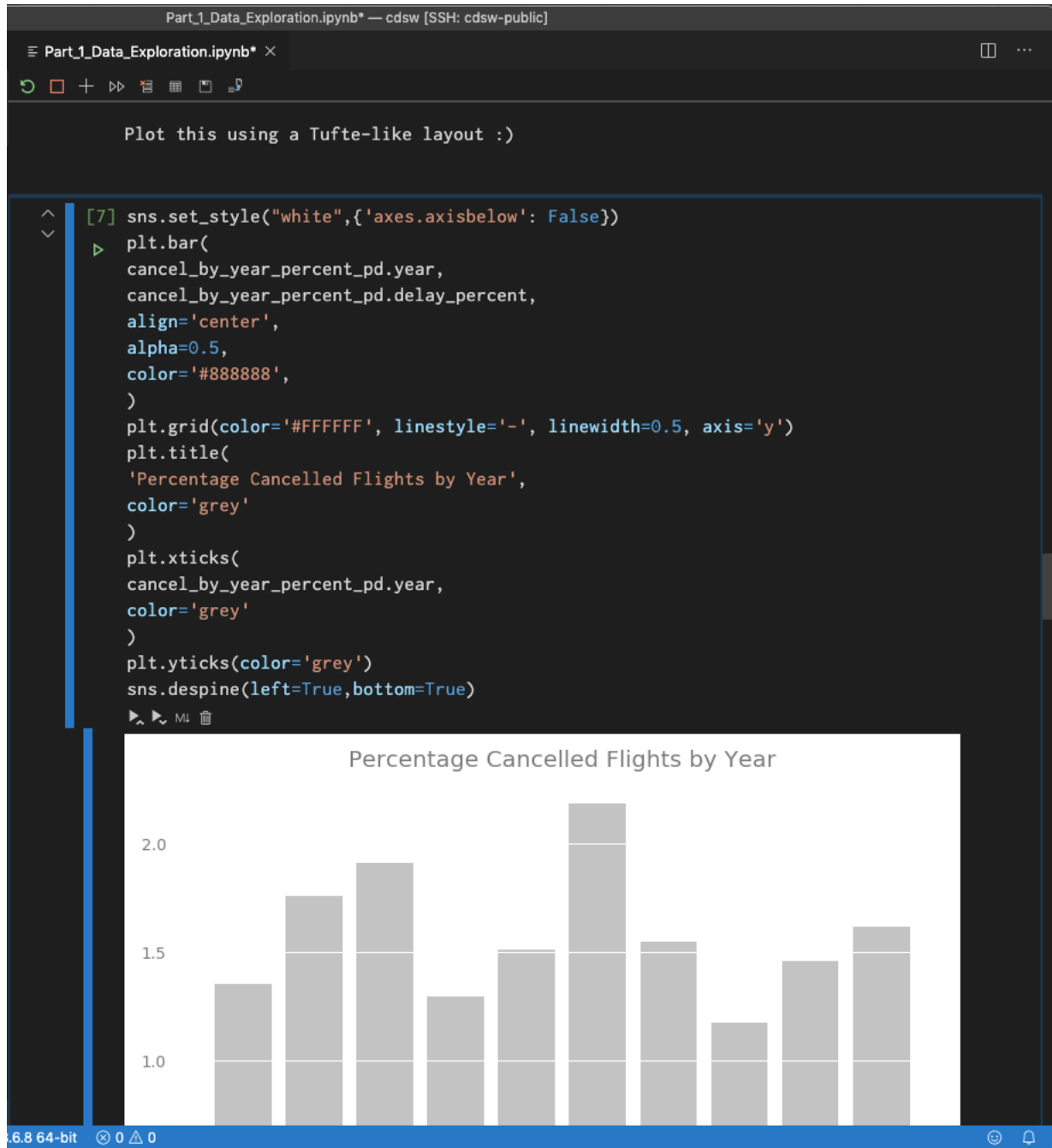
   This creates a Jupyter server that any new Notebooks you launch will run in.

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

3. 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.
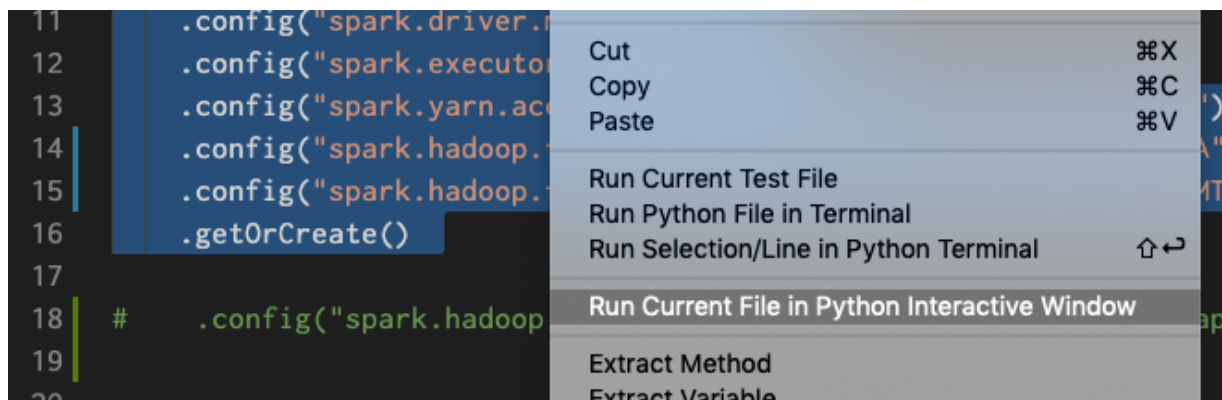
(Optional) Using VS Code with Git integration

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.
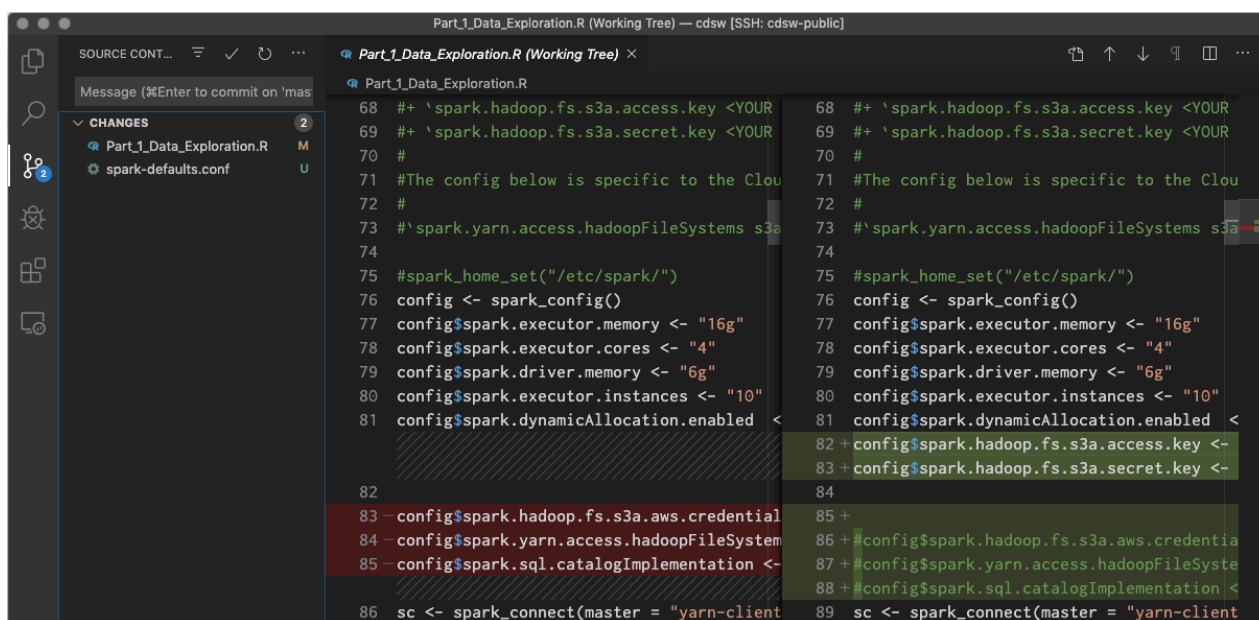
Limiting files in Explorer view

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 AI 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 AI.

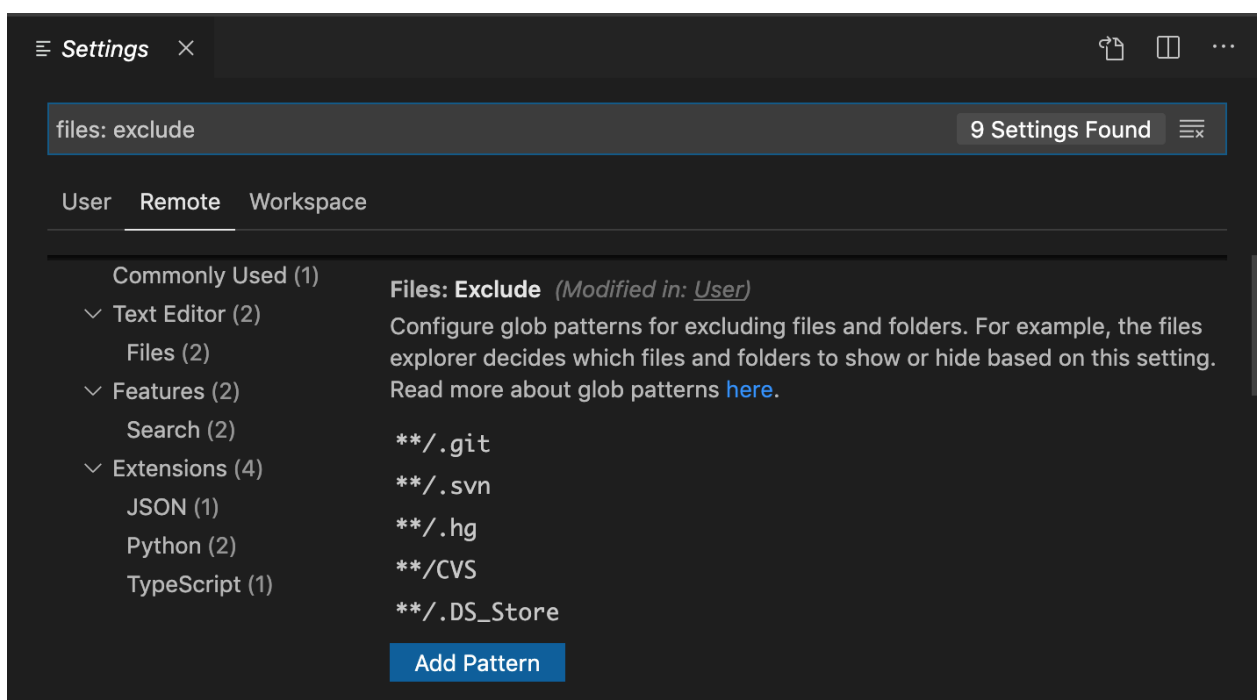Teams that already use Git for collaboration can continue to do so. Each team member will need to create a separate Cloudera AI project from the central Git repository. For anything but simple projects, Cloudera recommends using Git for version control. You should work on Cloudera AI the same way you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

Cloudera AI 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 AI 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 AI 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 AI 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 AI 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 Applications feature to support long-running web applications on Cloudera AI Workbenches.
>
> 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 AI 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 AI web application to access the UI for this session.

Limitations with port availability

Cloudera AI 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]/

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

Example: Flask application

This example demonstrates how to create and use a Flask application.

Consider the following code example for using a Flask application:

```
from flask import Flask
from werkzeug import serving
import re, os

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
@app.route("/foo2")
def hello_world2():
    return "<p>Hello, World!</p>"

@app.route('/fit_update', methods=["POST"])
```

```
def fit_update():
    return "<p>Fit Update: " + str(request.get_json()) + "</p>"

if __name__ == '__main__':
    PORT = os.getenv('CDSW_READONLY_PORT', '8090')
    app.run(host="127.0.0.1", port=int(PORT))
```

When creating or modifying an application, you can specify a new value for the CDSW_APP_POLLING_END
POINT environmental variable. Update the CDSW_APP_POLLING_ENDPOINT in the Project Settings and add the
following to your code:

Figure 1: Setting CDSW_APP_POLLING_ENDPOINT in Project Settings

Project Settings

General Runtime **Advanced** SSH Tunnels Delete Project

Environment Variables

Set project environment variables that can be accessed from your scripts.

Environment variable **values** are only visible to collaborators with **write** or higher access. They are a great way to securely store confidential information such as your AWS or database credentials. Names are available to all users with access to project.

| CDSW_APP_POLLING_ENDPOINT | healthcheck |
| PROJECT_OWNER | ..... |

```
parent_log_request = serving.WSGIRequestHandler.log_request
def log_request(self, *args, **kwargs):
    if self.path == '/healthcheck':
        return
    parent_log_request(self, *args, **kwargs)

def filter_healthcheck_logs():
    serving.WSGIRequestHandler.log_request = log_request

filter_healthcheck_logs()
```