Cloudera Data Science Workbench

CLI Reference

Date published: 2020-02-28
Date modified:

https://docs.cloudera.com/

Legal Notice

Contents

Cloudera Data Science Workbench Command Line Reference..... 4
Additional Usage Notes.....5
cdswctl Command Line Interface Client..... 6
Download and Configure the cdswctl.....6
(Optional) Generate an SSH Public/Private Key.....6
Download cdswctl and Add an SSH Key.....7
Initialize an SSH Connection to Cloudera Data Science Workbench for Pycharm.....7
Log in to cdswctl.....9
Prepare to manage models using the model CLI.....9
Create a model using the CLI.....10
View replica logs for a model using the CLI.....11
Using ML Runtimes with cdswctl..... 11
Querying the engine type..... 11
Listing runtimes..... 11
Starting sessions and creating SSH endpoints..... 13
Creating a model.....13

Cloudera Data Science Workbench Command Line Reference

This topic describes the commands available to the Cloudera Data Science Workbench command line utility cdsw that exists within a Cloudera Data Science Workbench deployment. This utility is meant to manage your Cloudera Data Science Workbench cluster. Running cdsw without any arguments will print a brief description of each command.

In addition, there is a cdswctl CLI client that offers different functionality that is meant for use by data scientists to manage their sessions. For information about that see Cloudera Data Science Workbench Command Line Reference

Start, Stop, Restart for CSD Deployments: The commands available for a CSD-based deployment are only a subset of those available for an RPM deployment. For example, the CLI for CSD deployments does not have commands such as cdsw start, stop, and restart available. Instead, these actions must be executed through the Cloudera Data Science Workbench service in Cloudera Manager. For instructions, see Starting, Stopping, and Restarting the Service.

Important: On Cloudera Data Science Workbench 1.4.0 (and lower), do not stop or restart Cloudera Data Science Workbench without using the cdsw_protect_stop_restart.sh script. This is to help avoid the data loss issue detailed in TSB-346 .

All of the following commands can be used in an RPM-based deployment. Those available for CSD-based deployments have been marked in the table.

Command	CS	Description and Usage
cdsw start		Initializes and bootstraps the master host. Use this command to start Cloudera Data Science Workbench.
cdsw stop		De-registers, resets, and stops a host. On a worker host, this command will remove the worker from the cluster. On the master host, this command will bring down the application and effectively tear down the Cloudera Data Workbench deployment.
cdsw restart		Run on the master host to restart application components. To restart a worker host, use cdsw stop, followed by cdsw join. These commands have been explained further in this topic.
cdsw join		Initializes a worker host. After a worker host has been added, run this command on the worker host to add it to the Cloudera Data Science Workbench cluster. This registers the worker hosts with the master, and increases the available pool of resources for workloads.
cdsw status	■	Displays the current status of the application. Starting with version 1.4, you can use cdsw status -v or cdsw status --verbose for more detailed output. The cdsw status command is not supported on worker hosts.
cdsw validate	■	Performs diagnostic checks for common errors that might be preventing the application from running as expected. This command should typically be run as the first step to troubleshooting any problems with the application, as indicated by cdsw status.
cdsw logs	■	Creates a tarball with diagnostic information for your Cloudera Data Science Workbench installation. If you file a case with Cloudera Support, run this command on each host and attach the resulting bundle to the case.
cdsw version	■	Displays the version number and type of Cloudera Data Science Workbench deployment (RPM or CSD).
cdsw help	■	Displays the inline help options for the Cloudera Data Science Workbench CLI.
cdswctl login		Enables you to log into the cdswctl client.
cdswctl projects list		Lists the projects.

Command	CS	Description and Usage
cdswctl models create		Creates a model with the specified parameters.
cdswctl models list		Lists all models. You can refine the search by specifying the modelId.
cdswctl models listBuild		Llists the builds for a model. You can monitor the status of the build by specifying the modelId and the projectId.
cdswctl models listDeployments		List the deployments for a model. You can refine the search by specifying the modelId. Use the statusSet parameter to check the status of the model being deployed .
cdswctl models deploy		Deploys a model with the specified parameters.
cdswctl models listReplicas		Enables you to view the list of model replicas. You also need this information to obtain replica logs .
cdswctl models getReplicaLogs		Enables you to view the logs for a model replica.
cdswctl models restart		Restarts a model. Usage: cdswctl models restart --modelDeploymentId=<*deployment_ID*> Note: Running this command does not change the resources if you previously ran the cdswctl models update command.
`cdswctl models update`		Changes the name, description, or visibility of the model. To change a model's resources, use the `cdswctl models deploy` command.
`cdswctl models delete`		Deletes a model. Usage: cdswctl models delete --id=<*model_ID*>

Additional Usage Notes

This topic provides additional usage notes commands in a RPM-based deployments.

Note: These notes apply only to RPM-based deployments. In case of CSD-based deployments where you cannot directly modify cdsw.conf, Cloudera Manager will prompt you if the Cloudera Data Science Workbench service needs to be restarted.

Changes to cdsw.conf: Make sure cdsw.conf is consistent across all Cloudera Data Science Workbench hosts. Any changes made to the file must be copied over to all the other hosts.

- Master Host - Changes to the JAVA_HOME, MASTER_IP, DOCKER_BLOCK_DEVICES, and APPLICATION_BLOCK_DEVICE parameters in cdsw.conf require a re-initialization of the master host. [DSE-1228]

```
cdsw stop
```

```
cdsw start
```

Changes to other cdsw.conf parameters such as domain name changes, or TLS and HTTP proxy changes, require a restart of the application components.

```
cdsw restart
```

- Worker Host - Changes to cdsw.conf on a worker host, require a restart of the worker host as follows:

```
cdsw stop
cdsw join
```

cdswctl Command Line Interface Client

Cloudera Data Science Workbench 1.6 and later ships with a CLI client that you can download from the Cloudera Data Science Workbench web UI.

The cdswctl client can perform the following tasks:

- Logging in
- Creating an SSH endpoint
- Listing sessions that are starting or running
- Starting or stopping a session

Other actions, such as creating a project, require you to use the Cloudera Data Science Workbench web UI. For information about the available commands, run the following command:

```
cdswctl --help
```

Download and Configure the cdswctl

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

- You have an SSH public/private key pair for your local machine.
- You have Contributor permissions for an existing Cloudera Data Science project. Alternatively, create a new project you have access to.
- The Site Administrator has not disabled remote editing for Cloudera Data Science Workbench.

(Optional) Generate an SSH Public/Private Key

This task is optional. If you already have an SSH public/private key pair, skip this task. The steps to create an SSH public/private key pair differ based on your operating system.

About this task

The following instructions are meant to be an example and are written for macOS using ssh-keygen.

Procedure

1. Open Terminal.

2. Run the following command and complete the fields:

```
ssh-keygen -t rsa -f ~/.ssh/id_rsa
```

Keep the following guidelines in mind:

- Make sure that the SSH key you generate meets the requirements for the local IDE you want to use. For example, PyCharm requires the -m PEM option because PyCharm does not support modern (RFC 4716) OpenSSH keys.
- Provide a passphrase when you generate the key pair. Use this passphrase when prompted for the SSH key passphrase.
- Save the SSH key to the default ~/.ssh location.

Download cdswctl and Add an SSH Key

Provides steps to download cdswctl and add an SSH key.

Procedure

1. Open the Cloudera Data Science Workbench web UI and go to Settings > Remote Editing for your user account.
2. Download cdswctl client for your operating system.

   If you are using the macOS executable, cdswctl will be unsigned and therefore cannot be launched on the recent version of macOS without performing the following additional steps:

   a) In the Finder on your Mac locate the app you want to open.

      Don’t use Launchpad to do this. Launchpad doesn’t allow you to access the shortcut menu.

   b) Control-click the app icon, then choose Open from the shortcut menu.

   c) Click Open.

      The app is saved as an exception to your security settings, and you can open it in the future by double-clicking it just as you can any registered app.
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 AAAAB3NzaC1yc2EAAAADAQABAAACAQCha2J5mW3i3BgtZ25/FOsxywpLVkx1RgmZ
unI
```

5. In SSH public keys for session access, paste the key.

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

Initialize an SSH Connection to Cloudera Data Science Workbench for Pycharm

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

Procedure

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

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

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

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

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

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

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

   The command uses the following defaults for optional parameters:

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

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

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

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

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

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

   Information for the SSH endpoint appears in the output:

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

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

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

   For example:

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

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

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

4. Test the connection.

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

Log in to cdswctl

Provides steps to log in to cdswctl.

Procedure

1. Open the Model CLI client.
2. Run the following command while specifying the actual values for the variables:

   ```
   cdswctl login -u <workspace_url> -n <username> -y <legacy_api_key>
   ```

   where

   - workspace_url is the workspace URL including the protocol (http(s)://domain.com)
   - username is your user name on the workspace
   - api_key is the Legacy API key that you can obtain from the CDSW web UI. Go to Settings API Keys > and copy the Legacy API Key (and not the Model API Key).

   A Login succeeded message is displayed.

   To see more information about the login command parameters, run

   ```
   cdswctl login --help
   ```

Prepare to manage models using the model CLI

Before you can start using the model CLI to automate model deployment or to perform any other tasks, you must install the scikit-learn machine learning library for Python through the CDSW web UI.

About this task

You must perform this task through the CDSW web UI.

Procedure

1. Create a new project with Python through the web UI.

   Python provides sample files that you can use to create models using CLI.
2. To start a new session, go to the **Sessions** page from the left navigation panel and click New Session.

   The **Start A New Session** page is displayed.
3. On the **Start A New Session** page, select Python 3 from the Engine Kernel drop-down menu, and click Start Session.

   A new “Untitled Session” is created.
4. From the input prompt, install the scikit-learn machine learning library for Python by running the following command:

   ```
   !pip3 install sklearn
   ```
5. Open the fit.py file available within your project from the left navigation panel.

   You can use the fit.py file to create a fitted model which creates a model.pkl file that you can use to deploy the actual model.

6. Run the fit.py file by clicking Run Run all .
   The model.pkl directory is created that you can see within your project on the left navigation pane.
7. Close the session by clicking Stop.

Create a model using the CLI

Provides steps to create a model using the CLI.

Procedure

1. Open a terminal window and log into cdswctl.
2. Obtain the engine image ID as described in the following steps:
   a) Run the following command:

      ```
      cdswctl projects list
      ```

      The project ID, your username, and the project name are displayed. For example:
      1: john-smith/petal-length-predictor
   b) Note the project ID, which is a number in front of your project name.
      In this case, it is "1".
3. Run the following command while specifying the project name and note the engine image ID:

   ```
   cdswctl engine-images list -p <project-name>
   ```

   For example,

   ```
   cdswctl engine-images list -p john-smith/petal-length-predictor
   ```

4. Create a model by using the following command:

   ```
   cdswctl models create
   --kernel="python3"
   --targetFilePath="predict.py"
   --targetFunctionName="predict"
   --name="Petal Length Predictor"
   --cpuMillicores=1000
   --memoryMb=2000
   --description="Model of the Iris dataset"
   --replicationType=fixed
   --numReplicas=1
   --visibility="private"
   --autoBuildModel
   --autoDeployModel
   --projectId=<project ID>
   --examples='{"request":{"petal_length":1}}'
   --engineImageId=<engine image ID from before>
   ```

   If the command runs successfully, the system displays the model details in a JSON format.
   For more information about the models create command parameters, run the following command:

   ```
   cdswctl models create --help
   ```

View replica logs for a model using the CLI

When a model is deployed, CDSW enables you to specify the number of replicas that must be deployed to serve requests. If a replica crashes or fails to come up, you can diagnose it by viewing the logs for every replica using the model CLI.

Procedure

1. Obtain the modelReplicaId by using the following command:

   ```
   cdswctl models listReplicas --modelDeploymentId=<model_deployment_ID>
   ```

   where the *model_deployment_ID* is the ID of a successfully deployed model.
2. To view the replica logs, run the following command:

   ```
   cdswctl models getReplicaLogs --modelDeploymentId=<model_deployment_ID> --
   modelReplicaId="<replica_ID>" --streams=stdout
   ```

   For example:

   ```
   cdswctl models getReplicaLogs --modelDeploymentId=2 --modelReplicaId="pe
   tal-length-predictor-1-2-6d6496b467-hp6tz" --streams=stdout
   ```

   The valid values for the streams parameter are stdout and stderr.

Using ML Runtimes with cdswctl

If a project is configured to use Runtimes, cdswctl workflows for starting sessions or models are slightly different.

Querying the engine type

You can query whether a project is configured using ML Runtimes or Legacy Engine.

Procedure

To determine if a project is configured to use either ML Runtimes or Legacy Engines, use the cdswctl projects get EngineType command and specify the project with the -p parameter.

For example, to determine if configured to use ML Runtimes:

```
cdswctl projects getEngineType -p demouser/runtimeproject
ml_runtime
```

```
cdswctl projects getEngineType -p demouser/legacyproject
legacy_engine
```

Listing runtimes

The first step to working with projects using runtimes is to query the available runtimes using the cdswctl runtimes list command.

About this task

The cdswctl runtimes list command returns all runtimes in a large JSON result. For easier consumption, you can post-process this result with some 3rd-party tool, such as jq or Python's json.tool.

Procedure

To query the available runtimes, use the cdswctl runtimes list command.

Note: The following examples are for presentation purposes only. Neither Python's json.tool nor jq are supported directly by Cloudera.

The following example pipes the cdswctl runtimes list result through Python's json.tool to produce a more readable output:

```
user@host:~ $ cdswctl runtimes list | python3 -m json.tool
{
    "runtimes": [
        {
            "id": 1,
            "imageIdentifier": "docker.repository.cloudera.com/cdsw/ml-run
time-workbench-python3.6-standard:2020.11.1-b6",
            "editor": "Workbench",
            "kernel": "Python 3.6",
            "edition": "Standard",
            "shortVersion": "2020.11",
            "fullVersion": "2020.11.1-b6",
            "maintenanceVersion": 1,
            "description": "Standard edition Python runtime provided by Cl
oudera"
        },
        {
            "id": 2,
            "imageIdentifier": "docker.repository.cloudera.com/cdsw/ml-run
time-jupyterlab-python3.7-standard:2020.11.1-b6",
            "editor": "JupyterLab",
            "kernel": "Python 3.7",
            "edition": "Technical Preview",
            "shortVersion": "2020.11",
            "fullVersion": "2020.11.1-b6",
            "maintenanceVersion": 1,
            "description": "Technical Preview JupyterLab Python runtime pro
vided by Cloudera"
        }
    ]
}
```

The following example pipes the cdswctl runtimes list result through jq to transform the JSON output into arbitrary formats:

```
user@host:~ $ cdswctl runtimes list | jq -r '.runtimes[] | "\(.id) \(.imageI
dentifier)"'
1
docker.repository.cloudera.com/cdsw/ml-runtime-workbench-python3.6-standa
rd:2020.11.1-b6
2
docker.repository.cloudera.com/cdsw/ml-runtime-jupyterlab-python3.7-standa
rd:2020.11.1-b6
```

The following example filters the cdswctl runtimes list result using jq to only show runtimes with specific editors and kernels:

```
user@host:~ $ cdswctl runtimes list | jq  '.runtimes[] | select((.editor ==
"Workbench") and (.kernel | contains("Python")))'

{
```

```
  "id": 1,
  "imageIdentifier": "docker.repository.cloudera.com/cdsw/ml-runtime-workben
ch-python3.6-standard:2020.11.1-b6",
  "editor": "Workbench",
  "kernel": "Python 3.6",
  "edition": "Standard",
  "shortVersion": "2020.11",
  "fullVersion": "2020.11.1-b6",
  "maintenanceVersion": 1,
  "description": "Standard edition Python runtime provided by Cloudera"
}
```

Starting sessions and creating SSH endpoints

Once you choose a runtime, you can start a session using the cdswctl sessions start command and create SSH endpoints using the cdswctl ssh-endpoint command.

About this task

The runtime ID used in the following steps is obtained using the steps outlined in *Listing runtimes*.

Procedure

1. To start a session with a runtime, use the cdswctl sessions start command, specifying the runtime ID with the -r parameter and the project with the -p parameter.

   For example:

   ```
   cdswctl sessions start -r 2 -p demouser/runtimeproject
   ```

2. To specify SSH endpoints for the runtimes sessions, use the cdswctl ssh-endpoint command and specify the runtime ID using the -r parameter. and the project with the -p parameter.

   For example:

   ```
   cdswctl ssh-endpoint -r 1 -p demouser/runtimeproject
   ```

Creating a model

Creating a model in a project that uses runtimes is similar to model creation with an legacy engine, but you must use a different parameter to specify the runtime ID.

About this task

To create a model in a project that uses runtimes you must use the --runtimeId= parameter to specify a runtime ID (instead of using the --engineImageId= and --kernel= parameters used for a legacy engine).

Procedure

To create a model in a project that uses runtimes use the --runtimeId= parameter to specify a runtime ID.

For example:

```
cdswctl models create --targetFilePath=predict.py --targetFunctionName=predi
ct
  --projectId=4 --name=created-using-cdswctl --description=created-using-cds
wctl
  --memoryMb=1024 --authEnabled --cpuMillicores=250 --autoBuildModel --aut
oDeployModel
  --examples='{"request":{"petal_length":1}}' --runtimeId=1
```