

Engines for Experiments and Models

Date published: 2020-02-28

Date modified: 2020-12-15

CLOUDERA

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

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

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

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

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

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

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

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

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

Contents

Engines for Experiments and Models.....	4
Snapshot Code.....	4
Build Image.....	4
Run Experiment / Deploy Model.....	6

Engines for Experiments and Models

In Cloudera Data Science Workbench, models, experiments, jobs, and sessions are all created and executed within the context of a project.

We've described the different ways in which you can customize a project's engine environment for sessions and jobs [here](#). However, engines for models and experiments are completely isolated from the rest of the project.

Every time a model or experiment is kicked off, Cloudera Data Science Workbench creates a new isolated Docker image where the model or experiment is executed. This isolation in build and execution makes it possible for Cloudera Data Science Workbench to keep track of input and output artifacts for every experiment you run. In case of models, versioned builds give you a way to retain build history for models and a reliable way to rollback to an older version of a model if needed.



The rest of this topic describes the engine build process that occurs when you kick off a model or experiment.

Snapshot Code

When you first launch an experiment or model, Cloudera Data Science Workbench takes a Git snapshot of the project filesystem at that point in time. It is important to note that this Git server functions behind the scenes and is completely separate from any other Git version control system you might be using for the project as a whole.

However, this Git snapshot will recognize the `.gitignore` file defined in the project. This means if there are any artifacts (files, dependencies, etc.) larger than 50 MB stored directly in your project filesystem, make sure to add those files or folders to `.gitignore` so that they are not recorded as part of the snapshot. This ensures that the experiment or model environment is truly isolated and does not inherit dependencies that have been previously installed in the project workspace.

By default, each project is created with the following `.gitignore` file:

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

Augment this file to include any extra dependencies you have installed in your project workspace to ensure a truly isolated workspace for each model or experiment.

Multiple `.gitignore` files

A project can include multiple `.gitignore` files. However, there can only be one `.git` directory in the project, located at the project root, `/home/cdsw/.git`, otherwise Experiment and Model deployment fails.

If you create a blank project, and then want to clone a repo into it, clone a single project to the root of the workspace (`/home/cdsw/.git`) to ensure that Experiments and Models work.

Build Image

Once the code snapshot is available, Cloudera Data Science Workbench creates a new Docker image with a copy of the snapshot.

This new image is based off the project's designated default engine image (configured at [Project Settings Engine](#)). The image environment can be customized by using environmental variables and a build script that specifies which packages should be included in the new image.

Environmental Variables

Previously (CDSW 1.7.1 and lower), the environment variables set at the site admin level and project level did not automatically get pulled into the builds created for models and experiments. They needed to be explicitly coded into the `cdsw-build.sh` file. With CDSW 1.7.2 and higher, experiments and models will automatically inherit these admin and project-level environment variables.

Note that custom mounts or environment variables configured in `cdsw.conf` (such as `NO_PROXY`, `HTTP(S)_PROXY`, etc.) are still not passed to the container builds for experiments and models (even though they are applied to sessions, jobs, and deployed models/experiments).

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

Build Script - `cdsw-build.sh`

As part of the Docker build process, Cloudera Data Science Workbench runs a build script called `cdsw-build.sh` file. You can use this file to customize the image environment by specifying any dependencies to be installed for the code to run successfully. One advantage to this approach is that you now have the flexibility to use different tools and libraries in each consecutive training run. Just modify the build script as per your requirements each time you need to test a new library or even different versions of a library.



Important:

- The `cdsw-build.sh` script does not exist by default -- it has to be created by you within each project as needed.
- The name of the file is not customizable. It must be called `cdsw-build.sh`.

The following sections demonstrate how to specify dependencies in Python and R projects so that they are included in the build process for models and experiments.

Python 3

For Python, create a `requirements.txt` file in your project with a list of packages that must be installed. For example:

Figure 1: `requirements.txt`

```
beautifulsoup4==4.6.0
seaborn==0.7.1
```

Then, create a `cdsw-build.sh` file in your project and include the following command to install the dependencies listed in `requirements.txt`.

Figure 2: `cdsw-build.sh`

```
pip3 install -r requirements.txt
```

Now, when `cdsw-build.sh` is run as part of the build process, it will install the `beautifulsoup4` and `seaborn` packages to the new image built for the experiment/model.

R

For R, create a script called `install.R` with the list of packages that must be installed. For example:

Figure 3: `install.R`

```
install.packages(repos="https://cloud.r-project.org", c("tidyr",
"stringr"))
```

Then, create a `cds-sw-build.sh` file in your project and include the following command to run `install.R`.

Figure 4: `cds-sw-build.sh`

```
Rscript install.R
```

Now, when `cds-sw-build.sh` is run as part of the build process, it will install the `tidyr` and `stringr` packages to the new image built for the experiment/model.

If you do not specify a build script, the build process will still run to completion, but the Docker image will not have any additional dependencies installed. At the end of the build process, the built image is then pushed to an internal Docker registry so that it can be made available to all the Cloudera Data Science Workbench hosts. This push is largely transparent to the end user.



Note: If you want to test your code in an interactive session before you run an experiment or deploy a model, run the `cds-sw-build.sh` script directly in the workbench. This will allow you to test code in an engine environment that is similar to one that will eventually be built by the model/experiment build process.

Run Experiment / Deploy Model

Once the Docker image has been built and pushed to the internal registry, the experiment/model can now be executed within this isolated environment.

In case of experiments, you can track live progress as the experiment executes in the experiment's Session tab.

Unlike experiments, models do not display live execution progress in a console. Behind the scenes, Cloudera Data Science Workbench will move on to deploying the model in a serving environment based on the computing resources and replicas you requested. Once deployed you can go to the model's Monitoring page to view statistics on the number of requests served/dropped and `stderr/stdout` logs for the model replicas.