

# CDE Workshop Student Guide

<b>Introduction</b>	<b>3</b>
<b>Prerequisites</b>	<b>4</b>
Verify access to the environment	4
Using a Browser	4
Using Powershell/Terminal	4
Download the resources	6
Update the files	6
<b>Lab 1 - Walkthrough of CDE Data Service</b>	<b>6</b>
<b>Lab 2 - Create and trigger ad-hoc Spark jobs</b>	<b>12</b>
Resource Creation	12
Job Creation	14
Triggering the jobs	17
<b>Lab 3 - Add schedule to the ad-hoc Spark jobs</b>	<b>20</b>
<b>Lab 4 - Orchestrate a set of jobs using Airflow</b>	<b>23</b>
<b>Lab 5 - Install and Configure CDE CLI</b>	<b>28</b>
For Mac users:	28
For Windows users:	32
<b>Lab 6 - Run jobs using CDE CLI</b>	<b>35</b>
Run a spark-scala job using CLI	35
Trigger PySpark jobs using CLI	35
<b>Lab 7 - Data Lineage and Auto-Scaling</b>	<b>37</b>
Data Lineage using Atlas	37
Auto-scaling in CDE	40

## Introduction

---

This document aims to introduce our partners the features of **CDE**, the Data Engineering Data Service of Cloudera Data Platform (**CDP**). During the course of this workshop, you will experience how simple it is to run and orchestrate spark jobs with the help of auto-scaling infrastructure. We will use Airflow for orchestrating the various jobs.

In this workshop:

- You will be given an active CDE service running in an existing/registered CDP environment in a given tenant.
- You will have a virtual cluster with the given configuration that serves as compute for the spark workload.
- You will run sample spark jobs as ad-hoc jobs.
  - PySpark
  - Spark-scala
- You will run the same spark jobs as part of a schedule.
- With the help of Airflow, you will orchestrate a set of Spark jobs and trigger them as a flow.
- You will use CDE CLI to trigger the jobs from terminal/powershell.
- You will see the data lineage using Atlas and witness the auto-scaling capabilities of the CDE Data service.

## Prerequisites

---

### Verify access to the environment

- Update your public IP address on the excel sheet shared with you. To get the public IP address, you can **follow any of the** following approaches.

#### Using a Browser

- Open <https://whatismyipaddress.com/> and IPv4 is what we are looking for.



#### Using Powershell/Terminal

- **WINDOWS:** Open Powershell and run the below command. You will get the public IP address as the output. Copy the entire command including both the parentheses.

```
(Invoke-WebRequest ifconfig.me/ip).Content
```

```
Administrator: Windows PowerShell (x86)
PS C:\Users\cdp_windows> (Invoke-WebRequest ifconfig.me/ip).Content
20.195.101.91
PS C:\Users\cdp_windows>
```

A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell (x86)". The command `(Invoke-WebRequest ifconfig.me/ip).Content` is entered and executed. The output, the public IP address "20.195.101.91", is highlighted with a red arrow. The prompt "PS C:\Users\cdp\_windows>" appears at the bottom.

- **MAC:** Open Terminal and run the below command. You will get the public IP address as the output.

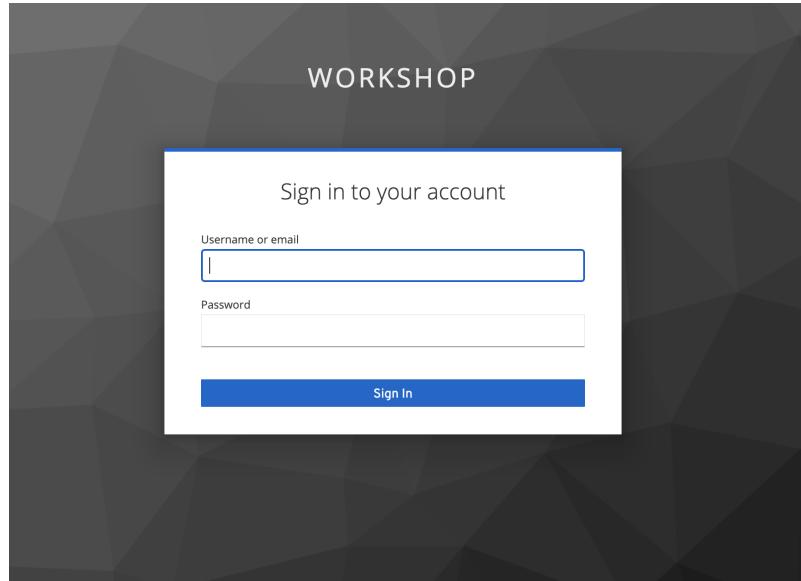
```
curl ifconfig.me
```

```
> curl ifconfig.me
49.37.184.246%
```

A screenshot of a terminal window showing the command `curl ifconfig.me` being run. The output, the public IP address "49.37.184.246%", is displayed at the end of the line.

- Open the below link and login with the credentials assigned to you.

<http://13.234.35.193/auth/realmss/workshop/protocol/saml/clients/samlclient>



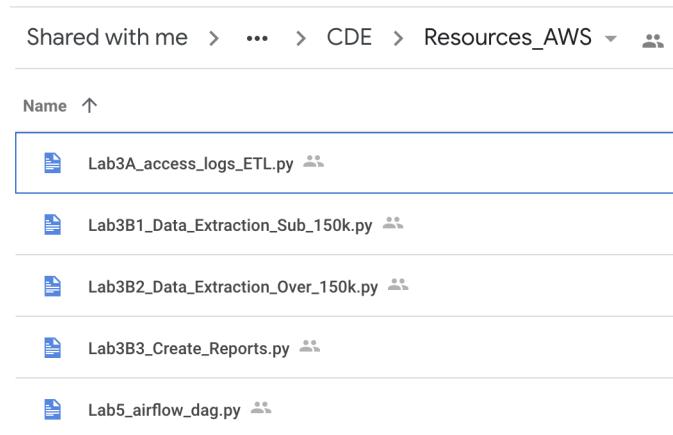
- You should land on the CDP Console as shown below.



## Download the resources

- Download the resources that are required for this workshop from the link below. (Or from the github link shared in the lab)

### [Resources\\_AWS](#)



## Update the files

- Go through each script and update the necessary values as mentioned in the script.
  - For all the scripts, update the username field with the username that you have been assigned to. You will find this at the starting of the script itself.

## Lab 1 - Walkthrough of CDE Data Service

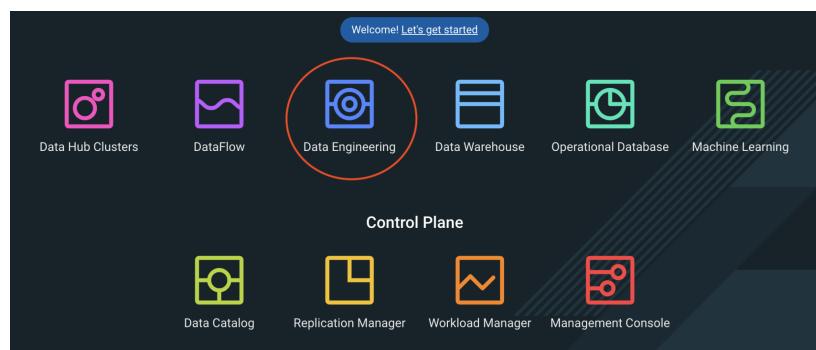
Cloudera Data Engineering (CDE) is a serverless service for Cloudera Data Platform that allows you to submit jobs to auto-scaling virtual clusters.

The CDE service involves several components:

- **Environment**
  - A logical subset of your cloud provider account including a specific virtual network.
- **CDE Data Service**
  - The long-running Kubernetes cluster and services that manage the virtual clusters. The CDE service must be enabled in an environment before you can create any virtual clusters.
- **Virtual Cluster**
  - An individual auto-scaling cluster with defined CPU and memory ranges. Virtual Clusters in CDE can be created and deleted on demand. Jobs are associated with clusters.
- **Job**
  - Application code along with defined configurations and resources. Jobs can be run on demand or scheduled.
- **Resource**
  - A defined collection of files such as a Python file or application JAR, dependencies, and any other reference files required for a job.
- **Job run**
  - An individual job run.

The above components can be accessed in the following ways:

- Go to the CDP console and click on Data Engineering.



- You will see a CDE service and an associated virtual cluster.

The screenshot shows two main sections: 'CDE Services' and 'Virtual Clusters'.

**CDE Services:** A card for 'pse-cde-workshop' is displayed, showing it is 'Enabled'. It lists 'aws pse-workshop-nov' and provides resource details: 1 NODES, 0 CPU, and 0 MB MEMORY.

**Virtual Clusters:** A card for 'cde-vc-1' is shown, which is associated with 'pse-cde-workshop'. It is 'Running' and displays resource usage: 0 CPU, 0 MB MEMORY, and 0 JOBS. A bar chart visualizes the usage over time.

- On the **CDE service pse-cde-workshop**, click on the pencil icon and observe the configuration and other details related to the service.

A detailed view of the 'pse-cde-workshop' service card is shown, enclosed in a blue border. The card includes the following information:

- Status:** Enabled (indicated by a green circle with a checkmark).
- Region:** aws pse-workshop-nov
- Resource Allocation:**

NODES	CPU	MEMORY
1	0	0 MB
- Action Buttons:** A blue pencil icon with a red arrow pointing to it, indicating where to click to edit the service configuration.

GRAFANA CHARTS RESOURCE SCHEDULER

Configuration Charts Logs Access Diagnostics

**Environment**

pse-workshop-nov

**Workload Type** Use SSD instances 

General - Medium (EBS)

**EBS Size** 

500 GB

**Capacity & Costs**

## Autoscale Range

## On-demand Instances



- Click on each tab and go through all the details related to the CDE service.
- Once done, click on the **Overview** to go back to the CDE home page.

Overview / pse-cde-workshop

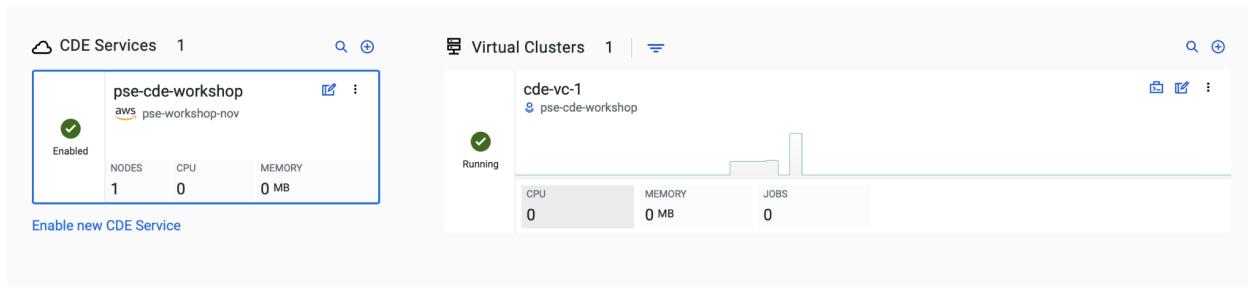
Enabled

**pse-cde-workshop**

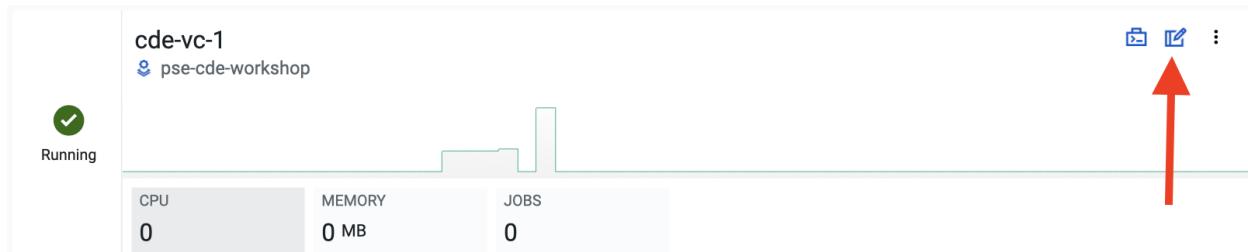
VERSION	CLUSTER ID	CREATED BY	NODES	CPU	MEMORY
1.12.0-b119	cluster-ddsc4xhh	Pannag Katti	1 / 75	0 / 1200	0 MB / 4800 GB

- You will land back on this page.

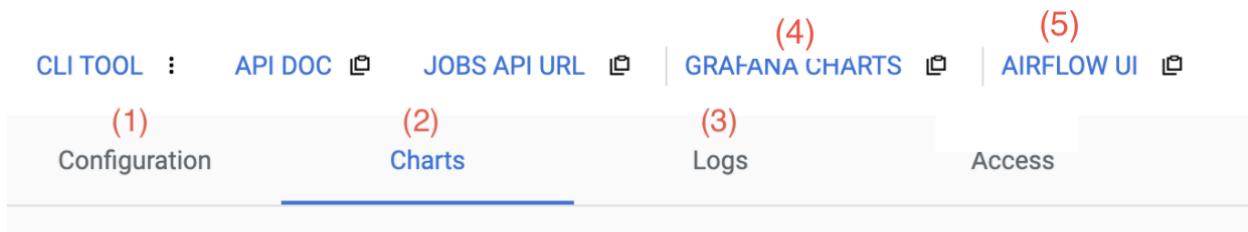
## Overview



- On the right side, you will see the virtual cluster attached to this CDE service **pse-cde-workshop**. Click on the pencil icon on the **virtual cluster(VC)** **cde-vc-1** and observe the configuration details.



- Notice the compute limits that are set for this virtual cluster. When the demand increases, CDE will autoscale this cluster as per these values. At the end of this lab, we will see this with the help of dashboards.
- Navigate to all the other tabs numbered in the screenshot below and observe the details related to the virtual cluster.

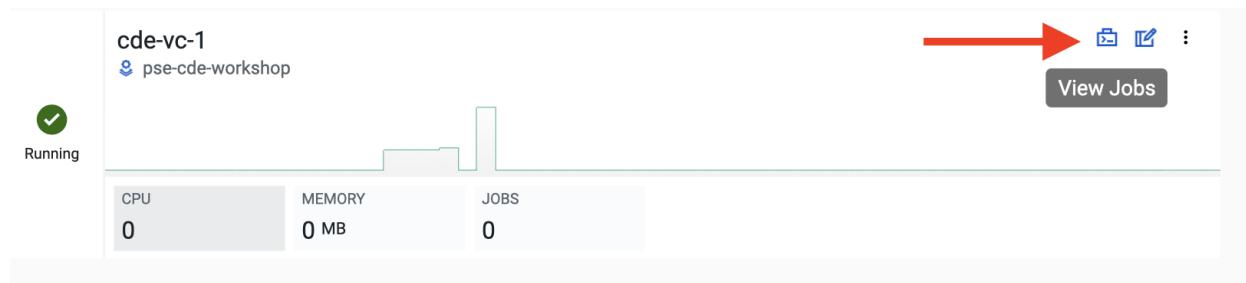


- Once done, click on the **Overview** to go back to the CDE home page.

## Overview / cde-vc-1



- On the virtual cluster **cde-vc-1** tab, click on view jobs. This will open a new page with details of the Job Runs, Jobs, and Resources.



- Click on each tab to get yourself acquainted.

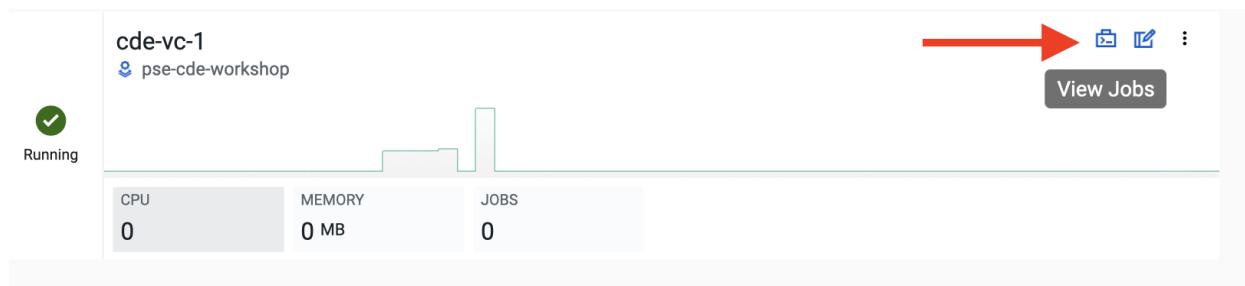
## Lab 2 - Create and trigger ad-hoc Spark jobs

In this lab, we will create spark jobs and run them on an ad-hoc basis, i.e., without any schedule. As part of this lab, we have taken two simple use-cases that can be addressed with the help of Spark jobs.

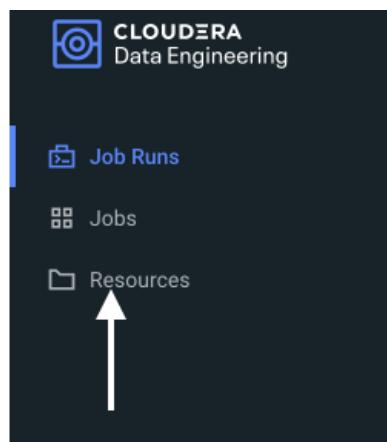
1. Log Data Cleansing using Spark
2. Analyze the Paycheck Protection Program Data
  - a. Report 1: Breakdown of all cities in Texas that retained jobs
  - b. Report 2: Breakdown of company type that retained jobs
3. PySpark job to enrich your data using an existing data warehouse

### Resource Creation

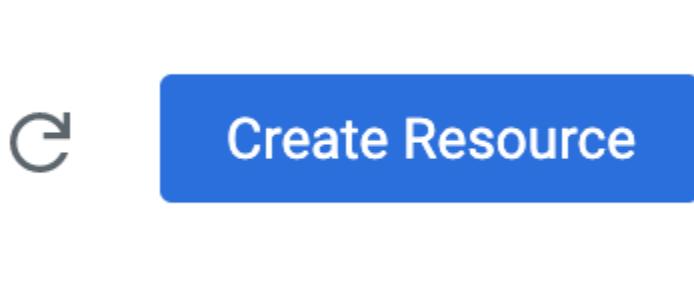
- On the virtual cluster **cde-vc-1** tab, click on view jobs. This will open a new page with details of the Job Runs, Jobs, and Resources.



- In the left pane, click on the **Resources** tab.



- You will get the **Resources** page to the right. Click on **Create Resource**.

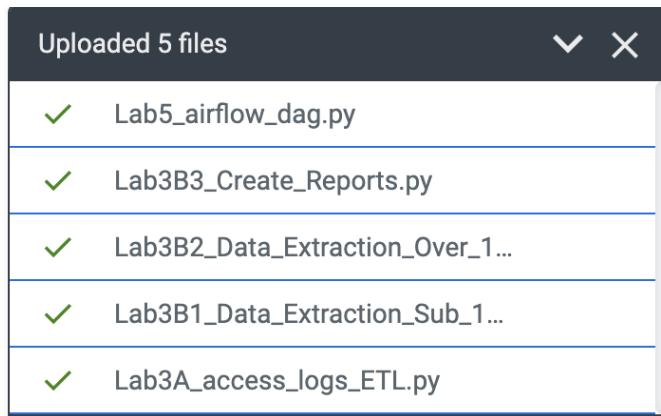


- Give a unique name(username-resources) and create the resource. This acts as your repository for storing all the scripts and dependencies.
- Once it is created, you will get an option to upload the files as shown below.

Drop files here or click on the "Upload Files" button to select them from your computer

Upload Files

- Click on **Upload Files** and select all the scripts downloaded from the [prerequisites](#) step. (**Please upload only .py and .scala files**)
- You will get a pop-up with all the files uploaded to your resource.



- Click on **Resources** at the top, select your resource and validate if all the seven **.py** files and one **.scala** are present in your resource. We are now ready to create jobs using these resources.

Resources / pkatti-resources / Files



## Job Creation

- We will now create the first job with the script ***Lab3A\_access\_logs\_ETL.py***.
- In the left pane, click on **Jobs**.
- You will get the **Jobs** page to the right. Click on **Create Job**.



- Select job type as **Spark**.
- Please give the job names as mentioned below.

<username>\_<script\_name\_without\_py\_extension>

Eg:- For wuser1, job1 name would be **wuser1\_Lab3A\_access\_logs\_ETL**

### Job Details

Job Type \*

Spark 2.4.7  Airflow

Name \*

wuser01\_Lab3A\_access\_logs\_ETL

- As this is a shared environment, please name the jobs with your username so that it helps in differentiating yours from others' jobs.
- In **Application File**, click on **Resource** and select the file ***Lab3A\_access\_logs\_ETL.py*** from your resource(<username>-resources).

Application File

File ⓘ  Resource ⓘ  URL ⓘ

Select file from Resource

**Select File**

Search Resources

mmehra\_resources

Lab3A\_access\_logs\_ETL.py

Lab3B1\_Data\_Extraction\_Sub\_150k.py

Lab3B2\_Data\_Extraction\_Over\_150k.py

Lab3B3\_Create\_Reports.py

Lab5\_airflow\_dag.py

AutoResource-1635833838991

AutoResource-1635582769149

1 - 8 of 8 < >

Select File

- Ignore the remaining configuration options. Do not enable the schedule now. This is how it should finally look like.

**Job Details**

Job Type \*

Spark 2.4.7  Airflow

Name \*

wuser01\_Lab3A\_access\_logs\_ETL

Application File

File  Resource  URL

✓ Lab3A\_access\_logs\_ETL.py X

Main Class

com.package.MainClass

Arguments (Optional)

Argument +

Configurations (Optional)

config\_key config\_value +

Python Version

Python 3  Python 2

**Advanced Options**

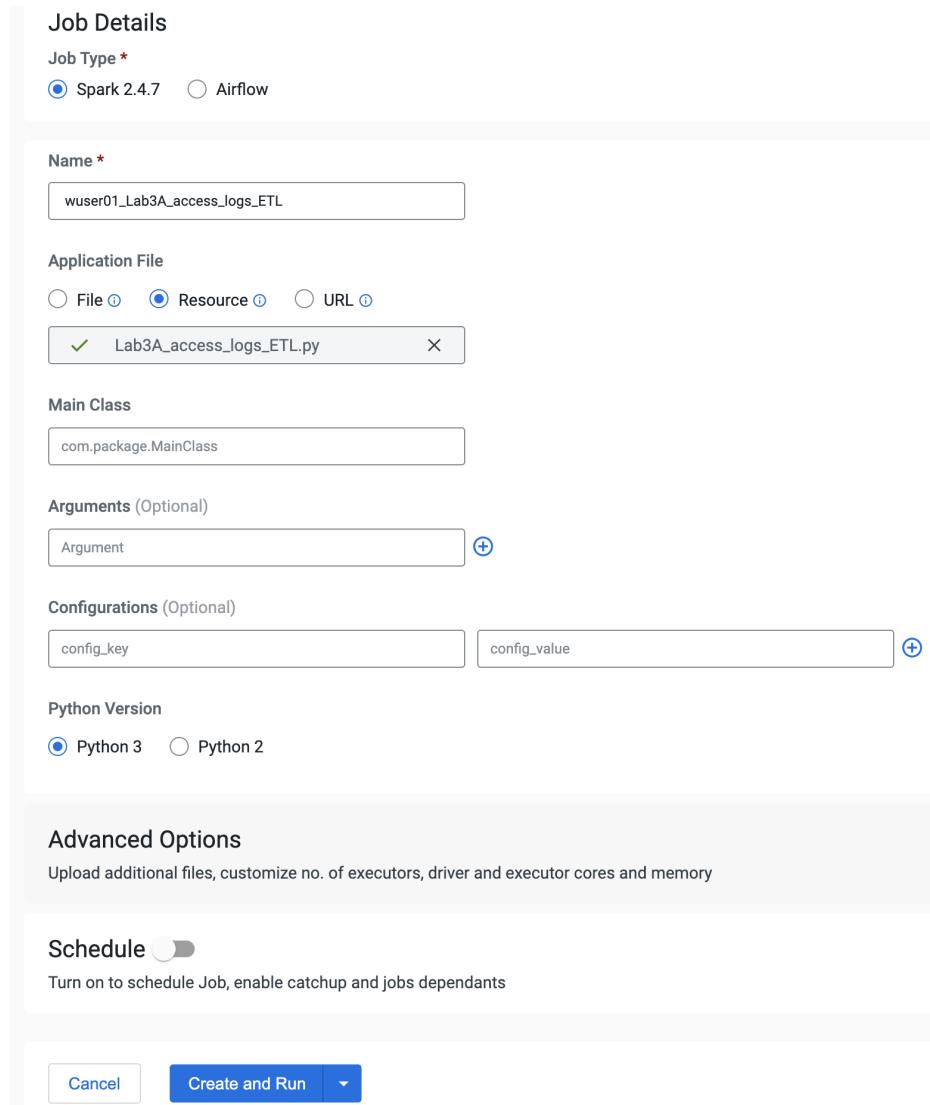
Upload additional files, customize no. of executors, driver and executor cores and memory

Schedule

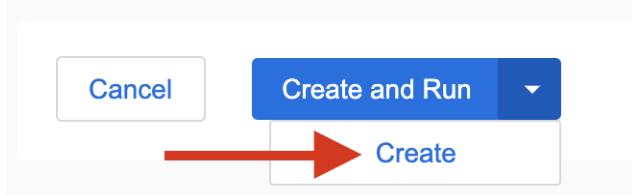
Turn on to schedule Job, enable catchup and jobs dependants

**Create and Run** ▾

Cancel Create and Run ▾ Create



- Click on the drop down option and click on **Create**.



- Similarly, create four other jobs with the same naming conventions. Please refer to the table below to confirm you are creating exactly the same.

For **wuser1**:

Jobs	Job Name	Script Used
Job1	wuser1_Lab3A_access_logs_ETL	Lab3A_access_logs_ETL.py
Job2	wuser1_Lab3B1_Data_Extraction_Sub_150k	Lab3B1_Data_Extraction_Sub_150k.py
Job3	wuser1_Lab3B2_Data_Extraction_Over_150k	Lab3B2_Data_Extraction_Over_150k.py
Job4	wuser1_Lab3B3_Create_Report	Lab3B3_Create_Report.py

- Create these jobs as **ad-hoc** jobs i.e., without any schedule.
- Once done, click on the **Jobs** tab and enter your username in the search bar and press **ENTER**. You should see four jobs as shown below with your username.

Status	Job	Type	Schedule	Modified On	Actions
Idle	wuser01_Lab3B3_Create_Report	Spark	Ad-Hoc	Nov 3, 2021, 12:05:56 PM	⋮
Idle	wuser01_Lab3B2_Data_Extraction_Over_150k	Spark	Ad-Hoc	Nov 3, 2021, 12:05:34 PM	⋮
Idle	wuser01_Lab3B1_Data_Extraction_Sub_150k	Spark	Ad-Hoc	Nov 3, 2021, 12:05:09 PM	⋮
Idle	wuser01_Lab3A_access_logs_ETL	Spark	Ad-Hoc	Nov 3, 2021, 12:04:40 PM	⋮

- Observe the type of the job is set to Spark and for schedule, it is Ad-hoc.

## Triggering the jobs

- You can trigger each job one by one and observe the different tabs under job run.
- To trigger the job, go to the **Jobs** tab, click on the 3-dotted icon, and click on **Run Now**.

Jobs

Search: wuser Filter By: Status Type

Status	Job	Type	Schedule	Modified On	Actions
○	wuser01_Lab3B3_Create_Reports	Spark	Ad-Hoc	Nov 3, 2021, 12:05:56 PM	⋮
○	wuser01_Lab3B2_Data_Extraction_Ov...	Spark	Ad-Hoc	Nov 3, 2021, 12:05:34 PM	⋮
○	wuser01_Lab3B1_Data_Extraction_Su...	Spark	Ad-Hoc	Nov 3, 2021, 12:05:09 PM	⋮
○	wuser01_Lab3A_access_logs_ETL	Spark	Ad-Hoc	Nov 3, 2021, 12:04:40 PM	⋮

Items per page: 10

⋮ Run Now Add Schedule Clone Configuration Delete

- To check the job logs, click on **Job Runs** and select the ID against the job that you have triggered.

Job Runs

Job Name Search Job Runs Filter By: Status Type

Status	Run ID	Job	Type	User	Duration	Start Time	Actions
○	47	wuser01_Lab3B2_Data_Extraction_Ov...	Spark	wuser01	Nov 3, 2021, 12:09:05 PM		⋮
○	46	wuser01_Lab3B1_Data_Extraction_Su...	Spark	wuser01	Nov 3, 2021, 12:09:04 PM		⋮
○	45	wuser01_Lab3A_access_logs_ETL	Spark	wuser01	Nov 3, 2021, 12:08:59 PM		⋮

Job Runs / 47

Status: Succeeded Job: wuser01\_Lab3B2\_Data\_Extra... Lineage: Atlas Duration: 2.4 min Start Time: Nov 3, 2021, 12:09:05 PM

Trends Configuration Logs Spark UI

Duration

2.6 MIN  
2.5 MIN  
2.4 MIN  
2.3 MIN  
2.2 MIN

Search by Run Id

Status	Run ID	Duration	Executor Memory	Drivers Cores	Executor Cores	User	Start Time	Actions
○	47	2.4 min	1G	1	1	wuser01	Nov 3, 2021, 12:09:05 PM	⋮

- For simplifying the job selection, you can choose the **User** filter and add your username and hit enter. You will see the list of jobs triggered by you.

## Job Runs

The screenshot shows a search interface for 'Job Runs'. At the top, there is a search bar with a magnifying glass icon and a dropdown menu labeled 'User' with a downward arrow. Below the search bar is a 'Search Job Runs' input field. To the right of the search bar is a small square icon with a bracket symbol. On the left side of the interface, there are three tabs: 'Run Id', 'Status', and 'User'. The 'User' tab is highlighted with a blue background and white text. A large red arrow points from the text above to the 'User' dropdown in the search bar.

- Navigate to different tabs in the job run page and you will see all that you need to observe for the run of a spark job.

The screenshot shows a navigation bar with four tabs: 'Trends' (highlighted with a blue underline), 'Configuration', 'Logs', and 'Spark UI'. The tabs are positioned horizontally at the top of the page.

## Lab 3 - Add schedule to the ad-hoc Spark jobs

In this lab, we will add a schedule to a job created as part of the previous lab.

- We will add a schedule to the job **Lab3A\_access\_logs\_ETL** (in your case it will be <username>\_Lab3A\_access\_logs\_ETL)
- Go to **Jobs** tab, click on the 3-dotted icon next to the job **Lab3A\_access\_logs\_ETL** and select **Add schedule**.

- You will land in the **Job Schedule** page. Click on **Create a Schedule**.

- Choose the **Cron Expression** option and enter the cron expression as given below.

**\*/10 \* \* \* \*** → This means that the job is scheduled to run every 10 minutes.

Jobs / wuser01\_Lab3A\_access\_J...

Status	Runs
Ad-Hoc	1

Run History Configuration Schedule Actions : vc1

Basic  Cron Expression

`*/5 * * * *`

Every 5 minutes (UTC)

Start Date: Wednesday, November 3, 2021 at 12:20:16 PM

End Date: Thursday, November 4, 2021 at 12:15:16 PM

Scheduling Configurations

Enable Catchup  
Kick off Job Runs for intervals that has not been run along with the current interval

Depends on Previous  
If the job runs are dependant, then the catchup will occur serially

Cancel Add Schedule

Jobs / wuser01\_Lab3A\_access\_J...

Status	Schedule	Runs
Scheduled	<code>*/5 * * * *</code>	1

Run History Configuration Schedule Actions : vc1

Basic  Cron Expression

`*/5 * * * *`

Every 5 minutes (UTC)

Start Date: Wednesday, November 3, 2021 at 12:20:16 PM

End Date: Thursday, November 4, 2021 at 12:15:16 PM

Scheduling Configurations

Enable Catchup  
Kick off Job Runs for intervals that has not been run along with the current interval

Depends on Previous  
If the job runs are dependant, then the catchup will occur serially

Cancel Save Changes

Job Schedule has been updated

- You can repeat the same process for the other jobs as well.

Jobs

vc1

Search Jobs Filter By: Status Type Create Job

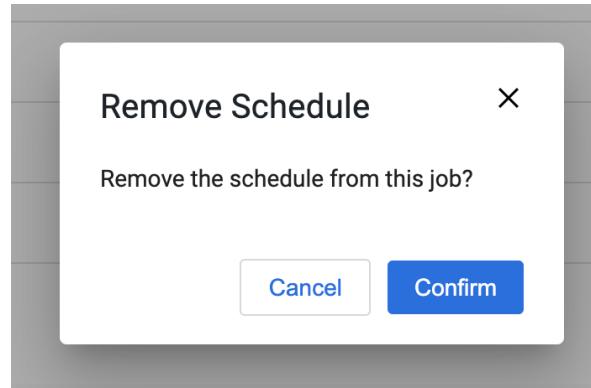
Status	Job	Type	Schedule	Modified On	Actions
○	wuser01_Lab3B1_Data_Extraction_Su...	Spark	<code>*/5 * * * *</code>	Nov 3, 2021, 12:20:04 PM	⋮
○	wuser01_Lab3B2_Data_Extraction_Ov...	Spark	<code>*/5 * * * *</code>	Nov 3, 2021, 12:19:43 PM	⋮
○	wuser01_Lab3B3_Create_Report	Spark	<code>*/5 * * * *</code>	Nov 3, 2021, 12:19:23 PM	⋮
○	wuser01_Lab3A_access_logs_ETL	Spark	<code>*/5 * * * *</code>	Nov 3, 2021, 12:18:07 PM	⋮

- You can modify the cron expression if you wish to. Refer to [this](#) to validate your cron expression.
- Wait for the scheduler to trigger the jobs at the scheduled time and observe the results and logs for the same.
- Once done, please delete the schedule for all the jobs for which it was added by following the below steps.**

- Go to the Jobs tab, click on the 3-dotted icon next to the job **Lab3A\_access\_logs\_ETL** and select **Remove schedule**.

The screenshot shows the 'Jobs' page in the Cloudera Data Engineering interface. A table lists four jobs: 'wuser01\_Lab3B1\_Data\_Extraction\_Su...', 'wuser01\_Lab3B2\_Data\_Extraction\_Ov...', 'wuser01\_Lab3B3\_Create\_Reports', and 'wuser01\_Lab3A\_access\_logs\_ETL'. The fourth job has its 'Actions' menu open, with the 'Remove Schedule' option highlighted by a red box.

Status	Job	Type	Schedule	Modified On	Actions
○	wuser01_Lab3B1_Data_Extraction_Su...	Spark	*/5 * * * *	Nov 3, 2021, 12:20:04 PM	⋮
○	wuser01_Lab3B2_Data_Extraction_Ov...	Spark	*/5 * * * *	Nov 3, 2021, 12:19:43 PM	⋮
○	wuser01_Lab3B3_Create_Reports	Spark	*/5 * * * *	Nov 3, 2021, 12:19:23 PM	⋮
○	wuser01_Lab3A_access_logs_ETL	Spark	*/5 * * * *	Nov 3, 2021, 12:18:07 PM	⋮



The screenshot shows the 'Jobs' page after the schedule was removed. The 'wuser01\_Lab3A\_access\_logs\_ETL' job now has an 'Ad-Hoc' schedule. A green notification bar at the top right indicates 'Job Schedule has been removed'.

Status	Job	Type	Schedule	Modified On	Actions
○	wuser01_Lab3A_access_logs_ETL	Spark	Ad-Hoc	Nov 3, 2021, 12:28:32 PM	⋮
○	wuser01_Lab3B1_Data_Extraction_Su...	Spark	*/5 * * * *	Nov 3, 2021, 12:20:04 PM	⋮
○	wuser01_Lab3B2_Data_Extraction_Ov...	Spark	*/5 * * * *	Nov 3, 2021, 12:19:43 PM	⋮
○	wuser01_Lab3B3_Create_Reports	Spark	*/5 * * * *	Nov 3, 2021, 12:19:23 PM	⋮

## Lab 4 - Orchestrate a set of jobs using Airflow

In this lab, we will create a flow with the help of a dag file that uses the jobs created in Lab3. Thus, you will be able to complete subsequent labs only if you have completed Lab3 successfully.

- Go to Jobs tab, click on **Create Job** and choose Airflow in Job type.
- Give the job name as below and upload the *Lab5\_airflow\_dag.py* file from the resources.

JOB NAME : <username>\_Lab5\_airflow\_dag

Example : For user **wuser01** the job name will be, **wuser01\_Lab5\_airflow\_dag**

- Click on **Create**.

Jobs / Create Job

Job Details

Job Type \*

Spark 2.4.7  Airflow

Name \*

wuser01\_Lab5\_airflow\_dag

DAG File \*

File  Resource  Editor

Lab5\_airflow\_dag.py

Cancel Create and Run Create

- Go to **Jobs** tab and observe the airflow job created with the schedule mentioned in the dag file.

### Job

Status	Job	Type	Schedule	Modified On ↓	Actions
Idle	wuser01_Lab5_airflow_dag	Airflow	*/5 * * * *	Nov 3, 2021, 1:02:39 PM	⋮
Idle	pkatti-Lab6B	Spark	Ad-Hoc	Nov 3, 2021, 12:57:19 PM	⋮

## DAG File

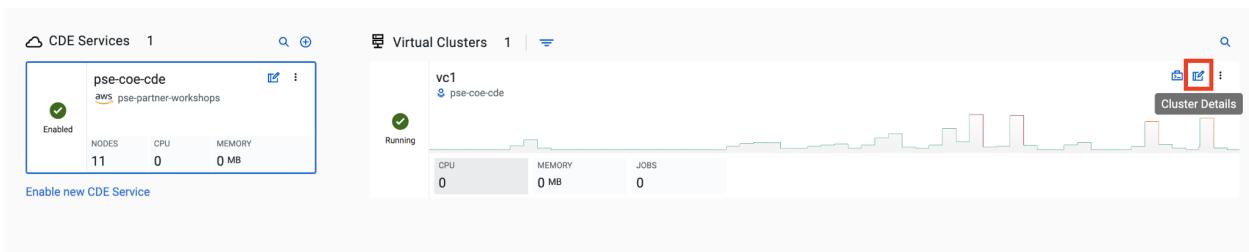
```

12 DAG_name = owner + "_Airflow_Dag"
13 job_name_1 = owner + "_Lab3B1_Data_Extraction_Sub_150k"
14 job_name_2 = owner + "_Lab3B2_Data_Extraction_Over_150k"
15 job_name_3 = owner + "_Lab3B3_Create_Report"
16
17
18 default_args = {
19     'owner': owner,
20     'retry_delay': timedelta(seconds=5),
21     'start_date': datetime.now(),
22     'email_on_failure': False,
23     'email_on_retry': False,
24     'retries': 0
25 }
26
27 dag = DAG(
28     DAG_name,
29     default_args=default_args,
30     schedule_interval='*/5 * * * *',
31     catchup=False,
32     is_paused_upon_creation=False
33 )
34
35 start = DummyOperator(task_id='start', dag=dag)
36
37 Data_Extraction_Sub_150k = CDEJobRunOperator(
38     task_id=job_name_1

```

- Go to the Virtual Cluster <> and click on **Cluster Details**.

Overview



- Click on **Airflow UI** and observe the schedule created for your job.

Overview / vc1

**vc1** Running

VERSION: 1.12.0-b119 VC ID: dex-app-hg98mkgj CREATED BY: Pannag Katti CPU: 3 MEMORY: 4 GB JOBS: 0

ENVIRONMENT DATA LAKE

**AIRFLOW UI**

Configuration Charts Logs

CDE Service: pse-partner-workshops

Autoscale Max Capacity: CPU 30 600, Memory (GB) 508 2400

Driver and Executors will run on: By default Driver would run on-demand and executors on spot

Spark Version: Spark 2.4.7

Enable Airflow Job Authoring UI (Technical Preview)

07:37 UTC

DAGs

All 3 Active 3 Paused 0

Filter DAGs by tag

Search DAGs

DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	Actions	Links		
wuser01_Airflow_Dag	wuser01	0000	/5 ****	0000000000000000	0000000000000000				...

**DAG: wuser01\_Airflow\_Dag**

schedule: \*/5 \* \* \* \*

Tree View Graph View Calendar View Task Duration Task Tries Landing Times Gantt Details Code

2021-11-03T07:39:37Z Runs 25 Update No DAG runs yet.

CDEJobRunOperator DummyOperator

Auto-refresh

```

graph TD
    start((start)) --> wuser01_Lab3B1_Data_Extraction_Sub_150k[wuser01_Lab3B1_Data_Extraction_Sub_150k]
    wuser01_Lab3B1_Data_Extraction_Sub_150k --> wuser01_Lab3B2_Data_Extraction_Over_150k[wuser01_Lab3B2_Data_Extraction_Over_150k]
    wuser01_Lab3B2_Data_Extraction_Over_150k --> wuser01_Lab3B3_Create_Reports[wuser01_Lab3B3_Create_Reports]
    wuser01_Lab3B3_Create_Reports --> end((end))

```

- Once the job has run successfully, we need to edit the job to pause the schedule.
- Click on the Jobs tab and locate the airflow job that you have just created.

- Next to the job, click on the 3 dots and click on Configuration.

Status	Job	Type	Schedule	Modified On	Actions
🕒	wuser39_airflow	Airflow	*/5 * * * *	Nov 9, 2021, 5:39:53 PM	
🕒	wuser04_Lab3B3_Create_Reports	Spark	Ad-Hoc	Nov 9, 2021, 5:11:05 PM	

- Under Configuration, click on Edit.

Run History      Configuration      Schedule      Airflow UI      Actions :

Name \*

Actions :

- In the DAG File section, click on the cross button to remove the dag file.

Run History      Configuration      Schedule

Name \*

DAG File \*

File    Resource



Lab5\_airflow\_dag.py



Cancel

Update and Run



- Select **Resource** and upload the file **Lab5A\_airflow\_dag\_STOP.py**.

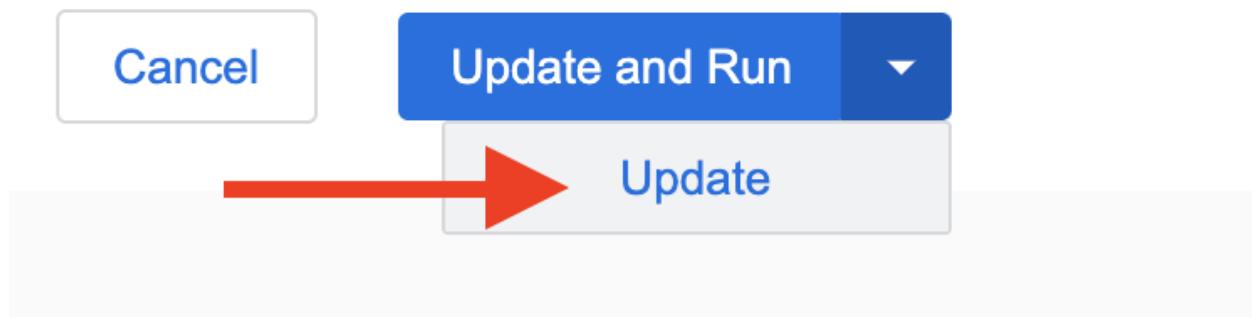
**DAG File \***

File [i](#)     Resource [i](#)

**Upload File \***

[Choose file](#) Python file

- Click on **Update**.

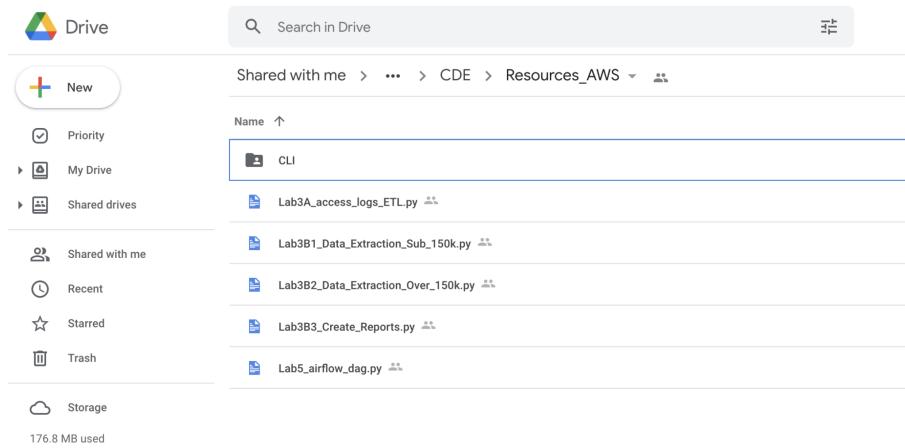


- Once done, go to the **Schedule** tab of the job and you should see the similar status as there is no schedule for the given job.

The screenshot shows the 'Schedule' tab of a job in the CDE UI. The tab bar includes 'Run History', 'Configuration', 'Schedule' (which is highlighted), and 'Airflow UI'. A message box displays the text: 'This job currently does not have a schedule.'

## Lab 5 - Install and Configure CDE CLI

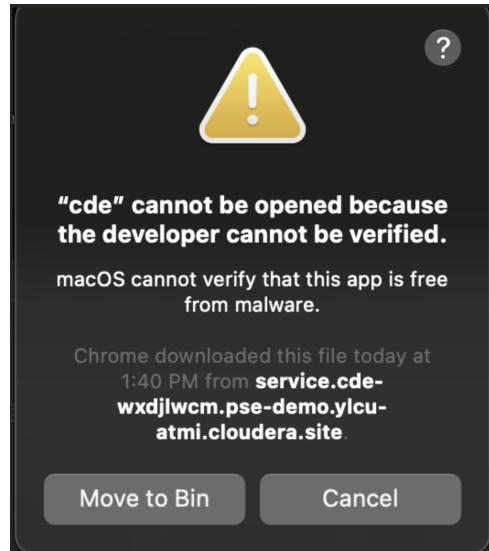
- In this lab, we will use the CDE CLI to create and run a spark job. This way, you can use the rich api's of CDE CLI to integrate any of your applications to communicate with the CDE service.
- You will find the CLI in the resources folder downloaded from the prerequisites step.



### For Mac users:

- Make sure that the `cde` file is executable by running the below command.  

```
chmod +x /path/to/cde
```
- Open the CDE CLI. You might get the below error.



- Go to System Preferences → Security and Privacy and add this app in the trust center.
- Once done, add the path to the cli in the PATH variable.  
Run the below command in the terminal to update the path variable.

```
export PATH=$PATH:"<path-to-cde>"
```

**Example :** If the file is situated in the path  
`/home/user/applications/cde`

Then the export command will look something like this

```
export PATH=$PATH:/home/user/applications/
```

**Note :** Once the terminal is closed, the step to add the path to the PATH variable needs to be performed again.

- To validate the installation, run the below command from the terminal.  
`cde --help`

```
> cde --help
Cloudera Data Engineering

Usage:
  cde [command]

Available Commands:
  airflow      Airflow commands
  backup       Create and Restore CDE backups
  credential   Manage CDE credentials
  help         Help about any command
  job          Manage CDE jobs
  resource    Manage CDE resources
  run          Manage CDE runs
  spark        Spark commands

Flags:
  --access-key-id string      access key identifier
  --access-key-secret string  access key secret
  --auth-cache-file string    token file cache location (default "$USERCACHE/token-cache")
  --auth-no-cache              do not cache authentication tokens
  --auth-pass-file string    authentication password file location
  --credentials-file string   CDP credentials file location
  --credentials-profile string CDP credentials profile name (default "default")
  -h, --help                  help for cde
  --hide-progress-bars        hide progress bars for file uploads
  --insecure                 API does not require authentication
  --skip-credentials-file    skip CDP credentials file discovery
```

- If you get the output as shown above, then the installation is completed successfully. We now need to configure the CLI to connect to our virtual cluster.
- For configuring the CDE CLI, we create a new file and add the cluster details and use it as an environment variable for connecting to the CDE virtual cluster.
- Create a file as config.yaml and add the following details.

```
touch config.yaml
```

```
[mehra@MacBook-Pro workshop % touch config.yaml
[mehra@MacBook-Pro workshop % ls
  cde      config.yaml
mehra@MacBook-Pro workshop % ]
```

```
vi config.yaml
```

```
user: <CDP_user>
vcluster-endpoint: <CDE_virtual_cluster_endpoint>
```

Here, **user** is the username you have been mapped in the excel sheet. Use the below url for **vcluster-endpoint**.

```
https://v2wvrmwc.cde-ddsc4xhh.pse-work.dp5i-5vkq.cloudera.site/dex/api/v1
```

```
user: wuser01
vcluster-endpoint: https://hg98mkgj.cde-7fsd4m65.pse-part.dp5i-5vkq.cloudera.site/dex/api/v1
```

- Open terminal and run the below command to create an environment variable.

```
export CDE_CONFIG=/path/to/config.yaml
```

- Run the below command to verify whether the above step is completed successfully. You should see the path-to-config.yaml as the output.

```
echo $CDE_CONFIG
```

- Run the below command to validate the configuration. Upon running it, you will be asked to provide the API password. Please enter the password as mentioned in the excel sheet.

```
cde job list
```

```
› cde job list
API User Password: 🔑
```

- Once you enter the password, you should see all the jobs present in the virtual cluster.

```

mnehra@MacBook-Pro workshop % cde job list
API User Password:
[
  {
    "name": "pkatti-Lab6A",
    "type": "spark",
    "created": "2021-11-03T07:19:23Z",
    "modified": "2021-11-03T07:19:09Z",
    "retentionPolicy": "keep_indefinitely",
    "mounts": [
      {
        "resourceName": "AutoResource-1635923577528"
      }
    ],
    "spark": {
      "file": "Lab6B1_PrepSetupDW.py",
      "driverMemory": "1g",
      "driverCores": 1,
      "executorMemory": "1g",
      "executorCores": 1,
      "conf": {
        "dax.safariEnabled": "false",
        "spark.pyspark.python": "python3"
      },
      "logLevel": "INFO"
    },
    "schedule": {
      "enabled": false,
      "user": "pkatti"
    }
  },
  {
    "name": "pkatti-Lab6B",
    "type": "spark",
    "created": "2021-11-03T07:19:20Z",
    "modified": "2021-11-03T07:19:09Z",
    "retentionPolicy": "keep_indefinitely",
    "mounts": [
      {
        "resourceName": "AutoResource-1635925315679"
      }
    ],
    "spark": {
      "file": "Lab6B2_EnrichData_ETL.py",
      "driverMemory": "1g",
      "driverCores": 1,
      "executorMemory": "1g",
      "executorCores": 1,
      "conf": {
        "dax.safariEnabled": "false",
        "spark.pyspark.python": "python3"
      },
      "logLevel": "INFO"
    },
    "schedule": {
      "enabled": false,
      "user": "pkatti"
    }
  },
  {
    "name": "test_s3.write",
    "type": "spark",
    "created": "2021-10-30T08:24:21Z",
    "modified": "2021-10-31T07:53:33Z",
    "retentionPolicy": "keep_indefinitely",
    "mounts": [
      {
        "resourceName": "AutoResource-1635582769149"
      }
    ],
    "spark": {
      "file": "test_s3.write.py",
      "driverMemory": "1g",
      "driverCores": 1,
      "executorMemory": "1g",
      "executorCores": 1,
      "conf": {
    
```

- If you get any error related to the certificate, please add the flag to skip tls verification.

```
cde job list --tls-insecure
```

- This marks the end of installation and configuration of CDE CLI. Now, head over to the next lab to trigger the jobs from CLI.

## For Windows users:

- Open Powershell and navigate to the folder where you have downloaded the cde.exe file.
- You can use the below command to navigate.

```
cd C:\Users\<path-to-cde.exe folder>
```

- Run the below command to start the cde cli. It will be executed in the background.

```
start .\cde.exe
```

```

PS C:\Users\cdp_windows> cd C:\Users\cdp_windows\Desktop
PS C:\Users\cdp_windows\Desktop> ls

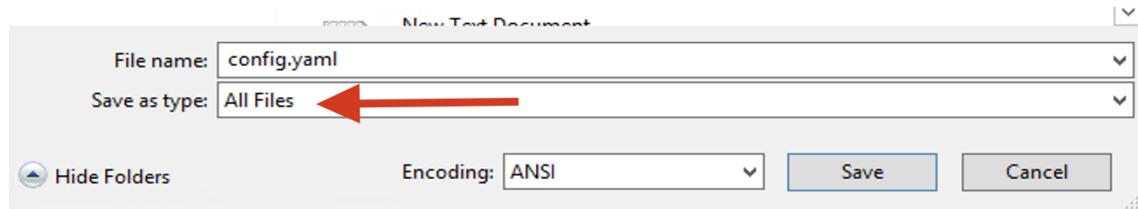
    Directory: C:\Users\cdp_windows\Desktop

Mode                LastWriteTime     Length Name
----                -----          ---- 
-a---        10/28/2021  8:37 AM    38056448 cde.exe
-a---        10/28/2021  8:52 AM       106 config.yaml
-a---        10/28/2021  9:00 AM         0 he.txt
-a---        10/28/2021  8:51 AM         0 New Text Document.txt

PS C:\Users\cdp_windows\Desktop> start .\cde.exe
PS C:\Users\cdp_windows\Desktop>

```

- Create a new text file and name it as *config.yaml*. Please note that while saving, choose the format as **All Files and NOT as Text Documents**.



- Add the following lines in this file.

```

user: <CDP_user>
vcluster-endpoint: <CDE_virtual_cluster_endpoint>

```

Here, **user** is the username you have been mapped in the excel sheet. Use the below url for **vcluster-endpoint**.

<https://v2wvrmwc.cde-ddsc4xhh.pse-work.dp5i-5vkq.cloudera.site/dex/api/v1>

- Open Powershell and run the below command to create an environment variable.

```
$env:CDE_CONFIG = "C:\Users\<path-to-config.yaml>"
```

- Run the below command for validation. You should see the path-to-config.yaml as the output.

```
ls $env:CDE_CONFIG
```

```
PS C:\Users\cdp_windows\Desktop> start .\cde.exe
PS C:\Users\cdp_windows\Desktop> $env:CDE_CONFIG = "C:\Users\cdp_windows\Desktop\config.yaml"
PS C:\Users\cdp_windows\Desktop> ls env:CDE_CONFIG
```

Name	Value
CDE_CONFIG	C:\Users\cdp_windows\Desktop\config.yaml

```
PS C:\Users\cdp_windows\Desktop>
```

- Run the below command to validate the configuration. Upon running it, you will be asked to provide the API password. Please enter the workload password as mentioned in the excel sheet.

```
.\cde job list
```

```
PS C:\Users\cdp_windows\Desktop> .\cde job list
API User Password: _
```

- If you get the below error related to certificate, please follow the next step to skip tls verification.

```
x509: certificate signed by unknown authority
[2] Could not verify authentication token
```

- Run the below command with the tls flag and enter the API password.

```
.\cde job list --tls-insecure
```

```
PS C:\Users\cdp_windows\Desktop> .\cde job list --tls-insecure
WARN: Plaintext or insecure TLS connection requested, take care before continuing. Continue? yes/no [no]: yes
API User Password:
```

- Once you enter the password, you should see all the jobs present in the virtual cluster.
- This marks the end of installation and configuration of CDE CLI. Now, head over to the next lab to trigger the jobs from CLI.

## Lab 6 - Run jobs using CDE CLI

You can use the CLI to create and update jobs, view job details, manage job resources, run jobs, and so on. Please use the link below to read more about the usage of CLI to manage CDE jobs.

<https://docs.cloudera.com/data-engineering/cloud/cli-access/topics/cde-cli-manage-jobs.html>

### Run a spark-scala job using CLI

As a first exercise in this lab, we will trigger a spark-scala job using the CDE CLI. Please note that you don't have to build a jar to submit the job to CDE.

- Locate and get the path of the script *Lab6A\_Data\_Extraction\_Avg\_Loan.scala* downloaded from the prerequisites step.
- Run the below command to submit this job to CDE.

```
cde spark submit /path/to/Lab6A_Data_Extraction_Avg_Loan.scala
```

```
[mmenia@MacBook-Pro workshop % cde spark submit /Users/mmenia/Workshop/Lab6A_Data_Extraction_Avg_Loan.scala
 1.2KB/1.2KB 100% [=====] Lab6A_Data_Extraction_Avg_Loan.scala
Job 103 submitted
Waiting for job run 103 to start... :
```

- Go to CDE UI and click on Job Runs. You will see a job submitted with the name `cli-submit-<username>-<temp-resource-id>`

Run ID	Job ↑	Type	User	Duration
103	cli-submit-wuser01-1635928294681	Spark	wuser01	1.3 MIN

- You can observe the logs and SparkUI for this Job Run.
- Please note that you are not creating this as a job in CDE. It will be an ad-hoc run without the need of registering it as a job.

### Trigger PySpark jobs using CLI

As a next exercise, we will trigger two jobs from the scripts we downloaded from the prerequisites step using CLI.

- On Terminal/Powershell, run the below command to trigger the first job using CLI.

```
cde spark submit <path-to-Lab6B1_PreSetupDW.py>
```

- You will see a job getting triggered from the terminal output. You can check the job status from the UI by following the similar approach followed in the previous exercise.
- Go to CDE UI and click on Job Runs. You will see a job submitted with the name `cli-submit-<username>-<temp-resource-id>`
- Run the below command to trigger the second job.

```
cde spark submit <path-to-Lab6B2_EnrichData_ETL.py>
```

- This completes the exercise of Lab6.

## Lab 7 - Data Lineage and Auto-Scaling

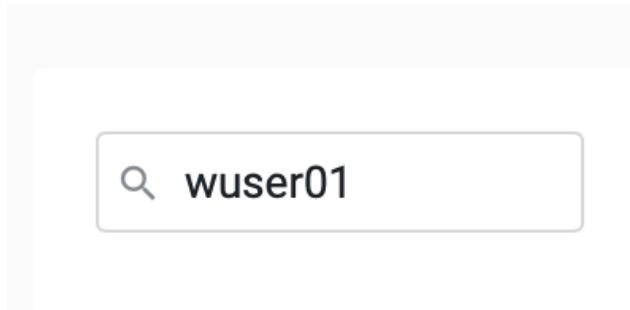
---

In this lab, you will go through the data lineage of the two use cases that we worked on. Additionally, you will also see the auto-scaling capabilities of CDE service with the rising demand for compute resources.

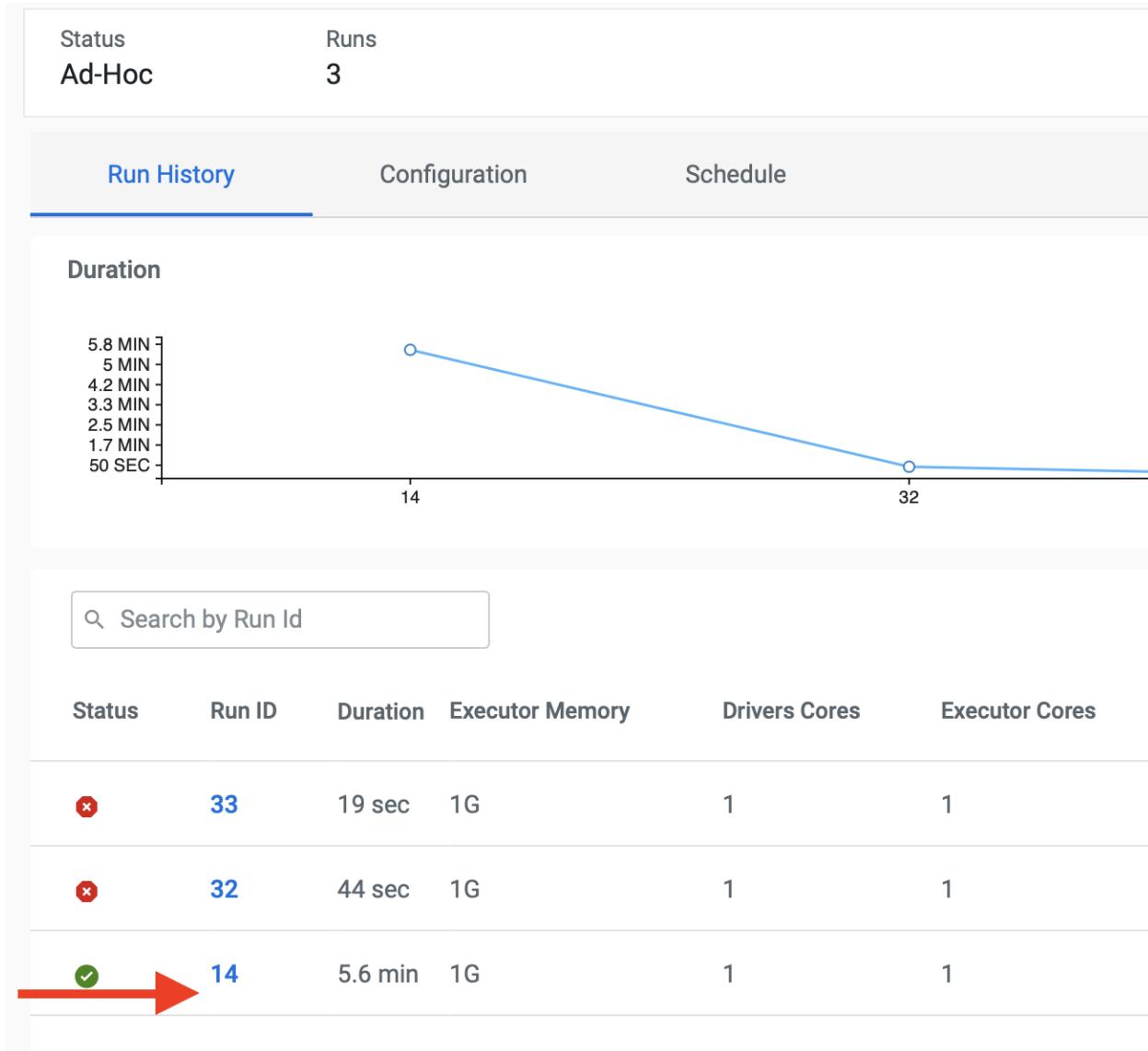
### Data Lineage using Atlas

- In the CDE UI, click on the Jobs tab. Go to the job `<username>_Lab3B3_Create_Report`s that you have created in the Lab2.
- To get the jobs, please filter the jobs with your username.

### Jobs



- In **Run History** tab, click on the successful Run ID i.e., the one with the green tick mark.



- Click on **Atlas** under Lineage.

## Job Runs / 14



- Click on the execution that you see in the list.

Name	Owner	Description	Type	Classifications	Term
execution-4			spark_process	<a href="#">+</a>	<a href="#">+</a>

- Click on **Lineage** to observe the Data Lineage for this job.

Terms: [+](#)

Properties

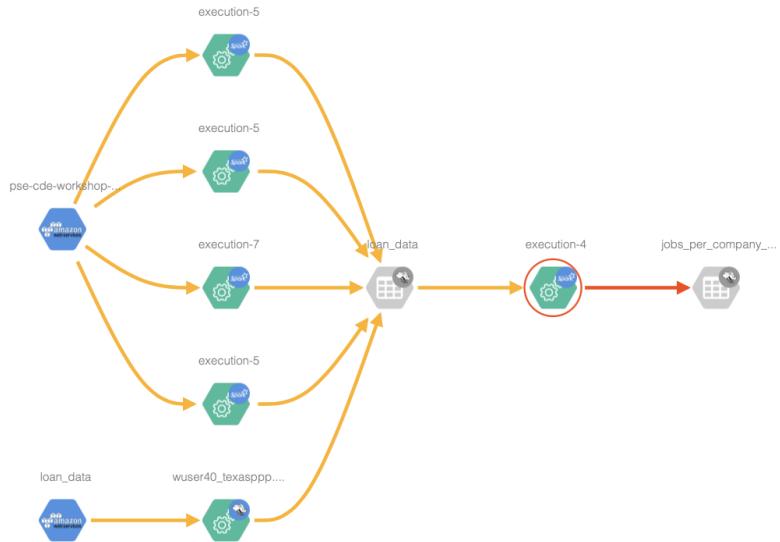
Lineage

Relationships

Classifications

Audits

○ Current Entity ⏳ In Progress → Lineage → Impact

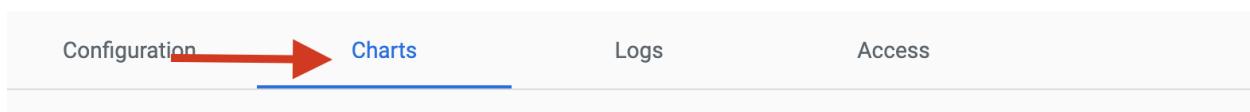


- Click on each entity to understand how the data is flowing from source to consumption.
- Follow the same steps for the last job you submitted using CLI.

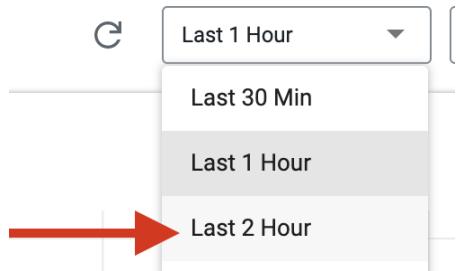
## Auto-scaling in CDE

- As a last step, we want you to witness the auto-scaling capabilities of CDE. At the start of the lab, you might have noticed the cpu and memory consumption of the virtual cluster. Please check the dashboard now to see how it has scaled up based on the demand experienced.
- On the CDE home page, click on the **Cluster Details** on the virtual cluster.
- Click on the **Charts** tab.

CLI TOOL : [API DOC](#) [JOBS API URL](#) [GRAFANA CHARTS](#) [AIRFLOW UI](#)



- Set the filter to **Last 2 Hour** and observe the varying load on cpu and memory.



- Click on **Grafana Charts** to view another set of metrics of the virtual cluster.

CLI TOOL : [API DOC](#) | [JOBS API URL](#) → [GRAFANA CHARTS](#) | [AIRFLOW UI](#)

- This marks the end of the overall CDE Hands-on Workshop session.

## **THANK YOU VERY MUCH FOR YOUR PARTICIPATION**

Please use the link below to provide your feedback on this workshop. It will be very valuable for us to add/modify/improve on the content and delivery.

<https://forms.gle/Buac2FzqzKtjJRWp6>