Cloudera Data Science Workbench

Cloudera Data Science Workbench Engines

Date published: 2020-02-28
Date modified: 2021-02-25

https://docs.cloudera.com/

Legal Notice

Contents

Cloudera Data Science Workbench Engines

Cloudera Data Science Workbench engines are responsible for running R, Python, and Scala code written by users and intermediating access to the CDH cluster.

You can think of an engine as a virtual machine, customized to have all the necessary dependencies to access the CDH cluster while keeping each project's environment entirely isolated. To ensure that every engine has access to the parcels and client configuration managed by the Cloudera Manager Agent, a number of folders are mounted from the host into the container environment. This includes the parcel path -/opt/cloudera, client configuration, as well as the host's JAVA_HOME. For more details on basic concepts and terminology related to engines in Cloudera Data Science Workbench, see Cloudera Data Science Workbench Engines.

Known Issues

- Apache Phoenix requires additional configuration to run commands successfully from within Cloudera Data Science Workbench engines (sessions, jobs, experiments, models).

  Workaround

  Explicitly set HBASE_CONF_PATH to a valid path before running Phoenix commands from engines.

  ```
  export HBASE_CONF_PATH=/usr/hdp/hbase/<hdp_version>/0/
  ```

Basic Concepts and Terminology

This section provides basic concepts and terminology for Cloudera Data Science Workbench engines.

Base Engine Image

The base engine image is a Docker image that contains all the building blocks needed to launch a Cloudera Data Science Workbench session and run a workload. It consists of kernels for Python, R, and Scala along with additional libraries that can be used to run common data analytics operations. When you launch a session to run a project, an engine is kicked off from a container of this image. The base image itself is built and shipped along with Cloudera Data Science Workbench.

New versions of the base engine image are released periodically. However, existing projects are not automatically upgraded to use new engine images. Older images are retained to ensure you are able to test code compatibility with the new engine before upgrading to it manually.

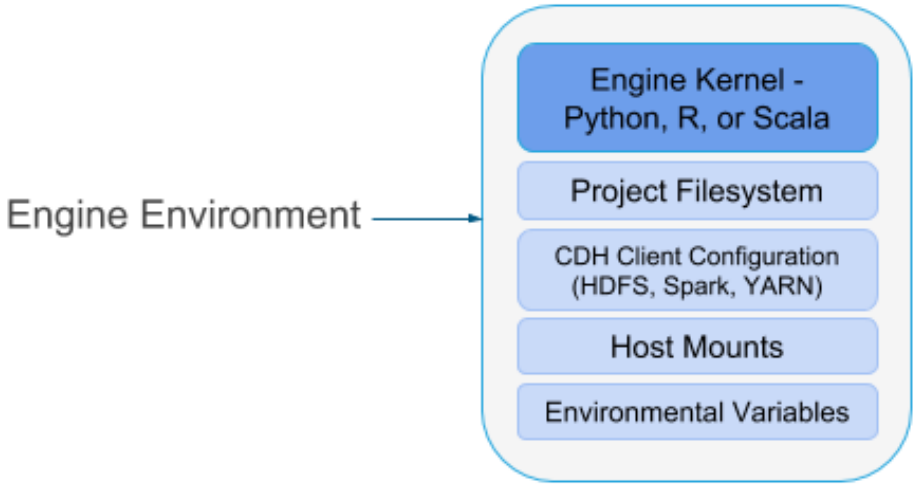
For more details on the libraries shipped within the base engine image, see Cloudera Data Science Workbench Engine Versions and Packaging.

Engine

The term engine refers to a virtual machine-style environment that is created when you run a project (via session or job) in Cloudera Data Science Workbench. You can use an engine to run R, Python, and Scala workloads on data stored in the underlying CDH cluster.

Cloudera Data Science Workbench allows you to run code using either a session or a job. A session is a way to interactively launch an engine and execute code while a job lets you batch process those actions and schedule them to run recursively. Each session and job launches its own engine that lives as long as the workload is running (or until it times out).

A running engine includes the following components:



- Kernel

  Each engine runs a kernel with an R, Python or Scala process that can be used to execute code within the engine. The kernel launched differs based on the option you select (either Python 2/3, PySpark, R, or Scala) when you launch the session or configure a job.

  The Python kernel is based on the Jupyter IPython kernel; the R kernel is custom-made for CDSW; and the Scala kernel is based on the Apache Toree kernel.
- Project Filesystem Mount

  Cloudera Data Science Workbench uses a persistent filesystem to store project files such as user code, installed libraries, or even small data files. Project files are stored on the master host at /var/lib/cdsw/current/projects.

  Every time you launch a new session or run a job for a project, a new engine is created ,and the project filesystem is mounted into the engine's environment at /home/cdsw. Once the session/job ends, the only project artifacts that remain are a log of the workload you ran, and any files that were generated or modified, including libraries you might have installed. All of the installed dependencies persist through the lifetime of the project. The next time you launch a session/job for the same project, those dependencies will be mounted into the engine environment along with the rest of the project filesystem.
- CDH and Host Mounts

  To ensure that each engine is able to access the CDH cluster, a number of folders are mounted from the CDSW gateway host into the engine's environment. For example, on a CSD deployment, this includes the path to the parcel repository (/opt/cloudera), client configurations for HDFS, Spark, YARN, as well as the host's JAVA_HOME.

  Cloudera Data Science Workbench works out-of-the-box for CDH clusters that use the default file system layouts configured by Cloudera Manager. If you customized your CDH cluster's filesystem layout (for example, modified the CDH parcel directory) or if there are other files on the hosts that should be mounted into the engines, use the Site Administration panel to include them.

  For detailed instructions, see Customized Engine Images.

ML Runtimes versus Legacy Engines

Cloudera Data Science Workbench offers both legacy engines and Machine Learning Runtimes. Both legacy engines and ML Runtimes are Docker images and contain OS, interpreters, and libraries to run user code in sessions, jobs, experiments, models, and applications. However, there are significant differences between these choices.

Legacy engines are monolithic in the sense that they contain the machinery necessary to run sessions using all four interpreter options that CML currently supports (Python 2, Python 3, R and Scala) and other support utilities (C and Fortran compilers, LaTeX, etc.).

ML Runtimes are thinner and more lightweight than the current monolithic engines. Rather than supporting multiple programming languages in a single engine, each Runtime variant supports a single interpreter version and a subset of utilities and libraries to run the user's code in Sessions, Jobs, Experiments, Models, or Applications.

While the end form factor (a docker image) remains the same for legacy engines and ML Runtimes, the build architecture and release process of ML Runtimes differs from legacy engines. Versioning and metadata helps make ML Runtimes content simpler to understand, both for the host workload app and the user.

Project Environments

This section describes how you can configure engine environments to meet the requirements of a project. This can be done by using environmental variables and by installing dependencies.

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/job timeout windows, you can use environmental variables to do so. Environmental variables can also be used to assign variable names to secrets, such as passwords or authentication tokens, to avoid including these directly in the code.

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

Dependencies

You can provide packages, such as Python libraries, in addition to the pre-installed package through the three methods.

You can provide packages, such as Python libraries, in addition to the pre-installed package through the following methods:

- Directly installing packages within projects
- Creating a custom engine with the required packages
- Mounting a path from the host which contains additional packages

One method may be more appropriate for your deployment than another method. For more information about each option, see Managing Engine Dependencies.

Configuring Engine Environments for Experiments and Models

To allow for versioning of experiments and models, Cloudera Data Science Workbench executes each experiment and model in a completely isolated engine.

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. These engines are built by extending the project's designated default engine image to include the code to be executed and any dependencies as specified.

For details on how this process works and how to configure these environments, see https://docs.cloudera.com/cdsw/1.9.1/models-and-experiments/topics/cdsw-engines-models-experiments.html.