Cloudera Data Science Workbench

Configuring Cloudera Data Science Workbench Engines

Date published: 2020-02-28
Date modified: 2020-12-15

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

Configuring Cloudera Data Science Workbench Engines

This topic describes how to configure and manage engines in Cloudera Data Science Workbench. Cloudera Data Science Workbench currently supports R, Python, and Scala engines.

You can use these engines to run data science projects either in isolation, as you would on your laptop, or connect to your CDH cluster using Cloudera Distribution of Apache Spark 2 and other libraries.

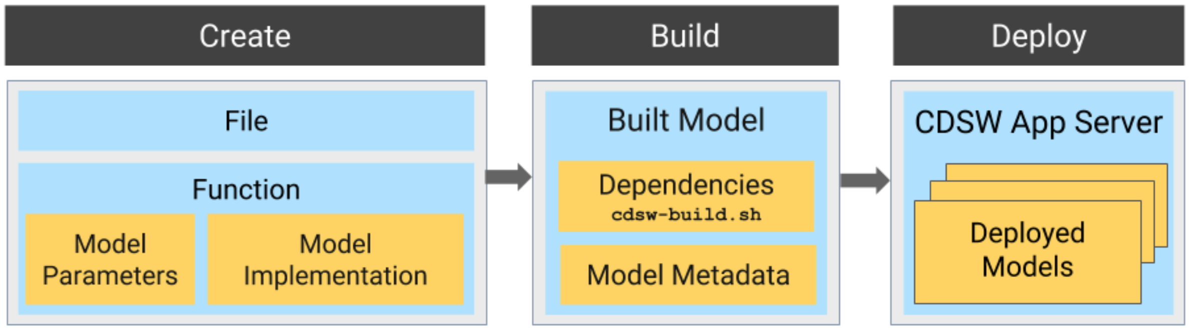
Concepts and Terminology

This topic provides model conceptual information.

Model

Model is a high level abstract term that is used to describe several possible incarnations of objects created during the model deployment process. For the purpose of this discussion you should note that 'model' does not always refer to a specific artifact. More precise terms (as defined later in this section) should be used whenever possible.

Stages of the Model Deployment Process



The rest of this section contains supplemental information that describes the model deployment process in detail.

Create

- File - The R or Python file containing the function to be invoked when the model is started.
- Function - The function to be invoked inside the file. This function should take a single JSON-encoded object (for example, a python dictionary) as input and return a JSON-encodable object as output to ensure compatibility with any application accessing the model using the API. JSON decoding and encoding for model input/output is built into Cloudera Data Science Workbench.

  The function will likely include the following components:

  - Model Implementation

    The code for implementing the model (e.g. decision trees, k-means). This might originate with the data scientist or might be provided by the engineering team. This code implements the model's predict function, along with any setup and teardown that may be required.
  - Model Parameters

    A set of parameters obtained as a result of model training/fitting (using experiments). For example, a specific decision tree or the specific centroids of a k-means clustering, to be used to make a prediction.

Build

This stage takes as input the file that calls the function and returns an artifact that implements a single concrete model, referred to as a model build.

- Built Model

  A built model is a static, immutable artifact that includes the model implementation, its parameters, any runtime dependencies, and its metadata. If any of these components need to be changed, for example, code changes to the implementation or its parameters need to be retrained, a new build must be created for the model. Model builds are versioned using build numbers.

  To create the model build, Cloudera Data Science Workbench creates a Docker image based on the engine designated as the project's default engine. This image provides an isolated environment where the model implementation code will run.

  To configure the image environment, you can specify a list of dependencies to be installed in a build script called cdsw-build.sh.

  For details about the build process and examples on how to install dependencies, see *Engines for Experiments and Models*.
- Build Number:

  Build numbers are used to track different versions of builds within the scope of a single model. They start at 1 and are incremented with each new build created for the model.

Deploy

This stage takes as input the memory/CPU resources required to power the model, the number of replicas needed, and deploys the model build created in the previous stage to a REST API.

- Deployed Model

  A deployed model is a model build in execution. A built model is deployed in a model serving environment, likely with multiple replicas.
- Environmental Variable

  You can set environmental variables each time you deploy a model. Note that models also inherit any environment variables set at the project and global level. However, in case of any conflicts, variables set per-model will take precedence.

  Note:

  - If you are using any model-specific environmental variables, these must be specified every time you re-deploy a model. Models do not inherit environmental variables from previous deployments.
  - 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).
- Model Replicas

  The engines that serve incoming requests to the model. Note that each replica can only process one request at a time. Multiple replicas are essential for load-balancing, fault tolerance, and serving concurrent requests. Cloudera Data science Workbench allows you to deploy a maximum of 9 replicas per model.
- Deployment ID

  Deployment IDs are numeric IDs used to track models deployed across Cloudera Data Science Workbench. They are not bound to a model or project.

Managing Engines

Site administrators and project administrators are responsible for making sure that all projects on the deployment have access to the engines they need.

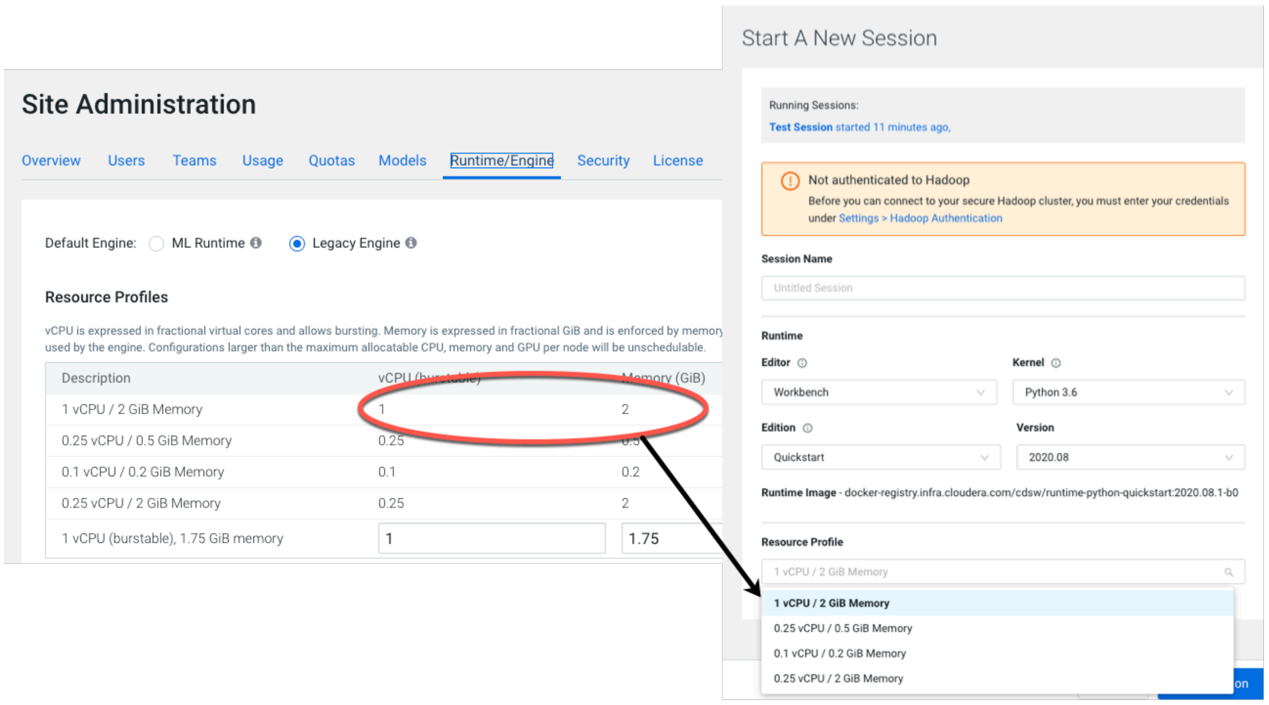
Required Role: Site Administrator

Site admins can create engine profiles, determine the default engine version to be used across the deployment, and white-list any custom engines that teams require. As a site administrator, you can also customize engine environments by setting global environmental variables and configuring any files/folders that need to be mounted into project environments on run time.

Managing Resource Profiles

Resource profiles define how many vCPUs and how much memory Cloudera Data Science Workbench will reserve for a particular workload (for example, session, job, model).

As a site administrator you can create several different vCPU, GPU, and memory configurations which will be available when launching a session/job. When launching a new session, users will be able to select one of the available resource profiles depending on their project's requirements.



Default Engine: ML Runtime [✓ Legacy Engine]

Resource Profiles

vCPU is expressed in fractional virtual cores and allows bursting. Memory is expressed in fractional GiB and is enforced by memory used by the engine. Configurations larger than the maximum allocatable CPU, memory and GPU per node will be unschedulable.

Description	vCPU (burstable)	Memory (GiB)
1 vCPU / 2 GiB Memory	1	2
0.25 vCPU / 0.5 GiB Memory	0.25	0.5
0.1 vCPU / 0.2 GiB Memory	0.1	0.2
0.25 vCPU / 2 GiB Memory	0.25	2
1 vCPU (burstable), 1.75 GiB memory	1	1.75

Start A New Session

Running Sessions:
Test Session started 11 minutes ago,

Not authenticated to Hadoop
Before you can connect to your secure Hadoop cluster, you must enter your credentials under Settings > Hadoop Authentication

Session Name: Untitled Session

Runtime
Editor: Workbench
Kernel: Python 3.6
Edition: Quickstart
Version: 2020.08

Runtime Image - docker-registry.infra.cloudera.com/cdsw/runtime-python-quickstart:2020.08.1-b0

Resource Profile: 1 vCPU / 2 GiB Memory
- **1 vCPU / 2 GiB Memory**
- 0.25 vCPU / 0.5 GiB Memory
- 0.1 vCPU / 0.2 GiB Memory
- 0.25 vCPU / 2 GiB Memory

To create resource profiles, go to the Admin Engines/Runtimes page, under Resource Profiles. Cloudera recommends that all profiles include at least 2 GB of RAM to avoid out of memory errors for common user operations.

You will see the option to add GPUs to the resource profiles only if your Cloudera Data Science Workbench hosts are equipped with GPUs, and you have enabled them for use by setting the relevant properties either in Cloudera Manager (for CSD) or in cdsw.conf (for RPM).

Burstable CPUs

CDSW configures no upper bound on the CPU resources that Workloads can use so that they can use all of the CPU resources available on the node where they are running. By configuring no CPU limits, CDSW enables efficient use of the CPU resources available on your cluster nodes:

- If the CPUs are idle then the workloads can burst and take advantage of the free CPU cycles. For example, if you've launched a session with 1vCPU but the code inside it requires more than 1vCPU, the workload container can consume all the available CPU cycles on the node where it's launched.
- When the cluster is highly utilized and CPU resources are sparse, Workloads will be limited to use the number of CPU resources configured in their resource profile.
- If multiple containers are attempting to use excess CPU, CPU time is distributed in proportion to the amount of CPU initially requested by each container.

Managing Engine Images

By default, Cloudera Data Science Workbench ships a base engine image that includes kernels for Python, R, and Scala, along with some additional libraries that can be used to run common data analytics operations. Occasionally, new engine versions are released and shipped with Cloudera Data Science Workbench releases.

Engine images are available in the Site Administrator panel at Admin Engines , under the Engine Images section.

There are two types of default engines: ML Runtime and Legacy Engines. Legacy engines 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 legacy 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.

As a site administrator, you can select which engine version is used by default for new projects. Furthermore, project administrators can explicitly select which engine image should be used as the default image for a project.

1. Click Admin > Runtime/Engine.
2. Choose the Default Engine you would like to use as the default for all newly created projects in this workspace.
3. Modify the remaining information on the page:
   - Resource Profiles listed in the table are selectable resource options for both legacy Engines and ML Runtimes (for example, when starting a Session or Job)
   - The remaining information on the page applies to site-level settings specific for legacy Engines.

If a user publishes a new custom Docker image, site administrators are responsible for white-listing such images for use across the deployment. For more information on creating and managing custom Docker images, see https://docs.cloudera.com/cdsw/1.9.0/extensible-engines/topics/cdsw-extensible-engines.html.

Configuring the Engine Environment

This section describes some of the ways you can configure engine environments to meet the requirements of your projects.

Environmental Variables

For information on how environmental variables can be used to configure engine environments in Cloudera Data Science Workbench, see https://docs.cloudera.com/cdsw/1.9.0/environment-variables/topics/cdsw-environment-variables.html.

CDH Parcel Directory

Starting with Cloudera Data Science Workbench 1.5, the CDH parcel directory property is no longer available in the Site Administration panel. By default, Cloudera Data Science Workbench looks for the CDH parcel at /opt/cloudera/parcels.

If you want to use a custom location for your parcels, use one of the following methods to configure this custom location:

CSD deployments: If you are using the default parcel directory, /opt/cloudera/parcels, no action is required. If you want to use a custom location for the parcel directory, configure this in Cloudera Manager as documented here.

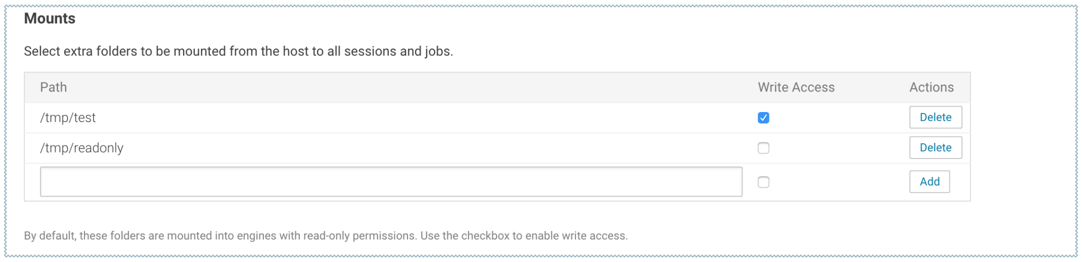
OR

RPM deployments: If you are using the default parcel directory, /opt/cloudera/parcels, no action is required. If you want to specify a custom location for the parcel directory, configure the DISTRO_DIR property in the cdsw.conf file on both master and worker hosts. Run cdsw restart after you make this change.

Configuring Host Mounts

By default, Cloudera Data Science Workbench will automatically mount the CDH parcel directory and client configuration for required services such as HDFS, Spark, and YARN into each project's engine. However, if users want to reference any additional files/folders on the host, site administrators will need to configure them here so that they are loaded into engine containers at runtime. Note that the directories specified here will be available to all projects across the deployment.

To configure additional mounts, go to Admin Engines and add the paths to be mounted from the host to the Mounts section.


Mounts

Select extra folders to be mounted from the host to all sessions and jobs.

Path	Write Access	Actions
/tmp/test	✓	Delete
/tmp/readonly	☐	Delete
	☐	Add

By default, these folders are mounted into engines with read-only permissions. Use the checkbox to enable write access.

The following table summarizes how mounts are loaded into engine containers in current and previous Cloudera Data Science Workbench releases.

CDSW Version	Mount Point Permissions in Engines
1.4.2 (and higher)	By default, mount points are loaded into engine containers with read-only permissions. CDSW 1.4.2 (and higher) also include a Write Access checkbox (see image) that you can use to enable read-write access for individual mounted directories. Note that these permissions will apply to all projects across the deployment.
1.4.0	Mount points are loaded into engine containers with read-only permissions.
1.3.x (and lower)	Mount points are loaded into engine containers with read-write permissions.

Points to Remember:

- When adding host mounts, try to be as generic as possible without mounting common system files. For example, if you want to add several files under /etc/spark2-conf, you can simplify and mount the /etc/spark2-conf directory; but adding the parent /etc might prevent the engine from running.

  As a general rule, do not mount full system directories that are already in use; such as /var, /tmp, or /etc. This also serves to avoid accidentally exposing confidential information in running sessions. Do not set JAVA_HOME to a path under these directories because CDSW sessions will mount the JAVA_HOME location from the host in the session engine container which can interfere with the operation of the engine container.
- Do not add duplicate mount points. This will lead to sessions crashing in the workbench.

Configuring Shared Memory Limit for Docker Images

You can increase the shared memory size for the sessions, experiments, and jobs running within an Engine container within your project. For Docker, the default size of the available shared memory is 64MB.

To increase the shared memory limit:

1. From CDSW web UI, go to Projects Project Settings Engine Advanced Settings .
2. Specify the shared memory size in the Shared Memory Limit field.
3. Click Save Advanced Settings to save the configuration and exit.

This mounts a volume with the tmpfs file system to /dev/shm and Kubernetes will enforce the given limit. The maximum size of this volume is the half of your physical RAM in the node without the swap.

Setting Time Zones for Sessions and Jobs

The default time zone for Cloudera Data Science Workbench sessions is UTC. This is the default regardless of the time zone setting on the Master host.

To change to your preferred time zone, for example, Pacific Standard Time (PST), navigate to Admin Engines . Under the Environmental Variables section, add a new variable with the name set to TZ and value set to America/Los_Angeles, and click Add.