Cloudera Data Engineering 1.4.1

Managing Cloudera Data Engineering jobs

Date published: 2020-07-30
Date modified: 2022-11-18

https://docs.cloudera.com/

Legal Notice

Contents

Configure users to create jobs

You must complete the manual steps to prepare the cluster for each user that needs to submit jobs. You can also configure a service account Kerberos key tab file to a machine user id in order to submit CDE jobs. The machine user will now have access to the hdfs storage through jobs.

Configuring LDAP users

You must complete the following manual steps to prepare the cluster for each user that needs to submit jobs. Perform these steps for each user that needs to submit jobs to the virtual cluster.

Before you begin

In Cloudera Data Engineering (CDE), jobs are associated with virtual clusters. Before you can create a job, you must create a virtual cluster that can run it. For more information, see Creating virtual clusters.

Procedure

1. If you already downloaded the utility script and uploaded it to an ECS or HDFS gateway cluster host as documented in Creating virtual clusters, you can skip to step 8.
2. Download cdp-cde-utils.sh to your local machine.
3. Create a directory to store the files, and change to that directory:

   ```
   mkdir -p /tmp/cde-1.3.4 && cd /tmp/cde-1.3.4
   ```
4. Embedded Container Service (ECS)

   Copy the extracted utility script (cdp-cde-utils.sh) to one of the Embedded Container Service (ECS) cluster hosts. To identify the ECS cluster hosts:

   a. Log in to the Cloudera Manager web interface.
   b. Go to Clusters Experience Cluster ECS Hosts .
   c. Copy the script to one of the listed hosts manually.

   Red Hat OpenShift Container Platform (OCP)

   Copy the extracted utility script (cdp-cde-utils.sh) and the OpenShift kubeconfig file to one of the HDFS service gateway hosts, and install the kubectl utility:

   a. Log in to the Cloudera Manager web interface.
   b. Go to Clusters Base Cluster HDFS Instances .
   c. Copy the script to one of the Gateway hosts manually.
   d. Copy the OCP kubeconfig file to the same host.
   e. On that host, install the kubectl utility following the instructions in the Kubernetes documentation.
5. On the cluster host that you copied the script to, set the script permissions to be executable:

   ```
   chmod +x /PATH/TO/cdp-cde-utils.sh
   ```

6. Identify the virtual cluster endpoint:

   a. In the Cloudera Manager web UI, go to the Experiences page, and then click Open CDP Private Cloud Experiences.
   b. Click the Data Engineering tile.
   c. Select the CDE service containing the virtual cluster you want to activate.
   d. Click Cluster Details.
   e. Click JOBS API URL to copy the URL to your clipboard.
   f. Paste the URL into a text editor to identify the endpoint host. For example, the URL is similar to the following:

      ```
      https://dfdj6kgx.cde-2cdxw5x5.ecs-demo.example.com/dex/api/v1
      ```

      The endpoint host is dfdj6kgx.cde-2cdxw5x5.ecs-demo.example.com.

7. On the ECS or HDFS gateway host, create a filename containing the user principal, and generate a keytab. If you do not have the ktutil utility, you might need to install the krb5-workstation package. The following example commands assume the user principal is psherman@EXAMPLE.COM

   a. Create a file named <*USERNAME*>.principal (for example, psherman.principal) containing the user principal:

      ```
      psherman@EXAMPLE.COM
      ```

   b. Generate a keytab named <*USERNAME*>.keytab for the user using ktutil:

      ```
      sudo ktutil
      ktutil:  addent -password -p psherman@EXAMPLE.COM -k 1 -e aes256-cts
      Password for psherman@EXAMPLE.COM:
      ktutil:  addent -password -p psherman@EXAMPLE.COM -k 2 -e aes128-cts
      Password for psherman@EXAMPLE.COM:
      ktutil:  wkt psherman.keytab
      ktutil:  q
      ```

8. Validate the keytab using klist and kinit:

   ```
   klist -ekt psherman.keytab
   Keytab name: FILE:psherman.keytab
   KVNO Timestamp           Principal
   ---- ------------------- ----------------------------------------------
   --------
      1 08/01/2021 10:29:47 psherman@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
      1 08/01/2021 10:29:47 psherman@EXAMPLE.COM (aes128-cts-hmac-sha1-96)

   kinit -kt psherman.keytab psherman@EXAMPLE.COM
   ```

   Make sure that the keytab is valid before continuing. If the kinit command fails, the user will not be able to run jobs in the virtual cluster. After verifying that the kinit command succeeds, you can destroy the Kerberos ticket by running kdestroy.

9. Use the cdp-cde-utils.sh script to copy the user keytab to the virtual cluster hosts:

   ```
   ./cdp-cde-utils.sh init-user-in-virtual-cluster -h <endpoint_host> -
   u <USER> -p <PRINCIPAL_FILE> -k <KEYTAB_FILE>
   ```

   For example, using the psherman user, for the dfdj6kgx.cde-2cdxw5x5.ecs-demo.example.com endpoint host:

   ```
   ./cdp-cde-utils.sh init-user-in-virtual-cluster -h dfdj6kgx.cde-2cdxw5x5
   .ecs-demo.example.com -u psherman -p psherman.principal -k psherman.keytab
   ```

10. Repeat these steps for all users that need to submit jobs to the virtual cluster.

Configuring service account key tab to the machine user

You can configure a service account Kerberos key tab file to a machine user id in order to submit CDE jobs. The machine user will now have access to the hdfs storage through jobs.

Procedure

1. *Create a CDP machine user*. For example, `mu_cde`.
2. *Generate access key* and secret key for the machine user and download the credential information.
3. On the ECS or HDFS gateway host, create a filename containing the user principal, and generate a keytab. If you do not have the `ktutil` utility, you might need to install the krb5-workstation package. The following example commands assume the user principal is `mu_cde@example.com`.
   a. Create a file named <username>.principal (for example, *MU_CDE.PRINCIPAL*) containing the user principal:
   ```
   mu_cde@example.com
   ```
   b. Generate a key tab file for the actual service account. For example, `svc_acc`. Name this keytab file as `<username>.keytab`, example: `mu_cde.keytab`. The generation of keytab can be done for the user using the `ktutil` command:
   ```
   sudo ktutil
   ktutil:  addent -password -p mu_cde@example.com -k 1 -e aes256-cts
   Password for pmu_cde@example.com:
   ktutil:  addent -password -p mu_cde@example.com -k 2 -e aes128-cts
   Password for mu_cde@example.com:
   ktutil:  wkt mu_cde.keytab

   ktutil:  q
   ```
4. Validate the keytab using `klist` and `kinit` commands:
   ```
   klist -ekt mu_cde.keytab

   Keytab name: FILE:mu_cde.keytab
   KVNO Timestamp           Principal
   ---- ------------------- -----------------------------------------------------
   -----
      1 08/01/2021 10:29:47 mu_cde@example.com (aes256-cts-hmac-sha1-96)
      1 08/01/2021 10:29:47 mu_cde@example.com (aes128-cts-hmac-sha1-96)
   kinit -kt mu_cde.keytab mu_cde@example.com
   ```
   Make sure that the keytab is valid before continuing. If the `kinit` command fails, the user will not be able to run jobs in the virtual cluster. After verifying that the `kinit` command succeeds, you can destroy the Kerberos ticket by running `kdestroy`.
5. Upload the keytab file for the user id as `mu_cde` and use the principal and keytab file of the actual service account (`svc_acc`).
   ```
   ./cdp-cde-utils.sh init-user-in-virtual-cluster -h <endpoint_host> -u <u
   ser> -p <principal_file> -k <keytab_file>
   ```
   For example, using the `mu-cde` user, for the dfdj6kgx.cde-2cdxw5x5.ecs-demo.example.com endpoint host:
   ```
   ./cdp-cde-utils.sh init-user-in-virtual-cluster -h dfdj6kgx.cde-2cdxw5x5
   .ecs-demo.example.com -u mu-cde -p mu-cde.principal -k mu-cde.keytab
   ```
6. Set the *Ranger authorization policy* for the `svc_acc account`.

Related Information

Ranger authorization policy

Creating a machine user in CDP

Generate Access Key

Creating jobs in Cloudera Data Engineering

A job in Cloudera Data Engineering (CDE) consists of defined configurations and resources (including application code). Jobs can be run on demand or scheduled.

Before you begin

Important: You must complete the manual steps to prepare the cluster for each user who need to submit jobs.

In Cloudera Data Engineering (CDE), jobs are associated with virtual clusters. Before you can create a job, you must create a virtual cluster that can run it. For more information, see Creating virtual clusters.

Procedure

1. Navigate to the Cloudera Data Engineering Overview page by clicking the Data Engineering tile in the Cloudera Data Platform (CDP) management console.
2. In the Environments column, select the environment containing the virtual cluster where you want to create the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster where you want to create the application.
4. In the left hand menu, click Jobs.
5. Click the Create Job button.
6. Provide the Job Details:
   a) Select Spark for the job type.
   b) Specify the Name.
   c) Select File or URL for your application file, and provide or specify the file. You can upload a new file or select a file from an existing resource.

      If you select the URL option and specify an Amazon AWS S3 URL, add the following configuration to the job:

      config_key: spark.hadoop.fs.s3a.delegation.token.binding

      config_value: org.apache.knox.gateway.cloud.idbroker.s3a.IDBDelegationTokenBinding
   d) If your application code is a JAR file, specify the Main Class.
   e) Specify arguments if required. You can click the Add Argument button to add multiple command arguments as necessary.
   f) Enter Configurations if needed. You can click the Add Configuration button to add multiple configuration parameters as necessary.
   g) If your application code is a Python file, select the Python Version, and optionally select a Python Environment.
7. Click Advanced Configurations to display more customizations, such as additional files, initial executors, executor range, driver and executor cores, and memory.

   By default, the executor range is set to match the range of CPU cores configured for the virtual cluster. This improves resource utilization and efficiency by allowing jobs to scale up to the maximum virtual cluster resources available, without manually tuning and optimizing the number of executors per job.

8. Click Schedule to display scheduling options.

   You can schedule the application to run periodically using the Basic controls or by specifying a Cron Expression.
9. Click Alerts and provide the email id to receive alerts. Click + to add more email IDs. Optionally, you can select when you want email alerts whether for job failures or missed job service-level agreements or both.

   Note: You must configure the Configure Email Alerting option while creating a virtual cluster to send your email alerts. For more information about configuring email alerts, see Creating virtual clusters.
10. If you provided a schedule, click Schedule to create the job. If you did not specify a schedule, and you do not want the job to run immediately, click the drop-down arrow on Create and Run and select Create. Otherwise, click Create and Run to run the job immediately.

Related Information

Generate Access Key

CDE example jobs and sample data

Cloudera Data Engineering provides a suite of example jobs that operate on example data to showcase its core capabilities and make the onboarding easier. The example jobs are a combination of Spark and Airflow jobs, which include scenarios such as reading and writing from object storage, running an Airflow DAG, and expanding on Python capabilities with custom virtual environments. Once loaded, these jobs can be run on demand or scheduled. The sample data will be loaded into the environment's default Data Lake location.

Before you begin

In Cloudera Data Engineering (CDE), jobs are associated with virtual clusters. Before you can create a job, you must create a virtual cluster that can run it. For more information, see Creating virtual clusters.

About this task

Below is the description of the different example jobs:

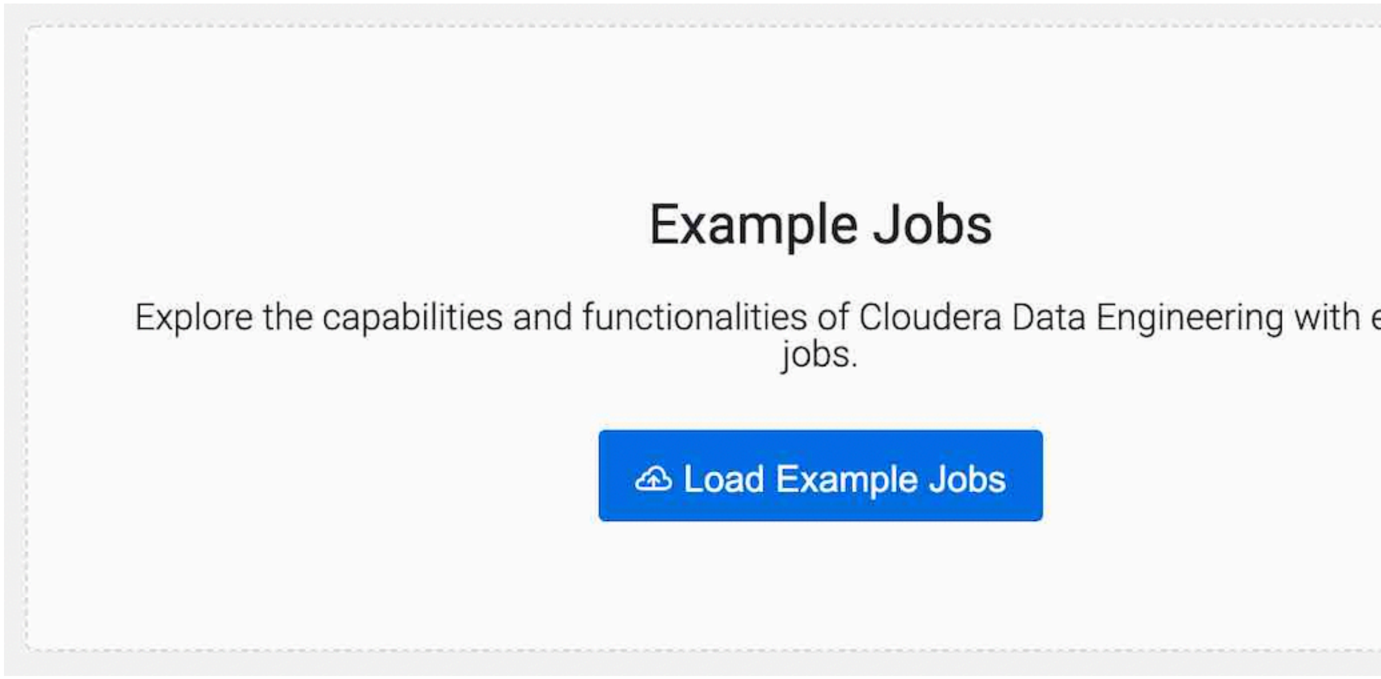
- example-load-data : this will load the sample data onto the environment data lake.

  Note: This will need to be run manually first if the sample jobs are loaded in any user defined virtual clusters.
- example-virtual-env: demonstrates CDE job configuration that utilizes Python Environment resource type to expand pyspark features via custom virtual env. This example adds pandas support.
- example-resources: demonstrates CDE job configuration utilizing file-based resource type. Resources are mounted on Spark driver and executor pods. This example uses an input file as a data source for a word-count Spark app.
- example-resources-schedules: demonstrates scheduling functionality for Spark job in CDE. This example schedules a job to run at 5:04am UTC each day.
- example-spark-pi: demonstrates how to define a CDE job. It runs a SparkPi using a scala example jar located on a s3 bucket.
- example-cdeoperator: demonstrates job orchestration using Airflow. This example uses a custom CDE Operator to run two Spark jobs in sequence, mimicking a pipeline composed of data ingestion and data processing.
- example-object-store: demonstrates how to access and write data to object store on different form factors: S3, ADLS, and HDFS. This example reads data already staged to object store and makes changes and then saves back the transformed data to object store.
- example-iceberg: demonstrates support for iceberg table format. This example reads raw data from object store and saves data in iceberg table format and showcases iceberg metadata info, such as snapshots.

Procedure

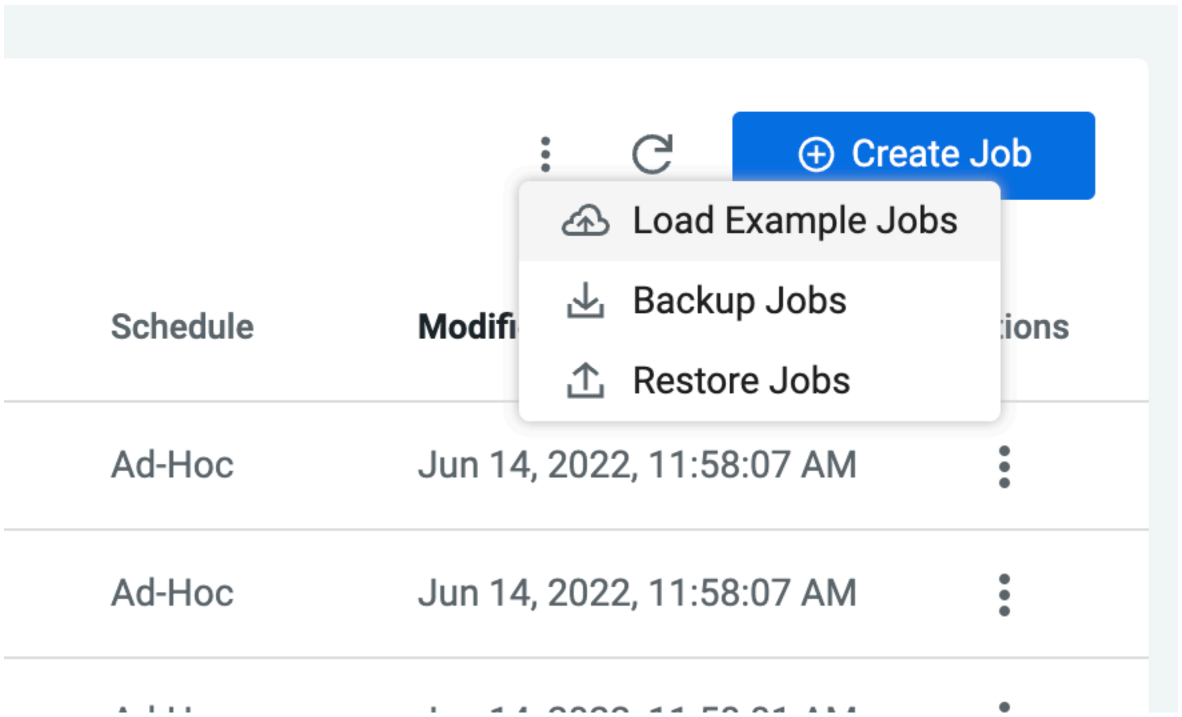
1. In the Cloudera Data Platform (CDP) management console, click the Data Engineering tile and click Overview.

2. In the CDE Services column, select the service containing the virtual cluster where you want to create the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster where you want to create the application.
4. Select Load Example Jobs from the two options that appear.

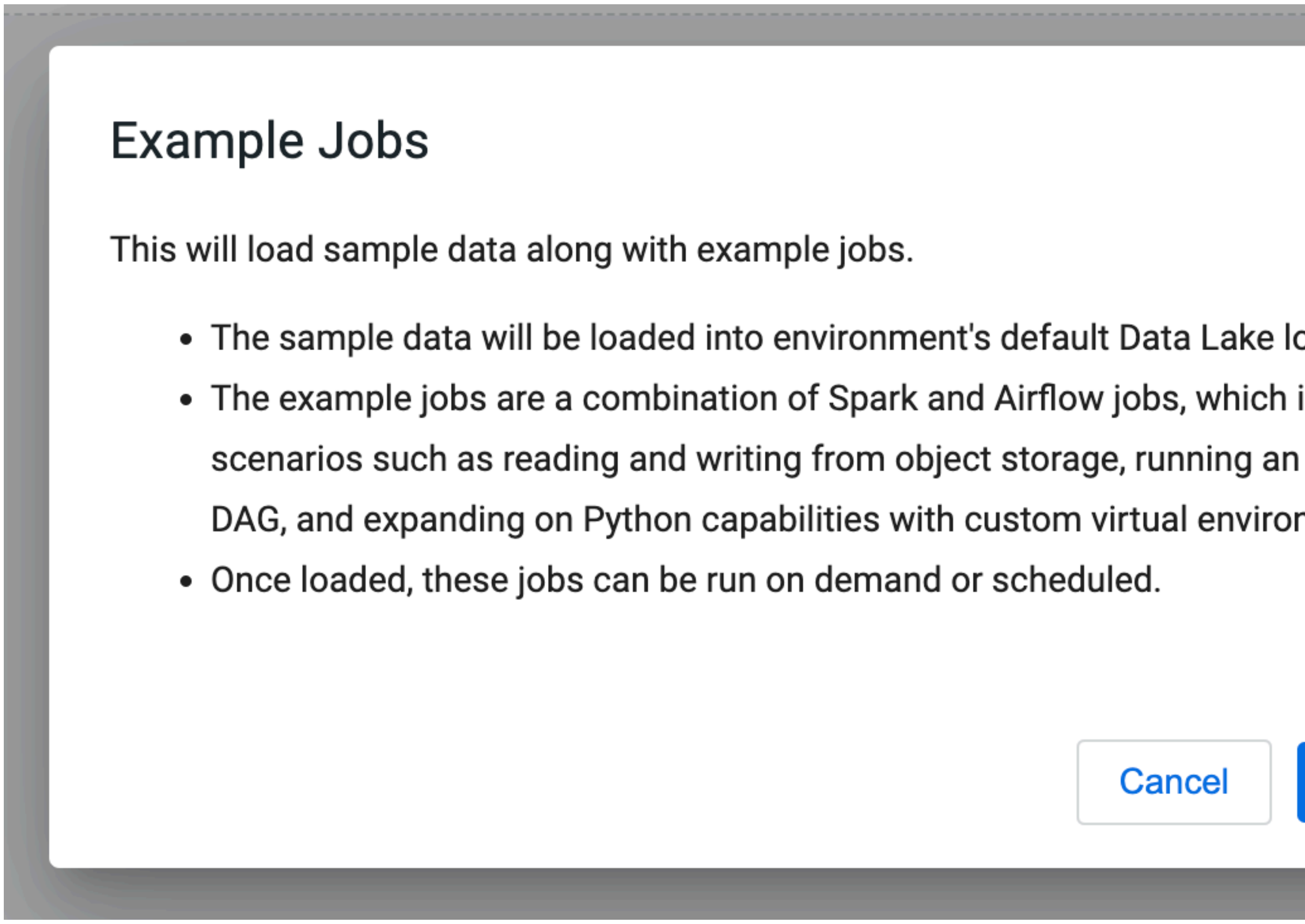

Example Jobs
Explore the capabilities and functionalities of Cloudera Data Engineering with e
jobs.
Load Example Jobs

Note: You will see this window only if you have no existing jobs in the virtual cluster.

5. If you have existing jobs in the virtual cluster, click on the hamburger icon on the jobs page to Load Example Jobs.



6. A dialog box appears explaining the example jobs and sample data. Click Confirm to load example jobs and sample data.



Example Jobs

This will load sample data along with example jobs.

- The sample data will be loaded into environment's default Data Lake lo
- The example jobs are a combination of Spark and Airflow jobs, which i scenarios such as reading and writing from object storage, running an DAG, and expanding on Python capabilities with custom virtual enviror
- Once loaded, these jobs can be run on demand or scheduled.

Results

Example jobs will be loaded in the virtual cluster and sample data will be loaded in the environment’s Data Lake location.

Managing jobs in Cloudera Data Engineering

It is often necessary to modify your Cloudera Data Engineering (CDE) jobs. CDE makes it easy to modify most aspects of your jobs, including replacing the application code and any supplemental files, as well as modifying configuration parameters and the schedule.

Before you begin

Procedure

1. In the Cloudera Data Platform (CDP) management console, click the Data Engineering tile and click Overview.
2. In the CDE Services column, select the environment containing the virtual cluster that your application is associated with.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the application you want to manage.
4. Select Jobs from the left hand menu.

5. Click on the name of the job you want to modify.
6. The Run History tab lists the recent job executions for the application. Click the Configuration tab to display the job configuration.
7. Click the Configure button to edit the application configuration.
8. Edit the configuration parameters you want to change, including uploading a modified JAR or Python file if necessary.
9. Click Advanced Configuration to see additional parameters and the job schedule. Make any necessary changes, and then click Update.

Running Jobs in Cloudera Data Engineering

Jobs in CDE can be run on demand, or scheduled to run on an ongoing basis. The following instructions demonstrate how to run a job in CDE.

Before you begin

Procedure

1. In the Cloudera Data Platform (CDP) management console, click the Data Engineering tile and click Overview.
2. In the CDE Services column, select the environment containing the virtual cluster where you want to run the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to run.
4. To run a job immediately, click the Actions menu next to the application, and then click Run.

   You can cancel a running job by clicking Cancel in the same Actions menu.

   Job Run Notices:The running jobs provide notifications, in the form of a Bell icon next to the job Run ID, when certain conditions are met, without having to parse low level logs, or navigating away to a cloud provider or Kubernetes interface. This will help you to identify why certain job is running slow or stuck, and take actions to rectify this.

Scheduling jobs in Cloudera Data Engineering

Jobs in Cloudera Data Engineering (CDE) can be run on demand, or scheduled to run on an ongoing basis. The following instructions demonstrate how to create or modify a schedule for an existing job.

Before you begin

Procedure

1. In the Cloudera Data Platform (CDP) management console, click the Data Engineering tile and click Overview.
2. In the CDE Services column, select the environment containing the virtual cluster where you want to schedule the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to schedule.
4. Click the Actions menu next to the application, and then click Configuration.
5. Click the Configure.
6. Click the Advanced Configurations link at the bottom of the page to view additional configuration parameters.

7. Select the Schedule toggle, and then set the Start time, End time, and Cron expression.

   The start and end times designate the time frame for which the schedule is active. The Cron expression uses the cron scheduling syntax to specify when the application should run within the start and end times. For information and examples of the cron syntax, see the Cron entry on Wikipedia.

   Note: Timestamps must be specified in ISO-8601 UTC format ('yyyy-MM-ddTHH:mm:ssZ'). UTC offsets are not supported.

   Note: Scheduled job runs start at the end of the first full schedule interval after the start date, at the end of the scheduled period. For example, if you schedule a job with a daily interval with a start_date of 14:00, the first scheduled run is triggered at the end of the next day, after 23:59:59. However if the start_date is set to 00:00, it is triggered at the end of the same day, after 23:59:59.
8. If you want to start a job immediately, check the Start job box.
9. Click Update to save your changes.

Deleting Jobs in Cloudera Data Engineering

If you no longer need a job, you can delete it. Deleting a job does not delete the job run history.

Before you begin

Procedure

1. In the Cloudera Data Platform (CDP) management console, click the Data Engineering tile and click Overview.
2. In the CDE Services column, select the environment containing the virtual cluster where you want to delete the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to delete.

Best practices for building Apache Spark applications

Follow these best practices when building Apache Spark Scala and Java applications:

- Compile your applications against the same version of Spark that you are running.
- Build a single assembly JAR ("Uber" JAR) that includes all dependencies. In Maven, add the Maven assembly plug-in to build a JAR containing all dependencies:

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <configuration>
    <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
  </configuration>
  <executions>
    <execution>
      <id>make-assembly</id>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
    </execution>
  </executions>
```

```
</plugin>
```

This plug-in manages the merge procedure for all available JAR files during the build. Exclude Spark, Hadoop, and Kafka classes from the assembly JAR, because they are already available on the cluster and contained in the runtime classpath. In Maven, specify Spark, Hadoop, and Kafka dependencies with scope provided. For example:

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.11</artifactId>
  <version>2.4.0.7.0.0.0</version>
  <scope>provided</scope>
</dependency>
```