

Using Custom Spark Runtime Docker Images via API/CLI (Preview)

Date published: 2022-09-06

Date modified: 2022-09-06

Legal Notice

© Cloudera Inc. 2022. 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

| | |
|---------------------|----------|
| Legal Notice | 2 |
| Contents | 3 |

This is a detailed user guide that demonstrates how to run Spark jobs using custom Spark runtime Docker images via API/CLI. Custom Spark runtime Docker images are used when custom packages and libraries need to be installed and used when executing Spark jobs. These custom packages and libraries can be proprietary software packages like RPMs that need to be compiled to generate the required binaries. Docker images allow you to pre-bake these dependencies into a self-contained Docker file that can be used across multiple Spark jobs.

Steps

1. Create a custom Docker image

Build the “custom-spark-dex-runtime” images based on the “dex-spark-runtime” image of the Cloudera Data Engineering (CDE) version.

The image should be based on the dex-spark-runtime of the current dex version.

The relevant dex-spark-runtime image is

```
<registry-host>/cloudera/dex/dex-spark-runtime-<spark version>-<cdh version>:<dex version>
```

Example: DockerFile for DEX 1.15.0-b117, Spark 2.4.8 and CDP Runtime version 7.2.14.0

FROM

```
docker-private.infra.cloudera.com/cloudera/dex/dex-spark-runtime-2.4.8-7.2.14.0-7.2.14.0:1.15.0-b117
```

USER root

```
RUN yum install -y git && yum clean all && rm -rf /var/cache/yum
```

```
RUN pip2 install virtualenv-api
```

```
RUN pip3 install virtualenv-api
```

```
USER ${DEX_UID}
```

2. Build the docker image by tagging it with the custom registry and push it to the custom registry.

Example:

```
mac@local:$ docker build --network=host -t
```

```
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8-7.2.14.0:1.15.0-b117
```

```
-custom . -f Dockerfile
```

```
mac@local:$ docker push
```

```
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8-7.2.14.0:1.15.0-b117
```

```
-custom
```

Here, the custom registry is “docker.my-company.registry.com” and the registry namespace is “custom-dex”.

Note

Obtain `$CDE_TOKEN` to execute the REST API examples by following [Getting a Cloudera Data Engineering API access token](#) document.

3. Create the credentials for the custom image registry.
Register docker registry image pull credentials using the CDE CLI or REST API.
These credentials are stored as a secret.

CLI

```
mac@local:~$ ./cde credential create --name docker-creds --type docker-basic --docker-server  
docker-sandbox.infra.cloudera.com --docker-username my-username
```

REST API

```
curl -X POST -k 'https://<dex-vc-host>/dex/api/v1/credentials' \  
-H "Authorization: Bearer ${CDE_TOKEN}" \  
-H 'accept: application/json' \  
-H 'Content-Type: application/json' \  
--data '{  
  "dockerBasic": {  
    "password": "password123",  
    "server": "docker-sandbox.infra.cloudera.com",  
    "username": "my-username"  
  },  
  "name": "docker-creds",  
  "type": "docker-basic"  
}'
```

Optional: If using a public docker registry, complete these steps.

Create a resource for the registries which do not require any authentication.
There is no need to specify the credentials.

CLI

```
mac@local:~$ cde resource create --name custom-image-resource --image  
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8-7.2.14.0:1.15.0-b117-custo  
m --image-engine spark2 --type custom-runtime-image
```

REST API

```
ccurl -X POST -k 'https://<dex-vc-host>/dex/api/v1/resources \  
'
```

```

-H "Authorization: Bearer ${CDE_TOKEN}" \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
--data '{
"customRuntimeImage": {
  "engine": "spark2",
  "image":
"docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8-7.2.14.0:1.15.0-b117
-custom"
},
"name": "custom-image-resource",
"type": "custom-runtime-image"
}'

```

Once done, skip to [step 5](#) to submit the job.

4. Create the “custom-runtime-image” resource by referring to the credential created previously.

Register “custom-spark-dex-runtime” docker image as a resource of type “custom-runtime-image” by specifying the name of the credential created in the previous step.

CLI

```

mac@local:$ ./cde resource create --name custom-image-resource --image
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8-7.2.14.0:1.15.0-b117
-custom --image-engine spark2 --type custom-runtime-image --image-credential docker-creds

```

REST API

```

curl -X POST -k 'https://<dex-vc-host>/dex/api/v1/resources \
-H "Authorization: Bearer ${CDE_TOKEN}" \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
--data '{
"customRuntimeImage": {
  "credential": "docker-creds",
  "engine": "spark2",
  "image":
"docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8-7.2.14.0:1.15.0-b117
-custom"
},
"name": "custom-image-resource",
"type": "custom-runtime-image"
}'

```

5. Submit a job by setting the “custom-spark-dex-runtime” image as a resource using the CDE CLI.

SPARK COMMAND

```
mac@local:~$ ./cde --user cdpuser1 spark submit /Users/my-username/spark-examples_2.11-2.4.4.jar
--class org.apache.spark.examples.SparkPi 1000
--runtime-image-resource-name=custom-image-resource
```

JOB COMMAND

```
mac@local:~$ ./cde --user cdpuser1 resource create --name spark-jar
mac@local:~$ ./cde --user cdpuser1 resource upload --name spark-jar --local-path
spark-examples_2.11-2.4.4.jar
```

```
mac@local:~$ ./cde --user cdpuser1 job create --name spark-pi-job-cli --type spark --mount-1-resource
spark-jar --application-file spark-examples_2.11-2.4.4.jar --class org.apache.spark.examples.SparkPi
--user cdpuser1 --arg 22 --runtime-image-resource-name custom-image-resource
```

6. The spark driver/executor pods should use this image and you can confirm it by opening a shell into those pods and verifying if the external installed libraries or files exist.

Error: Custom image resource with missing or wrong credentials

Creating a custom image resource with missing or wrong credentials should result in the below error which can be seen in the logs or in Kubernetes pod events.

Example

Failed to pull image

```
"docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8-7.2.14.0:1.15.0-b117-custom":
rpc error: code = Unknown desc = Error reading manifest 1.15.0-b117-custom in
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.8-7.2.14.0:errors: denied: requested
access to the resource is denied unauthorized: authentication required
```