Cloudera Data Science Workbench

Customized Engine Images

Date published: 2020-02-28
Date modified:

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

Customized Engine Images....................4

Creating a Customized Engine Image....................4

Create a Dockerfile for the New Custom Image....................4

Configure a Browser IDE at the Legacy Engine Level....................4

Add Docker Registry Credentials....................8

Build the New Image....................9

Distribute the Image....................9

Whitelist the Image in Cloudera Data Science Workbench....................9

End-to-End Example: MeCab....................10

Engine Limitations....................11

Related Resources....................11

Customized Engine Images

By default, Cloudera Data Science Workbench engines are preloaded with a few common packages and libraries for R, Python, and Scala. In addition to these, Cloudera Data Science Workbench 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 Data Science Workbench 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 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).

Creating a Customized Engine Image

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

For a complete example, see End Example: MeCab.

Create a Dockerfile for the New 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.

When creating the Dockerfile you must delete the Cloudera repository that is inaccessible because of the paywall by running the following:

```
RUN rm /etc/apt/sources.list.d/*
```

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

```
# Dockerfile

# Specify a Cloudera Data Science Workbench base image
FROM docker.repository.cloudera.com/cdsw/engine:8
RUN rm /etc/apt/sources.list.d/*
# Update packages on the base image and install beautifulsoup4
RUN apt-get update
RUN pip install beautifulsoup4 && pip3 install beautifulsoup4
```

Configure a Browser IDE at the Legacy Engine Level

You can make a browser IDE available to any project within a Cloudera Data Science Workbench deployment by creating a customized legacy engine image, installing the editor to it, and then whitelisting the custom image for projects as needed. Additionally, browser IDEs that require root permission to install, such as RStudio, can only be used as part of a customized legacy engine image.

Before you begin

Note: The following steps are only required if you want to use an editor that does not come pre-installed as part of the default engine image that Cloudera Data Science Workbench ships with.

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 build 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.

   The following sample Dockerfile is for RStudio:

```
FROM docker.repository.cloudera.com/cdsw/engine:10

WORKDIR /tmp

#Delete the Cloudera repository that is inaccessible because of the paywal
l

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 install -y --no-install-recommends \
    libapparmor1 \
    libclang-dev \
    lsb-release \
    psmisc \
    sudo && \
  apt-get clean && \
  apt-get autoremove && \
  rm -rf /var/lib/apt/lists/*

RUN wget --quiet https://download2.rstudio.org/server/bionic/amd64/rstudio
-server-1.2.5033-amd64.deb && \
    dpkg -i rstudio-server-1.2.5033-amd64.deb && \
    rm rstudio-server-1.2.5033-amd64.deb
COPY rserver.conf /etc/rstudio/rserver.conf

COPY rstudio-cdsw /usr/local/bin/rstudio-cdsw
RUN chmod 777 /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 Data Science Workbench gateway host, you must add the --networ k=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: docker push.
- 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>
  ```
- Distribute the image manually:

  a. Save the docker image as a tarball on the host where it was built

  ```
  docker image save -o ./<new_customized_engine>.tar <image-name>
  ```

  b. Distribute the image to all the Cloudera Data Science Workbench gateway hosts.

  ```
  scp ./<new_customized_engine>.tar root@<cdsw.your_company.com>:/tmp/
  ```

  c. Load the image on all the Cloudera Data Science Workbench gateway hosts.

  ```
  docker load --input /tmp/./<new_customized_engine>.tar
  ```

  d. To verify that the image was successfully distributed and loaded, run:

  ```
  docker images
  ```

6. Whitelist the image in Cloudera Data Science Workbench:

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

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

a) Go to the project Settings page.
b) Click Engines.
c) Select the new legacy engine from the dropdown list of available Docker images. This engine will now be used to launch sessions within this project.

8. Configure project(s) to use RStudio. When this is done, you will be able to select RStudio from the dropdown list of editors on the Launch New Session page. There are two ways to do this: for an individual project, or for all projects that use this engine.

Configure RStudio for an individual project

a) Go to the project Settings Editors .
b) Click New Editor.
c) 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
  ```

d) Click Save.

Configure RStudio for all projects that use this engine

a) Log in to the Cloudera Data Science Workbench web UI as a site administrator.
b) Click Admin Engines .
c) Under Engine Images, click the Edit button for the engine image that you whitelisted here in a previous step.
d) Click New Editor.

- 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
  ```

e) Click Save, then click Save again.

What to do next

For more information about how to create a customized engine image and limitations, see AWS Account Requirements

Add Docker Registry Credentials

To enable CDSW to fetch custom engines from a secure repository, as Administrator you need to add Docker registry credentials.

Procedure

1. Create a kubectl secret named regcred for your secured Docker registry.

The following command creates the secret in your Kubernetes cluster:

```
kubectl create secret docker-registry regcred
--docker-server=<server host> --docker-username=<username>
--docker-password=<password> -n <compute namespace eg. default>
```

2. Retrieve the above secret as a yaml file by entering the following:

```
kubectl get secret regcred --output=yaml -n
<compute namespace eg. default> > k8s-secret-regcred.yaml
```

3. Place the yaml file in the master node located at /etc/cdsw/patches.

   Set the file visibility for root only.
4. Restart CDSW service to pick up the new secret.
5. Launch a Session>Model>Job>Experiment using the engine image from the secured Docker registry.

Build the New 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 your Dockerfile makes any outside connection (for example, apt-get update, pip install, curl), you must add the --network=host option to the build command:

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

Distribute the Image

Once you have built a new custom engine, use one of the following ways to distribute the new image to all your Cloudera Data Science Workbench hosts.

Push the Image to a Public Registry such as DockerHub

Once you have built a new custom engine, you can push the image to a public registry such as Dockerhub.

About this task

For instructions, refer the Docker documentation: docker push.

Push the Image to Your Company's Docker Registry

Once you have built a new custom engine, you can push the image to your company's Docker registry.

About this task

Procedure

1. 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>
   ```
2. 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.

Whitelist the Image in Cloudera Data Science Workbench

White-listing a customized image in Cloudera Data Science Workbench is a two-step process.

Procedure

1. If you want to include an image in the allowlist for deployment, the site administrator must 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 list of whitelisted engine images.

2. If you want to start using the image in a project, the project administrator must 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 dropdown list of available Docker images. Sessions and jobs you run in your project will now have access to this engine.

End-to-End Example: MeCab

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

About this task

This is a sample Dockerfile that adds MeCab to the Cloudera Data Science Workbench base image.

```
# Dockerfile

FROM docker.repository.cloudera.com/cdsw/engine:8
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 Data Science Workbench project, perform the following steps.

Procedure

1. Build a new image with the Dockerfile.

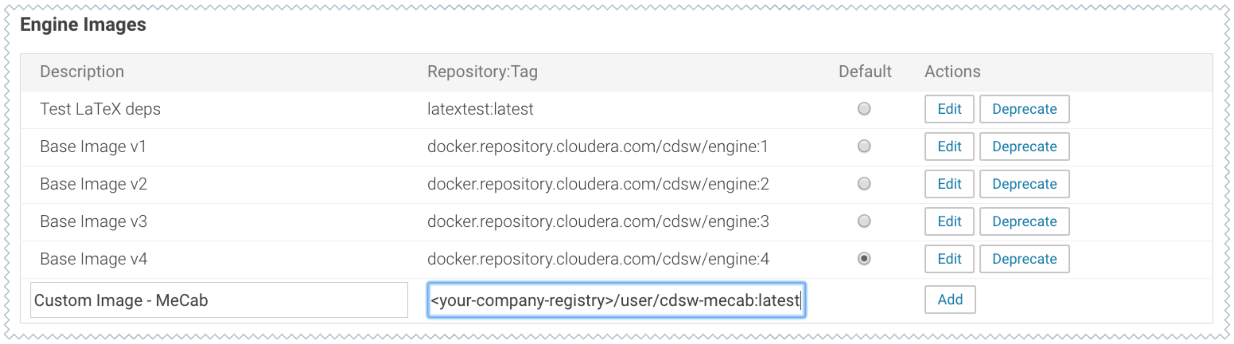
```
docker build --network=host -t <company-registry>/user/cdsw-mecab:latest .
 -f Dockerfile
```

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.

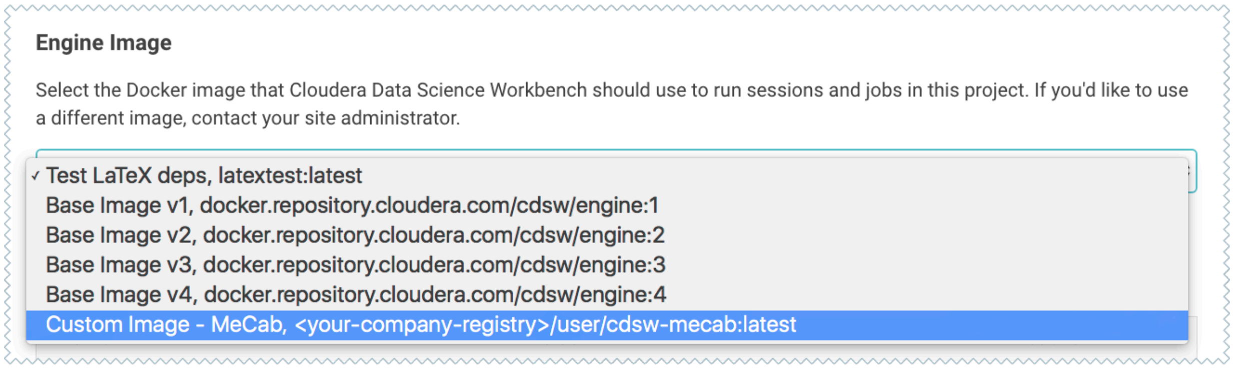


Engine Images

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.



Engine Image

Select the Docker image that Cloudera Data Science Workbench should use to run sessions and jobs in this project. If you'd like to use a different image, contact your site administrator.

✓ Test LaTeX deps, latextest:latest
Base Image v1, docker.repository.cloudera.com/cdsw/engine:1
Base Image v2, docker.repository.cloudera.com/cdsw/engine:2
Base Image v3, docker.repository.cloudera.com/cdsw/engine:3
Base Image v4, docker.repository.cloudera.com/cdsw/engine:4
Custom Image - MeCab, <your-company-registry>/user/cdsw-mecab:latest

   You should now be able to run this project on the customized MeCab engine.

Engine Limitations

This topic lists all the limitations for customized Engines images.

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

  Cloudera Bug: DSE-4420
- The contents of certain pre-existing standard directories such as /home/cdsw, /tmp, /opt/cloudera, 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. Or, create a new custom directory on all the Cloudera Data Science Workbench gateway hosts and save your files to those directories. Then, mount this directory to the custom engine.

Related Resources

This topic contains related resources for monitoring workbench activity.

- Models - Monitoring All Active Models.
- Tracking Disk Usage - Tracking Disk Usage on the Application Block Device