Cloudera AI

Managing Engines in Cloudera AI

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

CLOUDƎRA

Legal Notice

Contents

Managing Engines

This topic describes how to manage engines and configure engine environments to meet your project requirements.

Required Role: EnvironmentAdmin

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

By default, Cloudera AI ships a base engine image that includes kernels for Python, R, and Scala, along with some additional libraries (see *Configuring Cloudera AI Engines* for more information) that can be used to run common data analytics operations. Occasionally, new engine versions are released and shipped with Cloudera AI releases.

Engine images are available in the Site Administrator panel at Admin Engines , under the Engine Images section. 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. To do so, go to the project's Overview page and click Settings on the left navigation bar.

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 *Configuring the Engine Environment*.

Related Information
Configuring the Engine Environment
Installing Additional Packages

Creating Resource profiles

Resource profiles define how many vCPUs and how much memory the product will reserve for a particular workload (for example, session, job, model).

About this task

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.

Procedure

1. To create resource profiles, go to the Site Administration Runtime/Engine page.
2. Add a new profile 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 AI hosts are equipped with GPUs, and you have enabled them for use by setting the relevant properties in cdsw.conf.

Results

If there are two worker nodes and 10 vCPU available overall, if one user tries to establish a session with 8 vCPU, CDSW will not allow it. The memory and CPU must be contiguous (adjacent to each other). When a user spins a

session, the pod triggers on a single node and resources on the same node are utilized. This is expected behavior for Kubernetes.

Figure 1: Resource profiles available when launching a session

Configuring the engine environment

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

Install Additional Packages

For information on how to install any additional required packages and dependencies to your engine, see *Installing Additional Packages*.

Environmental Variables

For information on how environmental variables can be used to configure engine environments in Cloudera AI, see Engine Environment Variables.

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 64 MB.

To increase the shared memory limit:

1. From the 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.

Related Information

Engine Environment Variables

Installing additional packages

Set up a custom repository location

You can set up a custom default location for Python and R code package repositories. This is especially useful for air-gapped clusters that are isolated from the PIP and CRAN repositories on the public internet.

Python PIP repository

Custom PIP repositories can be set as default for all engines at a site or project level. The environmental variables can be set at the Project or Site level. If the values are set at the Site level, they can be overridden at the Project level.

1. Set the environmental variables at the appropriate level.

- For Site level, go to: Site Administration Engine
- For Project level, go to: Project Settings Engine

2. To set a new default URL for the PIP index, enter:

- PIP_INDEX_URL = <new url>
- PIP_EXTRA_INDEX_URL = <new url>

CRAN repository

Custom CRAN repositories must be set in a session or as part of a custom engine. To set a new default URL for a CRAN repository, set the following in the /home/cdsw/.Rprofile file:

options(repos=structure(c(CRAN="<mirror URL>")))

Burstable CPUs

Cloudera AI 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, Cloudera AI 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 have 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 is 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.

Installing additional packages

Cloudera AI engines are preloaded with a few common packages and libraries for R, Python, and Scala. However, a key feature of Cloudera AI is the ability of different projects to install and use libraries pinned to specific versions, just as you would on your local computer.

Note: Before downloading or using third-party content, you are responsible for reviewing and complying with any applicable license terms and making sure that they are acceptable for your use case.

Generally, Cloudera recommends you install all required packages locally into your project. This will ensure you have the exact versions you want and that these libraries will not be upgraded when Cloudera upgrades the base engine image. You only need to install libraries and packages once per project. From then on, they are available to any new engine you spawn throughout the lifetime of the project.

You can install additional libraries and packages from the workbench, using either the command prompt or the terminal.

Note:

Cloudera AI does not currently support installation of packages that require root access to the hosts. For such use-cases, you will need to create a new custom engine that extends the base engine image to include the required packages. For instructions, see *Creating a Customized Engine Image*.

(Python and R) install packages using Workbench command prompt

To install a package from the command prompt:

1. Navigate to your project's Overview page. Click Open Workbench and launch a session.
2. At the command prompt (see Native Workbench Console and Editor) in the bottom right, enter the command to install the package. Some examples using Python and R have been provided.

R

```
# Install from CRAN
install.packages("ggplot2")

# Install using devtools
install.packages('devtools')
library(devtools)
install_github("hadley/ggplot2")
```

Python 2

```
# Installing from console using ! shell operator and pip:
!pip install beautifulsoup

# Installing from terminal
pip install beautifulsoup
```

Python 3

```
# Installing from console using ! shell operator and pip3:
!pip3 install beautifulsoup4
# Installing from terminal
pip3 install beautifulsoup4
```

(Python only) using a requirements file

For a Python project, you can specify a list of the packages you want in a requirements.txt file that lives in your project. The packages can be installed all at once using pip/pip3.

1. Create a new file called requirements.txt file within your project:

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

2. To install the packages in a Python 3 engine, run the following command in the workbench command prompt.

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

For Python 2 engines, use pip.

```
!pip install -r requirements.txt
```

Related Information
Conda

Using Conda to manage dependencies

You can install additional libraries and packages from the workbench, using either the command prompt or the terminal. Alternatively, you might choose to use a package manager such as Conda to install and maintain packages and their dependencies. This topic describes some basic usage guidelines for Conda.

Cloudera AI recommends using pip for package management along with a requirements.txt file (as described in the previous section). However, for users that prefer Conda, the default engine in Cloudera AI includes two environments called python2.7, and python3.6. These environments are added to sys.path, depending on the version of Python selected when you launch a new session.

In Python 2 and Python 3 sessions and attached terminals, Cloudera AI automatically sets the CONDA_DEFAULT_ENV and CONDA_PREFIX environment variables to point to Conda environments under /home/cdsw/.conda.

However, Cloudera AI does not automatically configure Conda to pin the actual Python version. Therefore if you are using Conda to install a package, you must specify the version of Python. For example, to use Conda to install the feather-format package into the python3.6 environment, run the following command in the Workbench command prompt:

```
!conda install -y -c conda-forge python=3.6.9 feather-format
```

To install a package into the python2.7 environment, run:

```
!conda install -y -c conda-forge python=2.7.17 feather-format
```

Note that on sys.path, pip packages have precedence over conda packages.

> **Note:**
> - Cloudera AI does not automatically configure a Conda environment for R and Scala sessions and attached terminals. If you want to use Conda to install packages from an R or Scala session or terminal, you must manually configure Conda to install packages into the desired environment.

Creating an extensible engine with Conda

Cloudera AI also allows you to configure the engine environment, to include packages of your choice using Conda. To create an extended engine:

1. Add the following lines to a Dockerfile to extend the base engine, push the engine image to your Docker registry, and include the new engine in the allowlist, for your project. For more details on this step, see *Configuring the engine environment*.

Python 2

```
RUN mkdir -p /opt/conda/envs/python2.7
```

```
RUN conda install -y nbconvert python=2.7.17 -n python2.7
```

Python 3

```
RUN mkdir -p /opt/conda/envs/python3.6
RUN conda install -y nbconvert python=3.6.9 -n python3.6
```

2. Set the PYTHONPATH environmental variable as shown below. You can set this either globally in the site administrator dashboard, or for a specific project by going to the project's Settings Engine page.

Python 2

```
PYTHONPATH=$PYTHONPATH:/opt/conda/envs/python2.7/lib/python2.7/site-pack
ages
```

Python 3

```
PYTHONPATH=$PYTHONPATH:/opt/conda/envs/python3.6/lib/python3.6/site-pack
ages
```

Related Information

Conda

Configuring the engine environment

Engine environment variables

This topic describes how engine environmental variables work. It also lists the different scopes at which they can be set and the order of precedence that will be followed in case of conflicts.

Environmental variables allow you to customize engine environments for projects. 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.

Cloudera AI allows you to define environmental variables for the following scopes:

Global

A site administrator for your Cloudera AI deployment can set environmental variables on a global level. These values will apply to every project on the deployment.

To set global environmental variables, go to Admin Runtime/Engines .

Project

Project administrators can set project-specific environmental variables to customize the engines launched for a project. Variables set here will override the global values set in the site administration panel.

To set environmental variables for a project, go to the project's Overview page and click Settings Advanced .

Job

Environments for individual jobs within a project can be customized while creating the job. Variables set per-job will override the project-level and global settings.

To set environmental variables for a job, go to the job's Overview page and click Settings Set Environmental Variables .

Experiments

Engines created for execution of experiments are completely isolated from the project. However, these engines inherit values from environmental variables set at the project-level and/or global level. Variables set at the project-level will override the global values set in the site administration panel.

Models

Model environments are completely isolated from the project. Environmental variables for these engines can be configured during the build stage of the model deployment process. Models will also inherit any environment variables set at the project and global level. However, variables set per-model build will override other settings.

Engine environment variables

The following table lists Cloudera AI environment variables that you can use to customize your project environments. These can be set either as a site administrator or within the scope of a project or a job.

Environment Variable	Description
MAX_TEXT_LENGTH	Maximum number of characters that can be displayed in a single text cell. By default, this value is set to 800,000 and any more characters will be truncated. Default: 800,000
PROJECT_OWNER	The name of the Team or user that created the project.
SESSION_MAXIMUM_MINUTES	Maximum number of minutes a session can run before it times out. Default: 60*24*7 minutes (7 days) Maximum Value: 35,000 minutes
JOB_MAXIMUM_MINUTES	Maximum number of minutes a job can run before it times out. Default: 60*24*7 minutes (7 days) Maximum Value: 35,000 minutes
IDLE_MAXIMUM_MINUTES	Maximum number of minutes a session can remain idle before it exits. Default: 60 minutes Maximum Value: 35,000 minutes Idle timeouts for sessions vary by workbench type (runtime). • Standard Workbench: Sessions timeout regardless of activity in the browser or terminal. • PBJ Workbench: Sessions timeout if there is no browser activity and no terminal window is open. If a terminal window is open, the session will not timeout, regardless of whether there is activity in the terminal window. • Jupyterlab: Sessions timeout if there is no browser activity. Terminal window activity is not considered. • Custom runtimes: No idle timeout behavior is enforced on custom or third-party workbenches.
CONDA_DEFAULT_ENV	Points to the default Conda environment so you can use Conda to install/manage packages in the Workbench. For more details on when to use this variable, see *Installing Additional Packages*.

Per-Engine Environmental Variables: In addition to the previous table, there are some more built-in environmental variables that are set by the Cloudera AI application itself and do not need to be modified by users. These variables are set per-engine launched by Cloudera AI and only apply within the scope of each engine.

Environment Variable	Description
CDSW_PROJECT	The project to which this engine belongs.
CDSW_PROJECT_ID	The ID of the project to which this engine belongs.

Environment Variable	Description
CDSW_ENGINE_ID	The ID of this engine. For sessions, this appears in your browser's URL bar.
CDSW_MASTER_ID	If this engine is a worker, this is the CDSW_ENGINE_ID of its master.
CDSW_MASTER_IP	If this engine is a worker, this is the IP address of its master.
CDSW_PUBLIC_PORT	**Note:** This property is deprecated. See CDSW_APP_PORT and CDSW_READONLY_PORT for alternatives. A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_PUBLIC_PORT will be available in browsers at: http(s)://read-only-<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>. By default, CDSW_PUBLIC_PORT is set to 8080. A direct link to these web services will be available from the grid icon in the upper right corner of the Cloudera AI web application, as long as the job or session is still running. For more details, see *Accessing Web User Interfaces from Cloudera AI*. In Cloudera AI, setting CDSW_PUBLIC_PORT to a non-default port number is not supported.
CDSW_APP_PORT	A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_APP_PORT will be available in browsers at: http(s)://read-only-<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>. Use this port for applications that grant some control to the project, such as access to the session or terminal. A direct link to these web services will be available from the grid icon in the upper right corner of the Cloudera AI web application as long as the job or session runs. Even if the web UI does not have authentication, only Contributors and those with more access to the project can access it. For more details, see *Accessing Web User Interfaces from Cloudera AI*. Note that if the Site Administrator has enabled Allow only session creators to run commands on active sessions, then the UI is only available to the session creator. Other users will not be able to access it. Use 127.0.0.1 as the IP.
CDSW_READONLY_PORT	A port on which you can expose HTTP services in the engine to browsers. HTTP services that bind CDSW_READONLY_PORT will be available in browsers at: http(s)://read-only-<$CDSW_ENGINE_ID>.<$CDSW_DOMAIN>. Use this port for applications that grant read-only access to project results. A direct link to these web services will be available to users with from the grid icon in the upper right corner of the Cloudera AI web application as long as the job or session runs. Even if the web UI does not have authentication, Viewers and those with more access to the project can access it. For more details, see *Accessing Web User Interfaces from Cloudera AI*. Use 127.0.0.1 as the IP.
CDSW_DOMAIN	The domain on which Cloudera AI is being served. This can be useful for iframing services, as demonstrated in *Accessing Web User Interfaces from Cloudera AI*.
CDSW_CPU_MILLICORES	The number of CPU cores allocated to this engine, expressed in thousandths of a core.
CDSW_MEMORY_MB	The number of megabytes of memory allocated to this engine.
CDSW_IP_ADDRESS	Other engines in the Cloudera AI cluster can contact this engine on this IP address.
CDSW_APP_POLLING_ENDPOINT	Specify a custom endpoint that Cloudera AI uses to check the status of the application. The default value is '/'.

Related Information
Installing Additional Packages

Accessing environmental variables from projects

This topic shows you how to access environmental variables from your code.

Environmental variables are injected into every engine launched for a project, contingent on the scope at which the variable was set (global, project, etc.). The following code samples show how to access a sample environment variable called *DATABASE_PASSWORD* from your project code.

R

```
database.password <- Sys.getenv("DATABASE_PASSWORD")
```

Python

```
import os
database_password = os.environ["DATABASE_PASSWORD"]
```

Scala

```
System.getenv("DATABASE_PASSWORD")
```

Appending Values to Environment Variables:

You can also set environment variables to append to existing values instead of replacing them. For example, when setting the LD_LIBRARY_PATH variable, you can set the value to LD_LIBRARY_PATH:/path/to/set.

Customized engine images

This topic explains how custom engines work and when they shall be used.

By default, Cloudera AI engines are preloaded with a few common packages and libraries for R, Python, and Scala. In addition to these, Cloudera AI also allows you to install any other packages or libraries that are required by your projects. However, directly installing a package to a project as described above might not always be feasible. For example, packages that require root access to be installed, or that must be installed to a path outside /home/cdsw (outside the project mount), cannot be installed directly from the workbench.

For such circumstances, Cloudera AI allows you to extend the base Docker image and create a new Docker image with all the libraries and packages you require. Site administrators can then include this new image in the allowlist for use in projects, and project administrators set the new white-listed image to be used as the default engine image for their projects. For an end-to-end example of this process, see *End-to-End Example: MeCab*.

Note: You will need to remove any unnecessary Cloudera sources or repositories that are inaccessible because of the paywall.

Note that this approach can also be used to accelerate project setup across the deployment. For example, if you want multiple projects on your deployment to have access to some common dependencies (package or software or driver) out of the box, or even if a package just has a complicated setup, it might be easier to simply provide users with an engine that has already been customized for their project(s).

Related Resources

- The Cloudera Engineering Blog post on *Customizing Docker Images in Cloudera Maching Learning* describes an end-to-end example on how to build and publish a customized Docker image and use it as an engine in Cloudera AI.
- For an example of how to extend the base engine image to include Conda, see *Installing Additional Packages*.

Related Information

End-to-end example: MeCab

Installing Additional Packages

Customizing Docker Images in Cloudera AI

Creating a customized engine image

This section walks you through the steps required to create your own custom engine based on the Cloudera AI base image.

For a complete example, see *End-to-End Example: MeCab*.

Create a Dockerfile for the custom image

This topic shows you how to create a Dockerfile for a custom image.

The first step when building a customized image is to create a Dockerfile that specifies which packages you would like to install in addition to the base image.

For example, the following Dockerfile installs the beautifulsoup4 package on top of the base Ubuntu image that ships with Cloudera AI.

```
# Dockerfile

# Specify a Cloudera AI base image
FROM docker.repository.cloudera.com/cloudera/cdsw/engine:13-cml-2021.02-1

# Update packages on the base image and install beautifulsoup4
RUN apt-get update
RUN pip install beautifulsoup4 && pip3 install beautifulsoup4
```

Build the new Docker image

This topic shows you how to use Docker to build a custom image.

A new custom Docker image can be built on any host where Docker binaries are installed. To install these binaries, run the following command on the host where you want to build the new image:

```
docker build -t <image-name>:<tag> . -f Dockerfile
```

If you want to build your image on the Cloudera AI Workbench, you must add the --network=host option to the build command:

```
docker build --network=host -t <image-name>:<tag> . -f Dockerfile
```

Distribute the image

This topic explains the different methods that can be used to distribute a custom engine to all the hosts.

Once you have built a new custom engine, use one of the following ways to distribute the new image to all your Cloudera AI hosts:

Push the image to a public registry such as DockerHub

For instructions, refer the Docker documentation *docker push* and *Push images to Docker Cloud*.

Push the image to your company's Docker registry

When using this method, make sure to tag your image with the following schema:

```
docker tag <image-name> <company-registry>/<user-name>/<image-na
me>:<tag>
```

Once the image has been tagged properly, use the following command to push the image:

```
docker push <company-registry>/<user-name>/<image-name>:<tag>
```

The MeCab example at the end of this topic uses this method.

Related Information

docker push

Including images in allowlist for Cloudera AI projects

This topic describes how to include custom images in the allowlist so that they can be used in projects.

Including a customized image in Cloudera AI is a two-step process.

1. Include the image in the allowlist for the whole deployment.

   First, a site administrator will need to clear the new image for use on the deployment.

   a. Log in as a site administrator.
   b. Click Admin Engines .
   c. Add <company-registry>/<user-name>/<image-name>:<tag> to the allowlist of engine images.
2. Include the image in the allowlist for a specific project

   If you want to start using the image in a project, the project administrator will need to set this image as the default image for the project.

   a. Go to the project Settings page.
   b. Click Engines.
   c. Select the new customized engine from the drop-down list of available Docker images. Sessions and jobs you run in your project will now have access to this engine.

Limitations with customized engines

This topic lists some limitations associated with custom engines.

- Cloudera AI only supports customized engines that are based on the Cloudera AI base image.
- Cloudera AI does not support creation of custom engines larger than 10 GB.

  Cloudera Bug: DSE-4420
- Cloudera AI does not support pulling images from registries that require Docker credentials.

  Cloudera Bug: DSE-1521
- The contents of certain pre-existing standard directories such as /home/cdsw, /tmp, and so on, cannot be modified while creating customized engines. This means any files saved in these directories will not be accessible from sessions that are running on customized engines.

  Workaround: Create a new custom directory in the Dockerfile used to create the customized engine, and save your files to that directory.

End-to-end example: MeCab

This topic walks you through a simple end-to-end example on how to build and use custom engines.

This section demonstrates how to customize the Cloudera AI base engine image to include the MeCab (a Japanese text tokenizer) library.

This is a sample Dockerfile that adds MeCab to the Cloudera AI base image.

```
# Dockerfile

FROM docker.repository.cloudera.com/cloudera/cdsw/engine:13-cml-2021.02-1
RUN rm /etc/apt/sources.list.d/*
RUN apt-get update && \
    apt-get install -y -q mecab \
                          libmecab-dev \
                          mecab-ipadic-utf8 && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
RUN cd /tmp && \
    git clone --depth 1 https://github.com/neologd/mecab-ipadic-neologd.git
 && \
    /tmp/mecab-ipadic-neologd/bin/install-mecab-ipadic-neologd -y -n -p /v
ar/lib/mecab/dic/neologd && \
    rm -rf /tmp/mecab-ipadic-neologd
RUN pip install --upgrade pip
RUN pip install mecab-python==0.996
```

To use this image on your Cloudera AI project, perform the following steps.

1. Build a new image with the Dockerfile.

```
docker build --network=host -t <company-registry>/user/cdsw-mecab:latest .
 -f Dockerfile
```

2. Push the image to your company's Docker registry.

```
docker push <your-company-registry>/user/cdsw-mecab:latest
```

3. Whitelist the image, <your-company-registry>/user/cdsw-mecab:latest. Only a site administrator can do this.

Go to Admin Engines and add <company-registry>/user/cdsw-mecab:latest to the list of whitelisted engine images.

Engine Images				
Description	Repository:Tag	Default	Actions	
Test LaTeX deps	latextest:latest	○	Edit	Deprecate
Base Image v1	docker.repository.cloudera.com/cdsw/engine:1	○	Edit	Deprecate
Base Image v2	docker.repository.cloudera.com/cdsw/engine:2	○	Edit	Deprecate
Base Image v3	docker.repository.cloudera.com/cdsw/engine:3	○	Edit	Deprecate
Base Image v4	docker.repository.cloudera.com/cdsw/engine:4	⦿	Edit	Deprecate
Custom Image - MeCab	<your-company-registry>/user/cdsw-mecab:latest		Add	

4. Ask a project administrator to set the new image as the default for your project. Go to the project Settings, click Engines, and select company-registry/user/cdsw-mecab:latest from the dropdown.
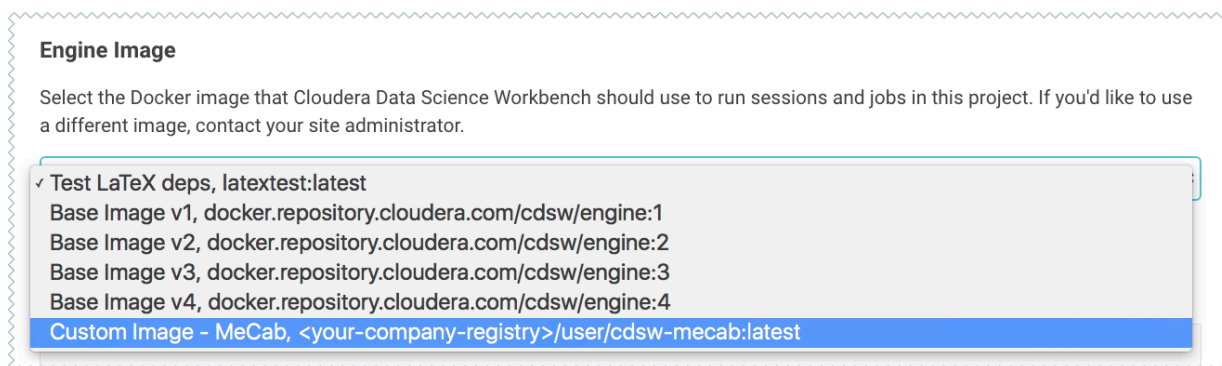
You shall now be able to run this project on the customized MeCab engine.

Configure a browser-based IDE at Legacy Engine level

You can make a browser-based Integrated Development Environment (IDE) available to any project within a Cloudera AI deployment by creating a customized legacy engine image, installing the editor to it, and adding it to the trusted list for a project. Additionally, browser IDEs that require root permission to install, such as RStudio, can only be used as part of a customized legacy engine image.

About this task

When a user launches a session, they can select the customized legacy engine with the editors available. The following steps describe how to make a customized legacy engine image for RStudio:

Procedure

1. Create a Dockerfile for the new custom image. Note that the Base Engine Image uses Ubuntu, and you must use Base Engine Image v9 or higher.

The following sample Dockerfile is for RStudio:

```
#Dockerfile
FROM docker.repository.cloudera.com/cdsw/engine:9-cml1.1

WORKDIR /tmp

#Delete the Cloudera repository that is inaccessible because of the payw
all
RUN rm /etc/apt/sources.list.d/*

#The RUN commands that install an editor
#For example: RUN apt-get install myeditor

RUN apt-get update && apt-get dist-upgrade -y && \
    apt-get install -y --no-install-recommends \
    libapparmor1 \
    libclang-dev \
    lsb-release \
    psmisc \
    sudo

#The command that follows RUN is the same command you used to install the
 IDE to test it in a the session.
```

```
RUN wget https://download2.rstudio.org/server/trusty/amd64/rstudio-server-
1.2.1335-amd64.deb && \
    dpkg -i rstudio-server-1.2.1335-amd64.deb

COPY rserver.conf /etc/rstudio/rserver.conf

COPY rstudio-cdsw /usr/local/bin/rstudio-cdsw

RUN chmod +x /usr/local/bin/rstudio-cdsw
```

2. Create rserver.conf:

```
# Must match CDSW_APP_PORT
www-port=8090
server-app-armor-enabled=0
server-daemonize=0
www-address=127.0.0.1
auth-none=1
auth-validate-users=0
```

Make sure that the www-port property matches the port set in the CDSW_APP_PORT environment variable (default 8090).

3. Create rstudio-cdsw:

```
#!/bin/bash
# This saves RStudio's user runtime information to /tmp, which ensures
several
# RStudio sessions can run in the same project simultaneously
mkdir -p /tmp/rstudio/sessions/active
mkdir -p /home/cdsw/.rstudio/sessions
if [ -d /home/cdsw/.rstudio/sessions/active ]; then rm -rf /home/cdsw/.rst
udio/sessions/active; fi
ln -s /tmp/rstudio/sessions/active /home/cdsw/.rstudio/sessions/active
# This ensures RStudio picks up the environment. This may not be necess
ary if
# you are installing RStudio Professional. See
# https://docs.rstudio.com/ide/server-pro/r-sessions.html#customizing-sess
ion-launches.
# SPARK_DIST_CLASSPATH is treated as a special case to workaround a bug in
 R
# with very long environment variables.
env | grep -v ^SPARK_DIST_CLASSPATH >> /usr/local/lib/R/etc/Renviron.site
echo "Sys.setenv(\"SPARK_DIST_CLASSPATH\"=\"${SPARK_DIST_CLASSPATH}\")"
 >> /usr/local/lib/R/etc/Rprofile.site

# Now start RStudio
/usr/sbin/rstudio-server start
```

4. Build the Dockerfile:

```
docker build -t <image-name>:<tag> . -f Dockerfile
```

If you want to build your image on a Cloudera AI Workbench, you must add the --network=host option to the build command:

```
docker build --network=host -t <image-name>:<tag> . -f Dockerfile
```

5. Distribute the image:

- Push the image to a public registry such as DockerHub.

  For instructions, refer the Docker documentation.
- Push the image to your company's Docker registry.

  When using this method, make sure to tag your image with the following schema:

  ```
  docker tag <image-name> <company-registry>/<user-name>/<image-name>:
  <tag>
  ```

  Once the image has been tagged properly, use the following command to push the image:

  ```
  docker push <company-registry>/<user-name>/<image-name>:<tag>
  ```

6. Add the image to the trusted list in Cloudera AI:

- a) Log in to the Cloudera AI web UI as a site administrator.
- b) Click Admin Engines .
- c) Add <company-registry>/<user-name>/<image-name>:<tag> to the list of trusted engine images.

7. Add the new legacy engine to the trusted list for a project:

- a) Go to the project Settings page.
- b) Click Engines.
- c) Select the new customized legacy engine from the dropdown list of available Docker images. Sessions and jobs you run in your project will have access to this engine.

8. Configure RStudio for the project. When this is done, you will be able to select RStudio from the dropdown list of editors on the Launch New Sesssion page.

- a) Go to Settings > Editors and click New Editor.
- b) Complete the fields:

  - Name: Provide a name for the editor. For example, RStudio. 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.

    For example, the following command will start RStudio:

    ```
    /usr/local/bin/rstudio-cdsw
    ```

- c) Save the changes.

Pre-Installed Packages in engines

Cloudera AI ships with several base engine images that include Python and R kernels, and frequently used libraries.

Base Engine 15-cml-2021.09-1

Engine 15 ships Python versions 2.7.18 and 3.6.13, and R version 3.6.3.

Items in bold indicate a new version since the last release.

Table 1: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0

Library	Version
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
matplotlib	2.2.4
seaborn	0.9.0
cython	0.29.13
six	1.16.0

Table 2: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.2.4
seaborn	0.9.0
cython	0.29.13
six	1.16.0

Table 3: R Libraries

Package	Version
RCurl	1.98.1.2
caTools	1.18.0
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.4
ggplot2	3.3.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.69
dplyr	1.0.0
httr	1.4.1
httpuv	1.5.4
jsonlite	1.6.1

Package	Version
magrittr	1.5
knitr	1.28
purrr	0.3.4
tm	0.7.7
proxy	0.4.24
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.12
devtools	2.3.0

Base Engine 14-cml-2021.05-1

Engine 14 ships Python versions 2.7.18 and 3.6.10, and R version 3.6.3.

Items in bold indicate a new version since the last release.

Table 4: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
matplotlib	3.1.2
seaborn	0.9.0
cython	0.29.13
six	1.15.0

Table 5: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.10
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0

Library	Version
matplotlib	2.2.4
seaborn	0.9.0
cython	0.29.13
six	1.15.0

Table 6: R Libraries

Package	Version
RCurl	1.98.1.2
caTools	1.18.0
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.4
ggplot2	3.3.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.69
dplyr	1.0.0
httr	1.4.1
httpuv	1.5.4
jsonlite	1.6.1
magrittr	1.5
knitr	1.28
purrr	0.3.4
tm	0.7.7
proxy	0.4.24
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.12
devtools	2.3.0

Related Information

Base Engine 9
Base Engine 10
Base Engine 11
Base Engine 12
Base Engine 13

Base Engine 13-cml-2020.08-1

Engine 13 ships Python versions 2.7.18 and 3.6.10, and R version 3.6.3.

Items in bold indicate a new version since the last release.

Note: This is the only engine available on CML Private Cloud 1.0.

Table 7: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
matplotlib	3.1.2
seaborn	0.9.0
cython	0.29.13
six	1.15.0

Table 8: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.10
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.2.4
seaborn	0.9.0
cython	0.29.13
six	1.15.0

Table 9: R Libraries

Package	Version
RCurl	1.98.1.2
caTools	1.18.0
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.4
ggplot2	3.3.1

Package	Version
cluster	2.1.0
codetools	0.2.16
foreign	0.8.69
dplyr	1.0.0
httr	1.4.1
httpuv	1.5.4
jsonlite	1.6.1
magrittr	1.5
knitr	1.28
purrr	0.3.4
tm	0.7.7
proxy	0.4.24
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.12
devtools	2.3.0

Related Information

Base Engine 12-cml-2020.06-2

Engine 12 ships Python versions 2.7.18 and 3.6.10, and R version 3.6.3.

Table 10: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
matplotlib	3.1.2
seaborn	0.9.0
Cython	0.29.13

Table 11: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.2.4
seaborn	0.9.0
Cython	0.29.13

Table 12: R Libraries

Package	Version
RCurl	1.98.1.2
caTools	1.18.0
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.4
ggplot2	3.3.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.69
dplyr	1.0.0
httr	1.4.1
httpuv	1.5.4
jsonlite	1.6.1
magrittr	1.5
knitr	1.28
purrr	0.3.4
tm	0.7.7
proxy	0.4.24
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.12
devtools	2.3.0

Base Engine 11-cml1.4

Engine 11 ships Python versions 2.7.17 and 3.6.9, and R version 3.6.2.

Table 13: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1
py4j	0.10.8.1
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13

Table 14: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13

Table 15: R Libraries

Package	Version
RCurl	1.95.4.12
caTools	1.17.1.3

Package	Version
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.3
ggplot2	3.2.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.73
dplyr	0.8.3
httr	1.4.1
httpuv	1.5.2
jsonlite	1.6
magrittr	1.5
knitr	1.26
purrr	0.3.3
tm	0.7.7
proxy	0.4.23
data.table	1.12.8
stringr	1.4.0
Rook	1.1.1
rJava	0.9.11
devtools	2.2.1

Related Information

Base Engine 9
Base Engine 10
Base Engine 12
Base Engine 13
Base Engine 14

Base Engine 10-cml1.3

Engine 10 ships Python versions 2.7.17 and 3.6.9, and R version 3.5.1.

Table 16: Python 3 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.17.2
pandas	0.25.1
pandas-datareader	0.8.1

Library	Version
py4j	0.10.8.1
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13

Table 17: Python 2 Libraries

Library	Version
ipython	5.1.0
requests	2.22.0
simplejson	3.16.0
numpy	1.16.5
pandas	0.24.2
pandas-datareader	0.8.0
py4j	0.10.8.1
futures	3.3.0
matplotlib	2.0.0
seaborn	0.9.0
Cython	0.29.13

Table 18: R Libraries

Package	Version
RCurl	1.95.4.12
caTools	1.17.1.3
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.3
ggplot2	3.2.1
cluster	2.1.0
codetools	0.2.16
foreign	0.8.73
dplyr	0.8.3
httr	1.4.1
httpuv	1.5.2
jsonlite	1.6
magrittr	1.5
knitr	1.26
purrr	0.3.3
tm	0.7.7
proxy	0.4.23
data.table	1.12.8

Package	Version
stringr	1.4.0
Rook	1.1.1
rJava	0.9.11
devtools	2.2.1

Related Information

Base Engine 9
Base Engine 11
Base Engine 12
Base Engine 13
Base Engine 14

Base Engine 9-cml1.2

Engine 9 ships Python 2.7.11 and 3.6.8, and R version 3.5.1.

Table 19: Python Libraries

Library	Version
ipython	5.1.0
requests	2.13.0
Flask	0.12.0
simplejson	3.10.0
numpy	1.13.3
pandas	0.20.1
pandas-datareader	0.2.1
py4j	0.10.7
futures	2.1.4
matplotlib	2.0.0
seaborn	0.8.0
Cython	0.25.2
kudu-python	1.2.0

Table 20: R Libraries

Package	Version
RCurl	1.95.4.12
caTools	1.17.1.2
svTools	0.9.5
png	0.1.7
RJSONIO	1.3.1.2
ggplot2	3.1.1
cluster	2.0.9
codetools	0.2.16

Package	Version
foreign	0.8.71
dplyr	0.8.1
httr	1.4.0
httpuv	1.5.1
jsonlite	1.6
magrittr	1.5
knitr	1.23
purrr	0.3.2
tm	0.7.6
proxy	0.4.23
data.table	1.12.2
stringr	1.4.0
Rook	1.1.1
rJava	0.9.11
devtools	2.0.2

Related Information

Base Engine 10
Base Engine 11
Base Engine 12
Base Engine 13
Base Engine 14