Cloudera Data Engineering 1.15.2

Using Cloudera Data Engineering resources

Date published: 2020-07-30
Date modified: 2022-09-14

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. 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

Using Python virtual environments with Cloudera Data Engineering

Cloudera Data Engineering (CDE) supports Python virtual environments to manage job dependencies by using the python-env *resource* type.

A *resource* in CDE is a named collection of files or other resources referenced by a job. The python-env resource type allows you to specify a requirements.txt file that defines a virtual environment that you can then associate with a CDE job. You can specify any Python package in the requirements.txt file, including those with C dependencies.

Creating a Python virtual environment resource

After you have created the requirements.txt file, you can create the Python virtual environment resource.

Note: For python-env resources, you can only upload a requirements.txt file. Python environment resources do not support arbitrary file upload. If the local file is named something other than requirements.txt, you must add the flag --resource-path requirements.txt to the command, which renames the file to requirements.txt in the resource.

You can also specify a PyPi mirror for a Python virtual environment resource using the --pypi-mirror flag. This requires network access to the mirror from the CDP environment.

For CDE CLI

Before you begin

- Download and configure the CDE CLI.
- Create a requirements.txt file specifying the Python package and version dependencies required by your CDE job.

Steps

1. Run the cde resource create command as follows to create a Python virtual environment resource.

   Note: You can also specify a PyPi mirror for a Python virtual environment resource using the --pypi-mirror flag. This requires network access to the mirror from the CDP environment.

   ```
   cde resource create --name cde-python-env-resource --type python-env --p
   ython-version python3
   ```

2. Upload the requirements.txt file to the resource.

   Note: For python-env resources, you can only upload a requirements.txt file. Python environment resources do not support arbitrary file upload. If the local file is named something other than requirem ents.txt, you must add the flag --resource-path requirements.txt to the command, which renames the file to requirements.txt in the resource.

   ```
   cde resource upload --name cde-python-env-resource --local-path ${HOME}/
   requirements.txt
   ```

Result

When you first create a Python virtual environment resource, CDE builds the environment according to the requ irements.txt file. During this build time, you cannot run a job associated with the virtual environment. You can check the status of the environment by running cde resource list-events --name *<RESOURCE_NAME>*. For example:

```
cde resource list-events --name cde-python-env-resource
```

The environment is ready when you see a message similar to the following:

```
   {
     "id": 4,
     "message": "Job pp-84kgdgf6-resource-builder-cde-python-env-resourc
e-1634911572 succeeded, marking resource with ready status",
     "created": "2021-10-22T14:09:13Z"
   }
```

For Web UI

Before you begin

- Create a requirements.txt file specifying the Python package and version dependencies required by your CDE job.

Steps

1. Go to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera Data Platform (CDP) management console.
2. In the CDE Services column, select the service containing the virtual cluster where you want to create the Python virtual environment.
3. In the Virtual Clusters column on the right, click the View Jobs icon for the virtual cluster where you want to create the Python virtual environment.
4. Click Resources in the left menu.
5. Click Create Resource at the top right.
6. Specify a resource name, and then select Python Environment from the Type drop-down menu.
7. Choose the Python version for the environment and optionally specify the PyPi Mirror URL. The PyPi mirror must be accessible from the CDP environment.
8. Click Create.
9. Click Upload File and select the requirements.txt file from your local machine. You can also drag-and-drop the file to the outlined area on the page.

Result

The UI displays Building the resource... while the Python virtual environment is building. After the environment is built, the page displays the Python packages and versions included in the environment.

Associating a Python virtual environment with a Cloudera Data Engineering job

You can associate the Python virtual environment with a CDE job at the time of creation, or you can update an existing job.

For CDE CLI

Before you begin

- Download and configure the CDE CLI.
- Create a Python virtual environment CDE resource.
- Create a CDE job.

Steps

1. Using the CDE CLI, run the cde job update command to associate a Python virtual environment with the job.

   ```
   cde job update --name pyspark-example --python-env-resource-name cde-pyt
   hon-env-resource
   ```

   Note: You can specify a Python virtual environment resource at job creation time as well, using the flag --python-env-resource-name. For example:

   ```
   cde job create --type spark --application-file pyspark-example.p
   y --python-env-resource-name cde-python-env-resource --name pysp
   ark-example
   ```

For Web UI

Before you begin

- Create a Python virtual environment CDE resource.
- Create a CDE job.

Steps

1. Go to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera Data Platform (CDP) management console.
2. In the CDE Services column, select the service containing the virtual cluster and job you want to configure.
3. In the Virtual Clusters column on the right, click the View Jobs icon for the virtual cluster containing the job you want to configure.
4. Click Jobs in the left menu.
5. Click the job you want to modify.
6. Go to the Configuration tab.
7. Click Edit.
8. In the Python Environment section, click Select Python Environment.
9. Select the Python virtual environment resource you want to use, and then click Select Resource.
10. At the bottom of the page, click Update and Run to run the job immediately, or click the drop-down arrow on the button and select Update to update the job without running it.

Updating Python virtual environment resources

Currently, Python virtual environments cannot be updated. Instead, create a new Python virtual environment resource and update the job to reference the new resource.

For CDE CLI

Before you begin

- Download and configure the CDE CLI.
- Create a new requirements.txt file specifying the Python package and version dependencies required by your CDE job.

Steps

1. Create a new python-env resource.

   ```
   cde resource create --name new-cde-python-env-resource --type python-env
    --python-version python3
   ```

2. Upload the new or updated requirements.txt file to the new resource.

```
cde resource upload --name new-cde-python-env-resource --local-path ${HO
ME}/requirements.txt
```

3. Update the CDE job to specify the new resource.

```
cde job update --name pyspark-example --python-env-resource-name new-cde
-python-env-resource
```

For Web UI

Before you begin

- Create a new requirements.txt file specifying the Python package and version dependencies required by your CDE job.

Steps

1. Create a new Python virtual environment using the new requirements.txt file, following the instructions in Creating a Python virtual environment resource.
2. Update the CDE job to reference the new Python virtual environment, following the instructions in Associating a Python virtual environment with a CDE job.

Related Information

Creating a Python virtual environment resource

Associating a Python virtual environment with a Cloudera Data Engineering job

Using Custom Spark Runtime Docker Images Via API/CLI

This is a detailed usage guide to demonstrate how to run jobs using custom spark runtime with examples.

Steps

1. Create a custom docker image

   Build “custom-spark-dex-runtime” images based on the dex-spark-runtime image of the CDE version.

   Note: 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 vers
ion>:<dex version>
```

   Example: DockerFile for DEX 1.15.1-b22, spark 2.4.7 and CDH version 7.1.7.74

```
DockerFile for DEX 1.15.1-b22, spark 2.4.7 and CDH version 7.1.7.74
FROM docker-private.infra.cloudera.com/cloudera/dex/dex-spark-runtime-2.
4.7-7.1.7.74-7.1.7.74:1.15.1-b22
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 tagging it with the custom registry to be used 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-2.4.7-7.1.7.74:1.15.1-b22-custom . -f Dockerf
ile
mac@local:$ docker push docker.my-company.registry.com/custom-dex/dex-sp
ark-runtime-2.4.7-7.1.7.74:1.15.1-b22-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.

   Note: If using public registry skip to this step

   For CLI

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

   For 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"
}’
```

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

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

   For CLI

```
mac@local:$ ./cde resource create --name custom-image-resource --image
docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.7-7.1.7.
74:1.15.1-b22-custom --image-engine spark2 --type custom-runtime-image -
-image-credential docker-creds
```

   For 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-r
untime-2.4.7-7.1.7.74:1.15.1-b22-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 CLI

For 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
```

For 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-exampl
es_2.11-2.4.4.jar --class org.apache.spark.examples.SparkPi --user cdpus
er1 --arg 22 --runtime-image-resource-name custom-image-resource
```

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

Public docker registries

Create the resource for the registries which do not require any auth. You do not need to specify the credentials.

For CLI

```
mac@local:$ cde resource create --name custom-image-resource --image doc
ker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.7-7.1.7.74:
1.15.1-b22-custom --image-engine spark2 --type custom-runtime-image
```

For 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-ru
ntime-2.4.7-7.1.7.74:1.15.1-b22-custom"
  },
  "name": "custom-image-resource",
   "type": "custom-runtime-image"
}’
```

Once done, skip to #step 5 to submit the job.

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-ru
ntime-2.4.7-7.1.7.74:1.15.1-b22-custom":
rpc error: code = Unknown desc = Error reading manifest 1.15.1-b22-custom in
 docker.my-company.registry.com/custom-dex/dex-spark-runtime-2.4.7-7.1.7.7
4:
errors: denied: requested access to the resource is denied unauthorized: aut
hentication required
```