CLOUDERA

CDE Workshop
Student Guide

Introduction

This document aims to introduce to 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:

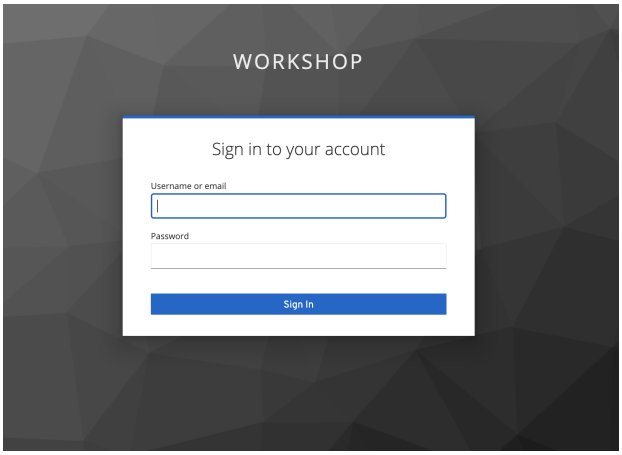
- You will be given an active CDE service running in an existing/registered CDP environment in a given tenant.
- You will have a virtual cluster with the given configuration that serves as compute for the spark workload.
- You will run sample spark jobs as ad-hoc jobs.
    - PySpark
    - Spark-scala
- 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

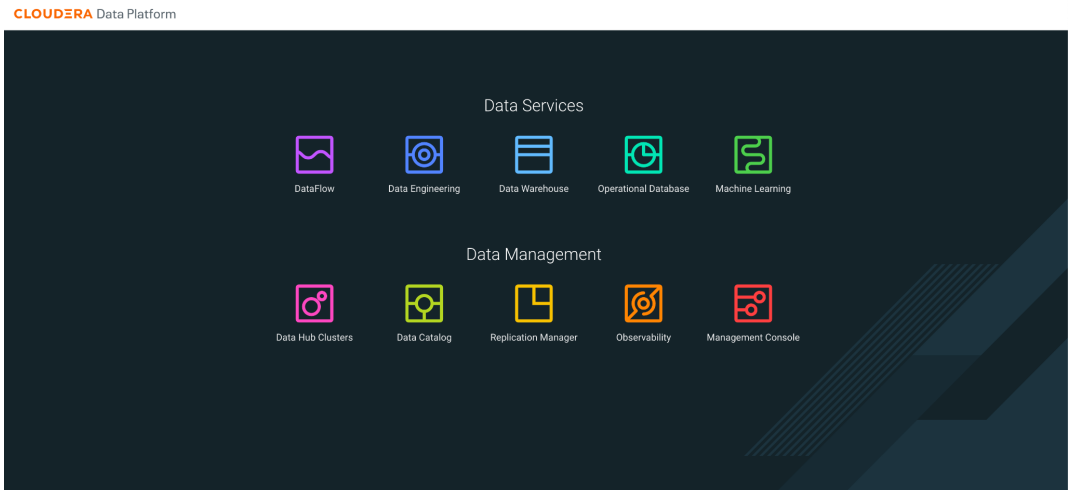
Verify access to the environment

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

  <Will be shared by the instructor at the start>



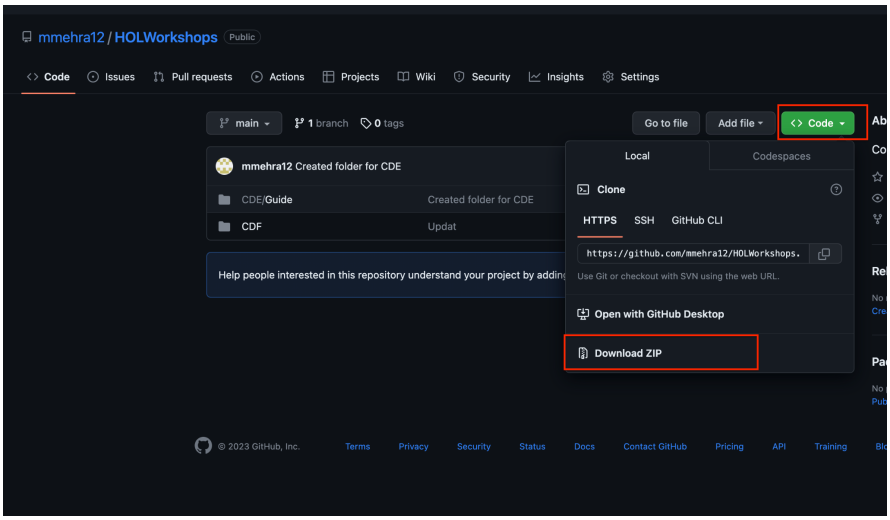
- You should land on the CDP Console as shown below.



Download the resources

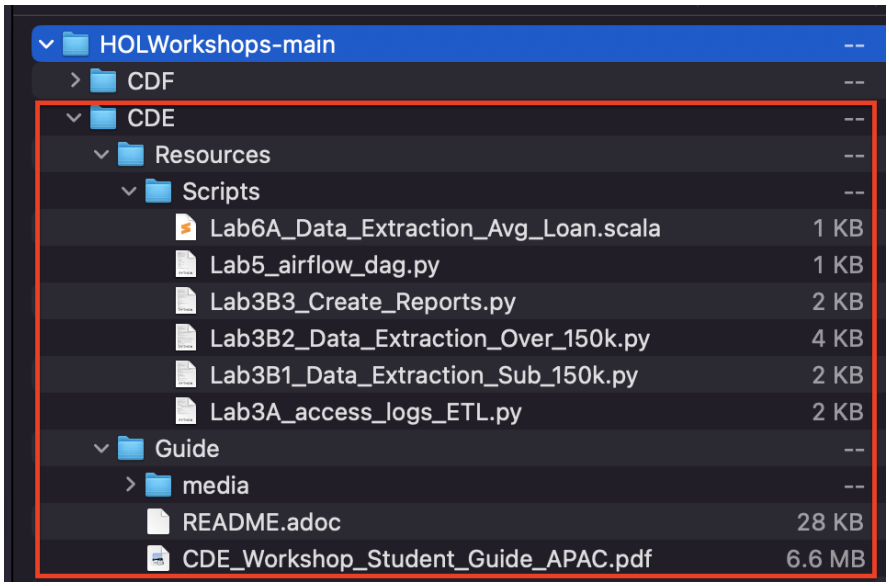
There are two ways in which you can access the scripts/resources.

- Download the zip file from the GitHub repository. https://github.com/mmehra12/HOLWorkshops

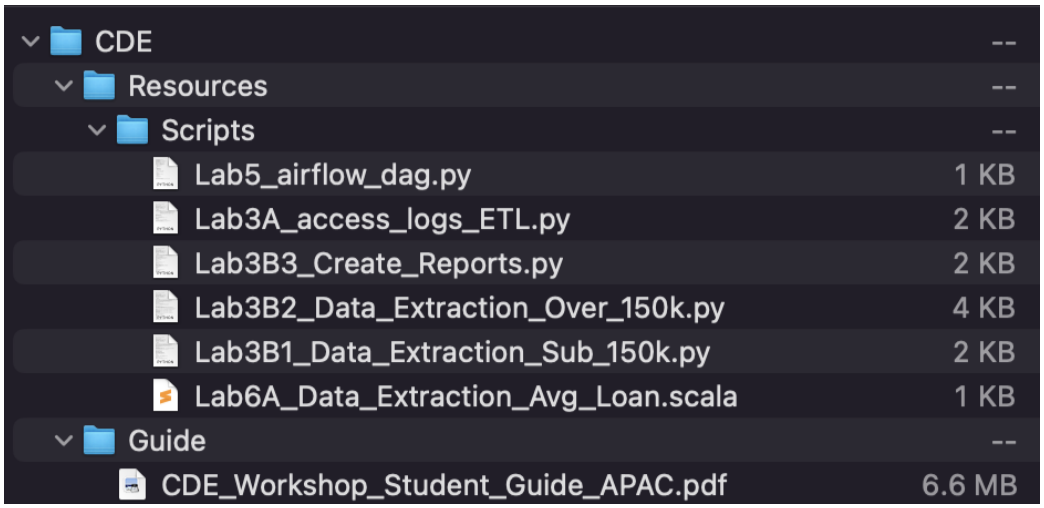


After decompressing the ZIP file the folder structure should look something like this

Note : We will use the CDE folder for this session, you can ignore the CDF content.

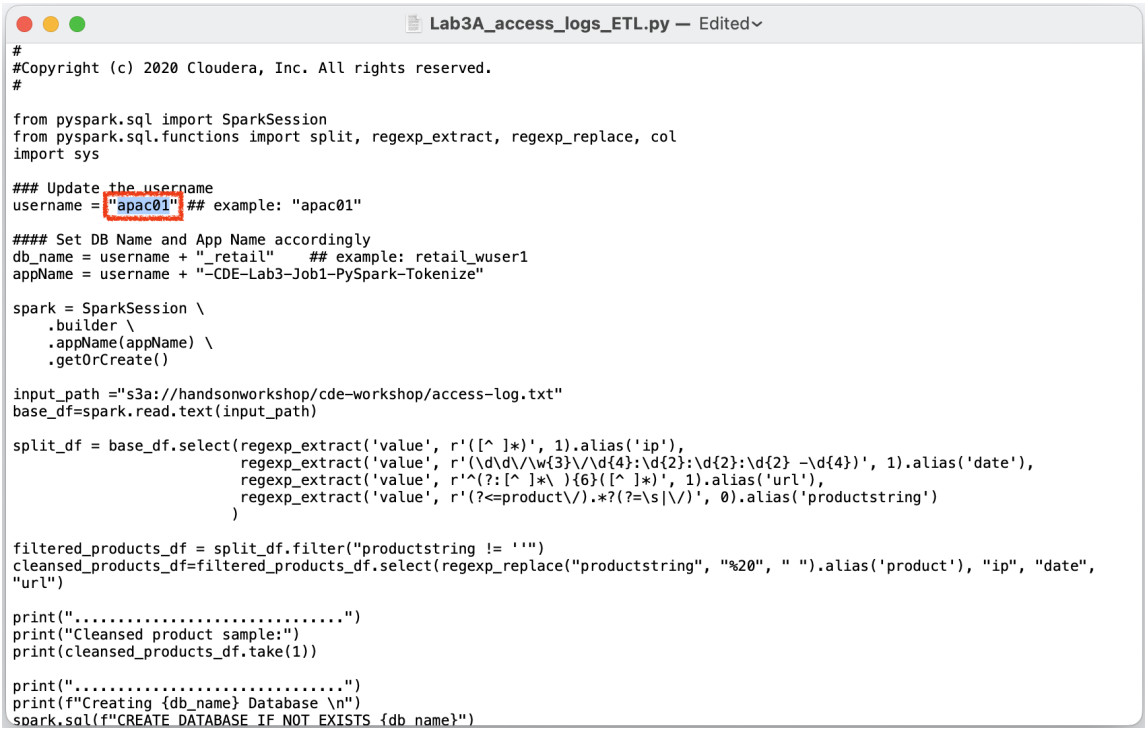


- The resources were also sent to you on your registered email an hour before the event. Please download the zip file attached to the email. After decompressing the ZIP file the folder structure should look something like this.



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

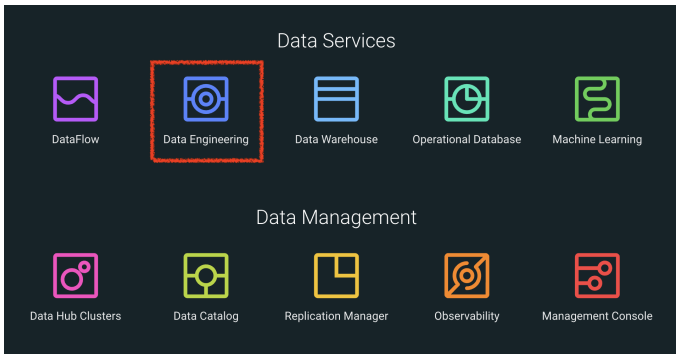
Cloudera Data Engineering (CDE) is a serverless service for Cloudera Data Platform that allows you to submit jobs to auto-scaling virtual clusters.

The CDE service involves several components:

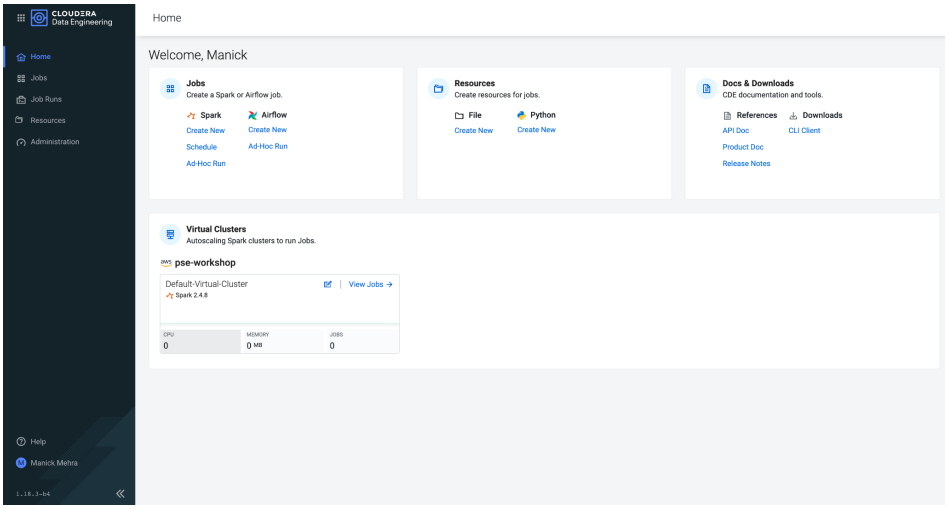
- **Environment**
  - A logical subset of your cloud provider account including a specific virtual network.
- **CDE Data Service**
  - The long-running Kubernetes cluster and services that manage the virtual clusters. The CDE service must be enabled in an environment before you can create any virtual 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:

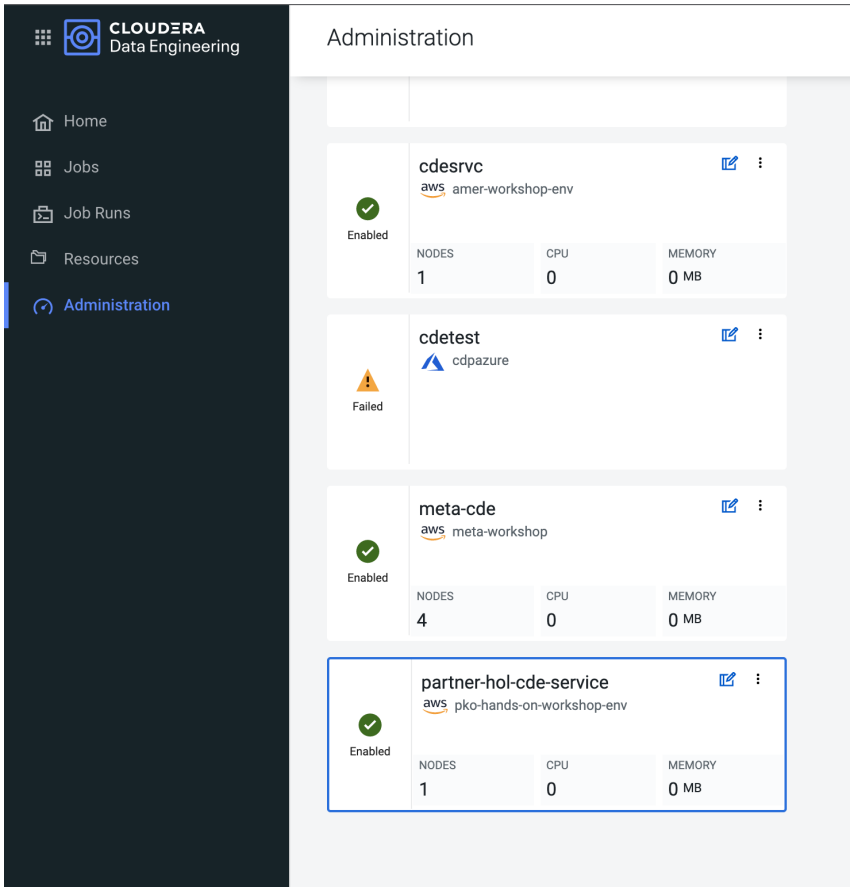
- Go to the CDP console and click on Data Engineering.



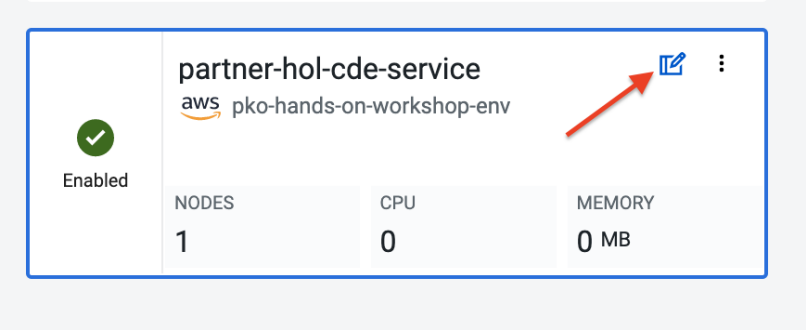
- You will see the CDE homepage.
  Note : If the page load takes a while, you can move to the next step, we can come back to this later

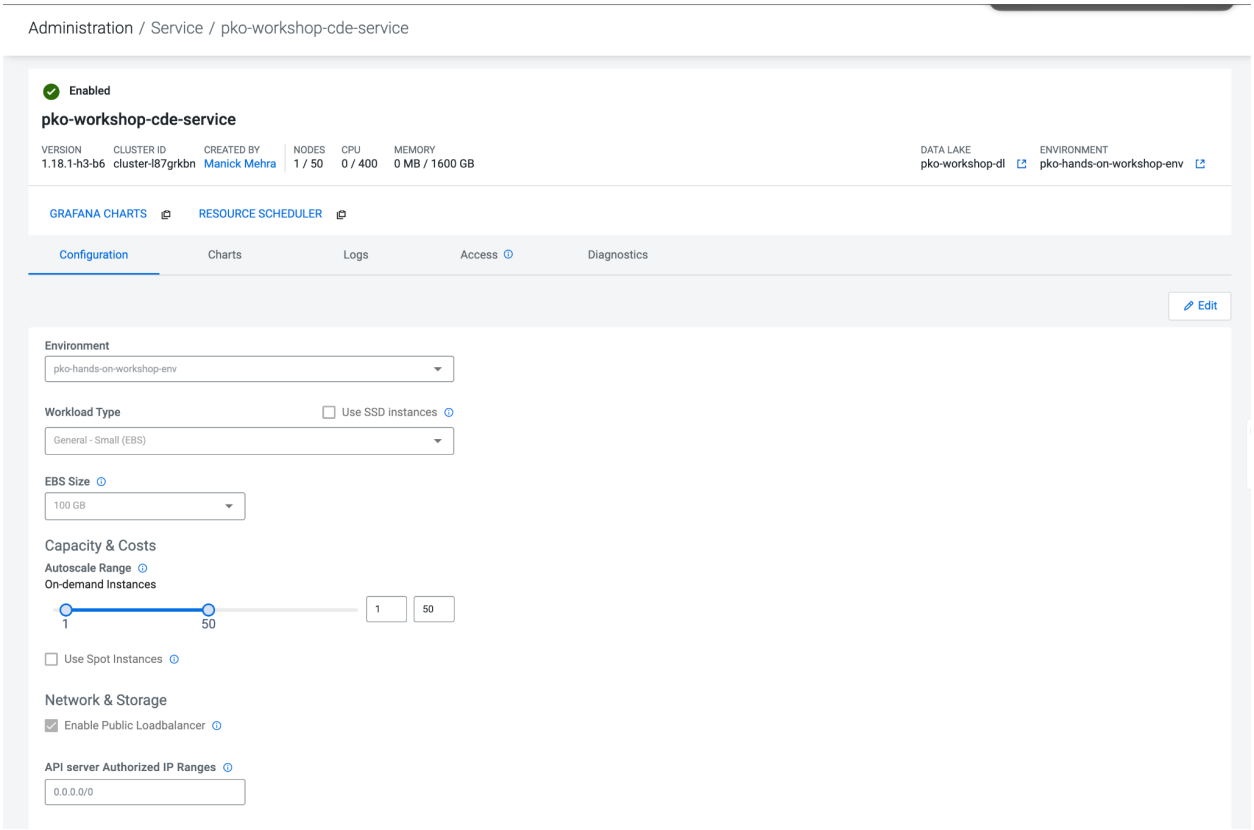


- We should have a CDE service running which we will use for this workshop. To check this Select the **ADMINISTRATION** option on the left menu on your screen. You should be able to see all the CDE Service and their status.



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





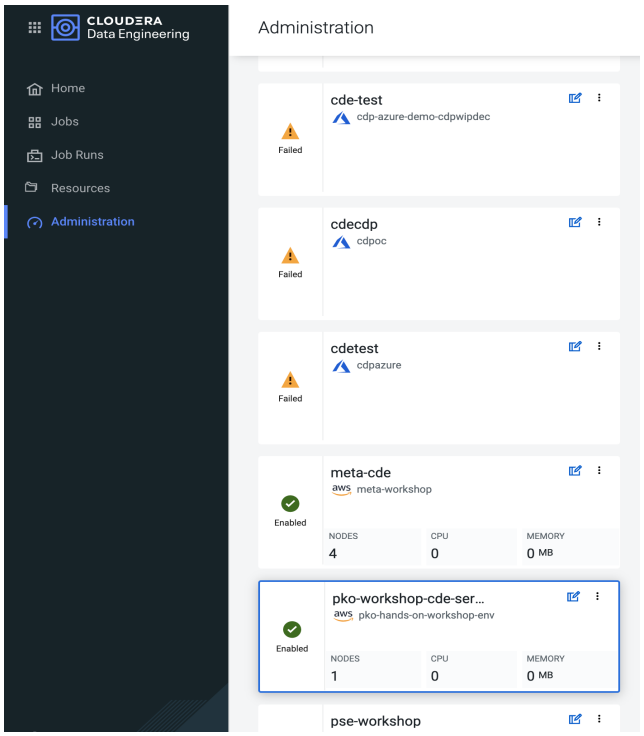
- Click on each tab and go through all the details related to the CDE service.

- Once done, click on the **Home** on the left tab to go back to the CDE home page.This page shows us the active CDE services and the associate clusters. Let's start with accessing the virtual cluster that is assigned to you.

ACCESSING THE VIRTUAL CLUSTER

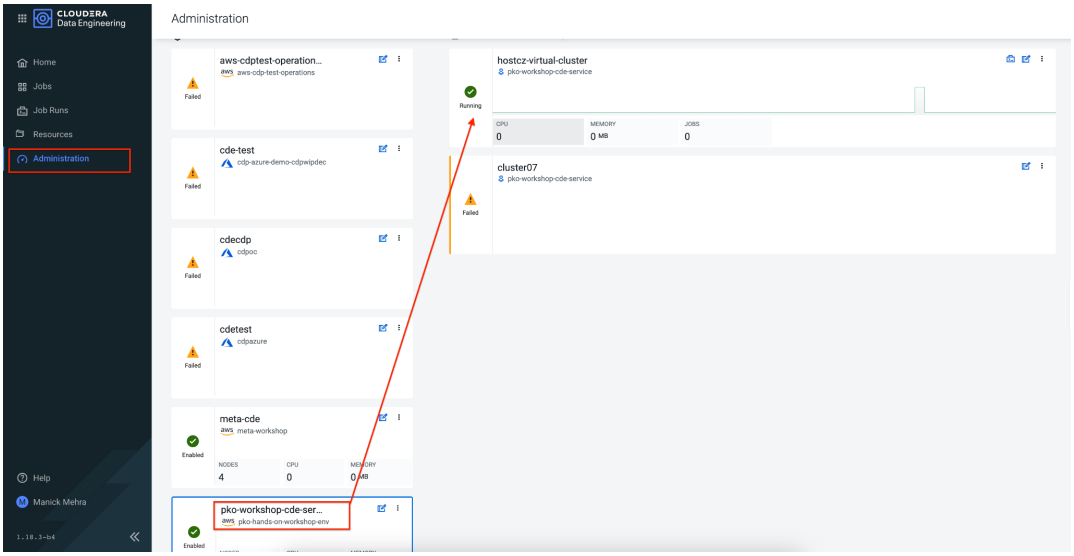
Step 1 : Select the appropriate CDE Service

Go to the Administration page and select your CDE Service (In our case **partner-hol-cde-service**)



Step 2 : Virtual cluster selection

a. Select the CDE Service and click on the virtual cluster that was assigned to you.


CLOUDERA
Data Engineering
Home
Jobs
Job Runs
Resources
Administration
Help
Manick Mehra
Administration
aws-cdptest-operation...
aws-cdp-test-operations
Failed
cde-test
cdp-azure-demo-cdpwipdec
Failed
cdecdp
cdpoc
Failed
cdetest
cdpazure
Failed
meta-cde
meta-workshop
Enabled
NODES
4
CPU
0
MEMORY
0 MB
pko-workshop-cde-ser...
pko-hands-on-workshop-env
Enabled
hostcz-virtual-cluster
pko-workshop-cde-service
Running
CPU
0
MEMORY
0 MB
JOBS
0
cluster07
pko-workshop-cde-service
Failed

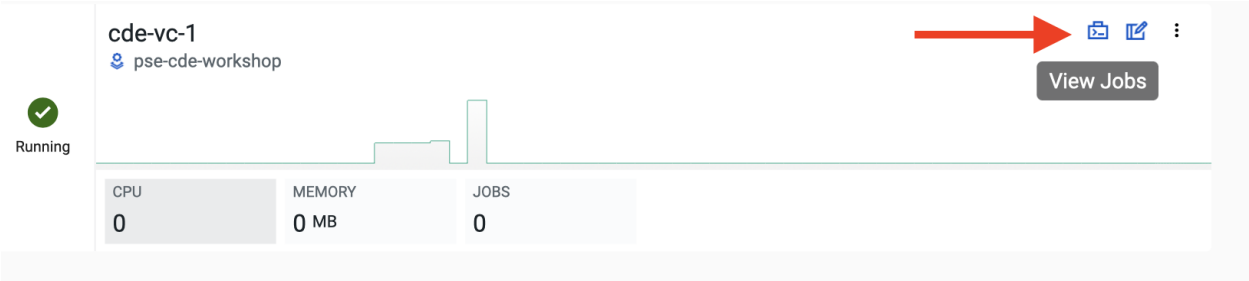
Lab 2 - Create and trigger ad-hoc Spark jobs

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.

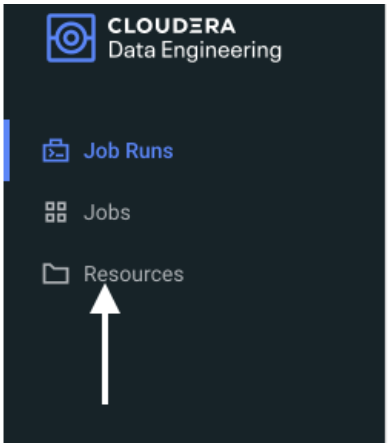
1. Log Data Cleansing using Spark
2. Analyze the Paycheck Protection Program Data
    a. Report 1: Breakdown of all cities in Texas that retained jobs
    b. Report 2: Breakdown of company type that retained jobs
3. PySpark job to enrich your data using an existing data warehouse

Resource Creation

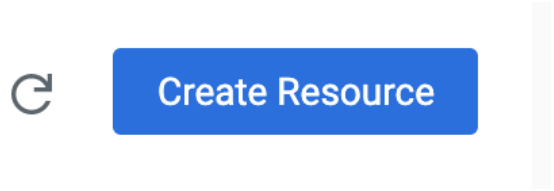
- On the virtual cluster **Cluster Name** : *<username>-virtual-cluster* [Virtual cluster created in Lab 1] tab**,** click on view jobs. This will open a new page with details of the Job Runs, Jobs, and Resources.



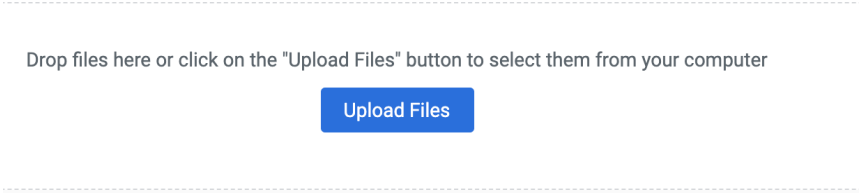
- In the left pane, click on the **Resources** tab.



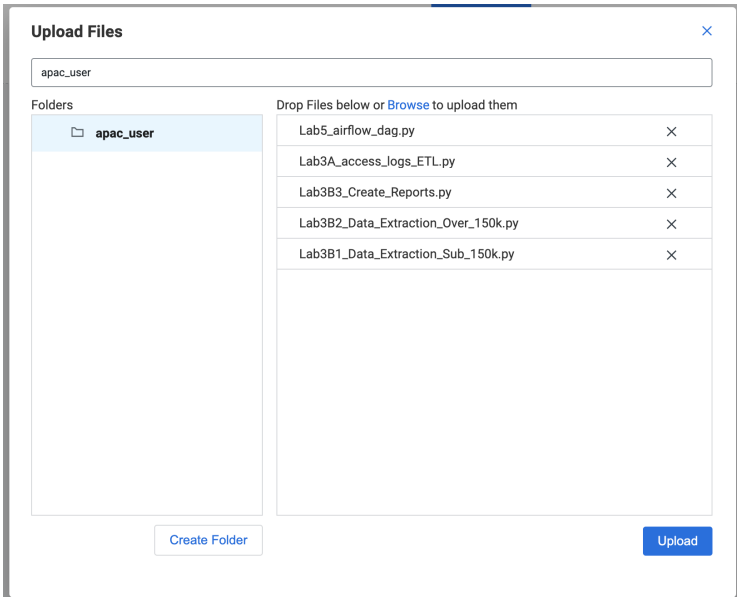
- You will get the **Resources** page to the right. Click on **Create Resource**.



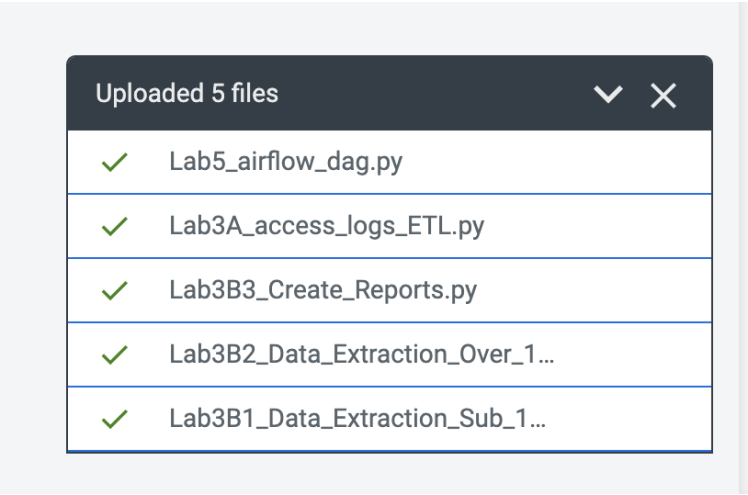
- Give a unique name(username-resources) and create the resource. This acts as your repository for storing all the scripts and dependencies.
- Once it is created, you will get an option to upload the files as shown below.



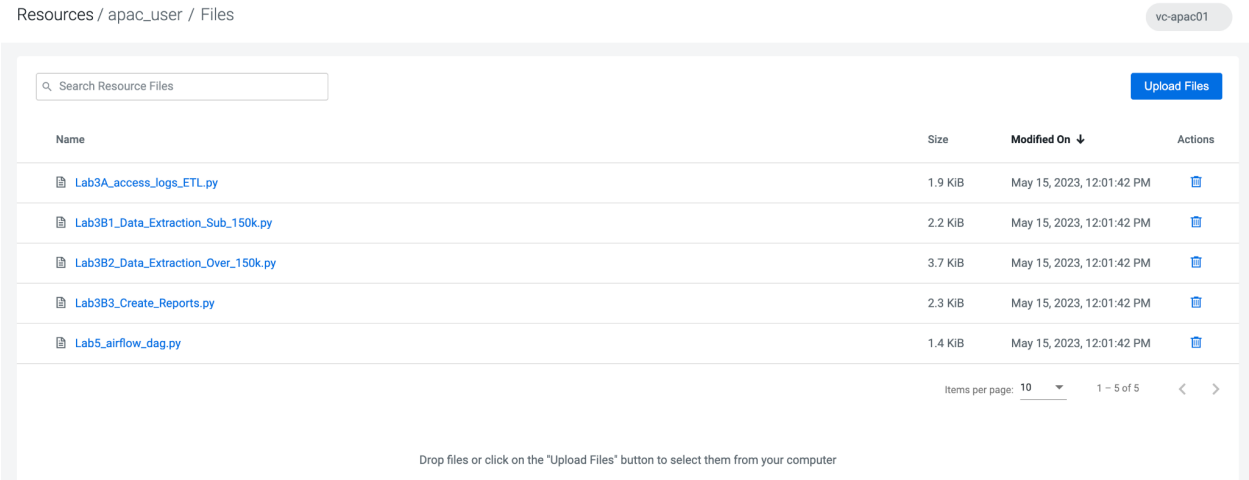
- Click on **Upload Files** and select all the scripts downloaded from the prerequisites step. (**Please upload only .py files**). Click on Upload



- You will get a pop-up with all the files uploaded to your resource.

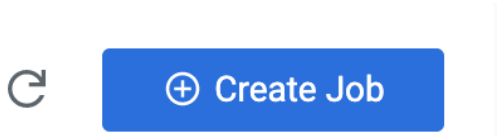


- Validate if all the five *.py* files are present in your resource. We are now ready to create jobs using these resources.



Job Creation

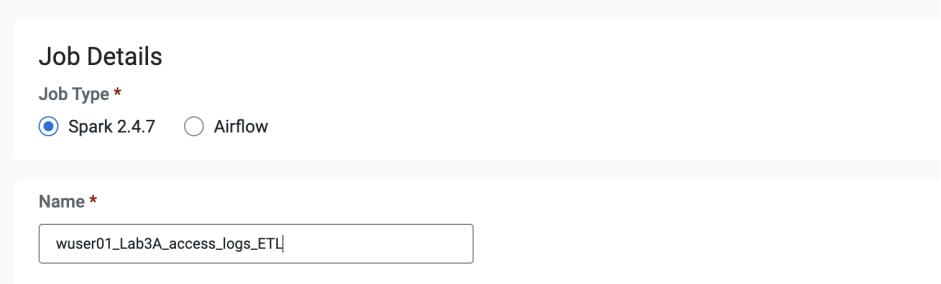
- We will now create the first job with the script ***Lab3A_access_logs_ETL.py***.
- In the left pane, click on **Jobs.**
- You will get the **Jobs** page to the right. Click on **Create Job**.



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

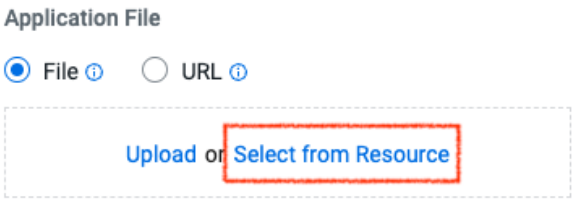
<username>_<script_name_without_py_extension>

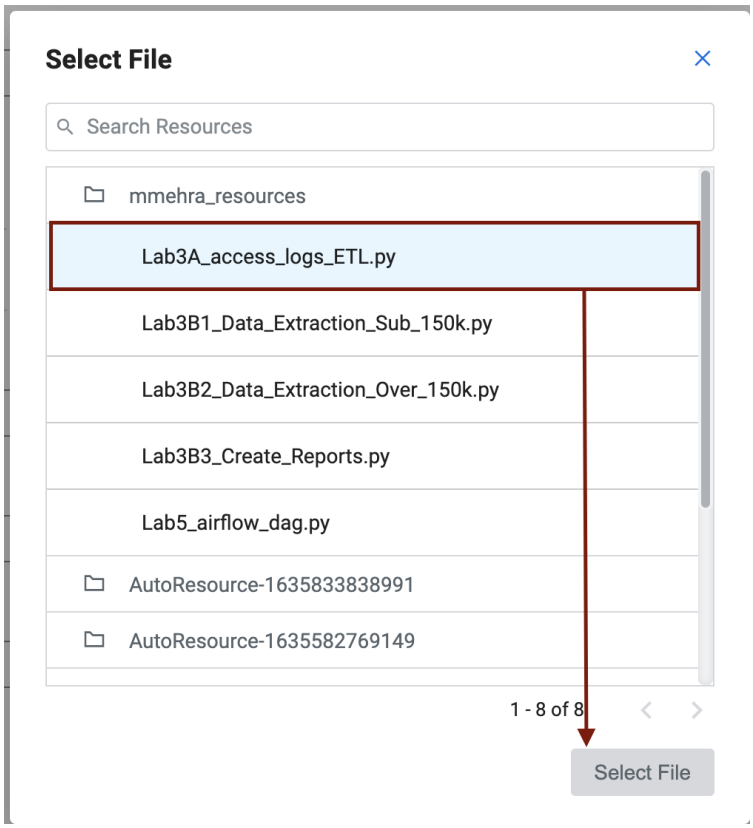
Eg:- For apac01, job1 name would be **apac01_Lab3A_access_logs_ETL**



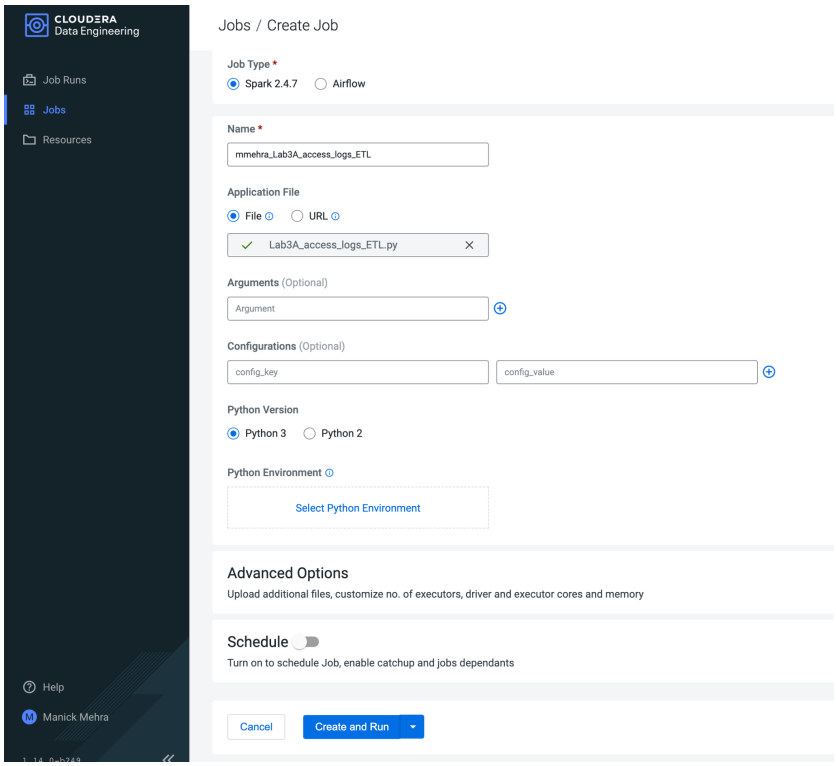
- **As this is a shared environment, please name the jobs with your username so that it helps in differentiating yours from others' jobs.**

- In **Application File**, click on **Select from Resource** and select the file ***Lab3A_access_logs_ETL.py*** from your resource(<username>-resources).

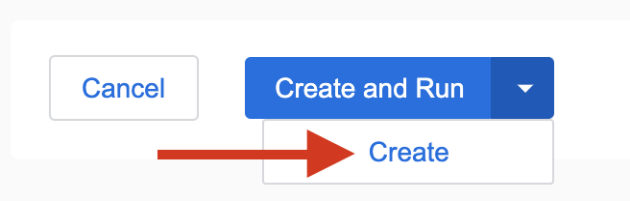




- Ignore the remaining configuration options. Do not enable the schedule now. This is how it should finally look like.



- Click on the drop down option and click on **Create**. (do not click Create and Run)

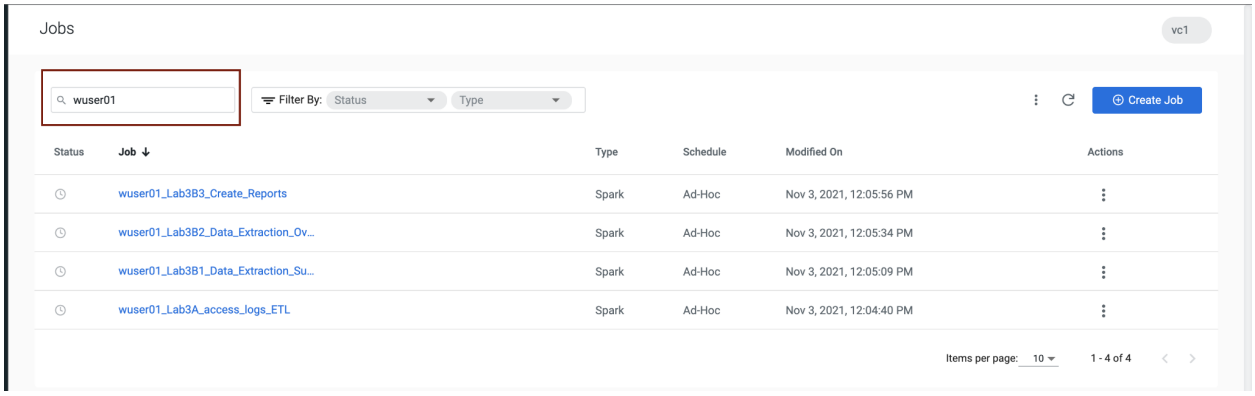


- Similarly, create three other jobs with the same naming conventions. Please refer to the table below to confirm you are creating exactly the same.

For **apac01:**

Jobs	Job Name	Script Used
Job1	apac01_Lab3A_access_logs_ETL	Lab3A_access_logs_ETL.py
Job2	apac01_Lab3B1_Data_Extraction_Sub_150k	Lab3B1_Data_Extraction_Sub_150k.py
Job3	apac01_Lab3B2_Data_Extraction_Over_150k	Lab3B2_Data_Extraction_Over_150k.py
Job4	apac01_Lab3B3_Create_Reports	Lab3B3_Create_Reports.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.



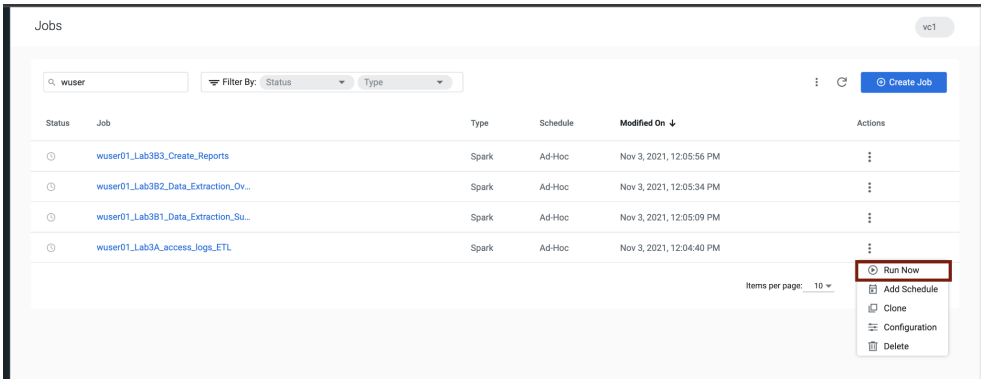
- Observe the type of the job is set to Spark and for schedule, it is Ad-hoc.

Triggering the jobs

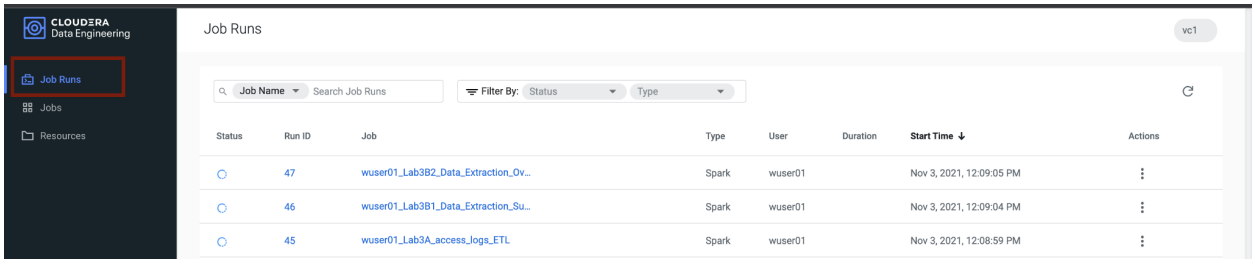
- You need to trigger the jobs in the following order
  - JOB 1 : apac01_Lab3A_access_logs_ETL
  - JOB 2 : apac01_Lab3B1_Data_Extraction_Sub_150k
  - JOB 3 : apac01_Lab3B2_Data_Extraction_Over_150k
  - JOB 4 : apac01_Lab3B3_Create_Reports(Run once JOB 2 and JOB 3 have completed successfully)

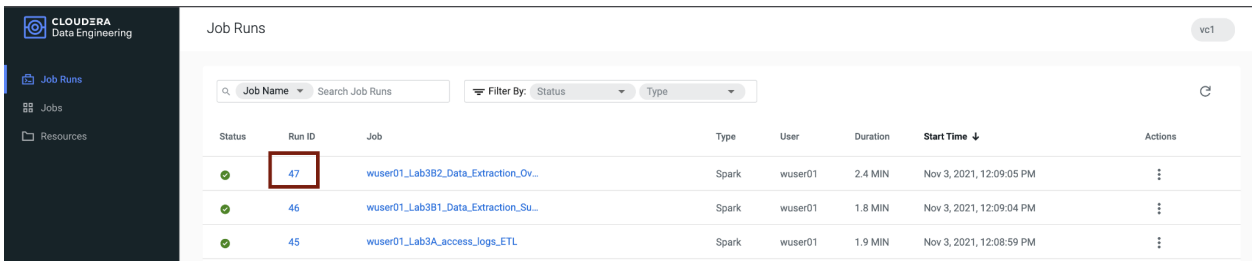
  NOTE : JOB 1, JOB 2 and JOB 3 can be triggered one after the other. JOB 4 should be executed after the successful completion of JOB 2 and JOB 3

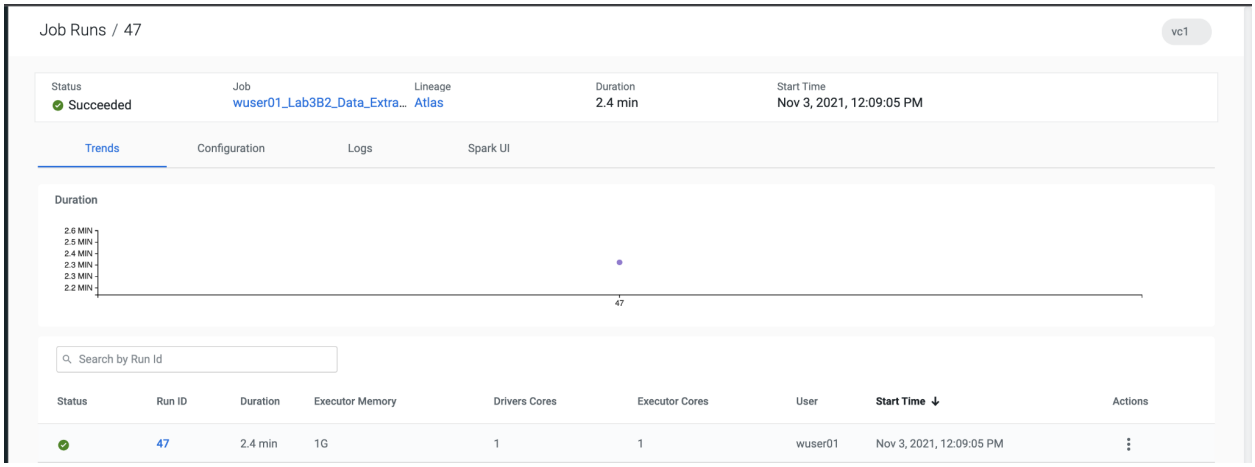
- To trigger the job, go to the **Jobs** tab, click on the 3-dotted icon, and click on **Run Now.**



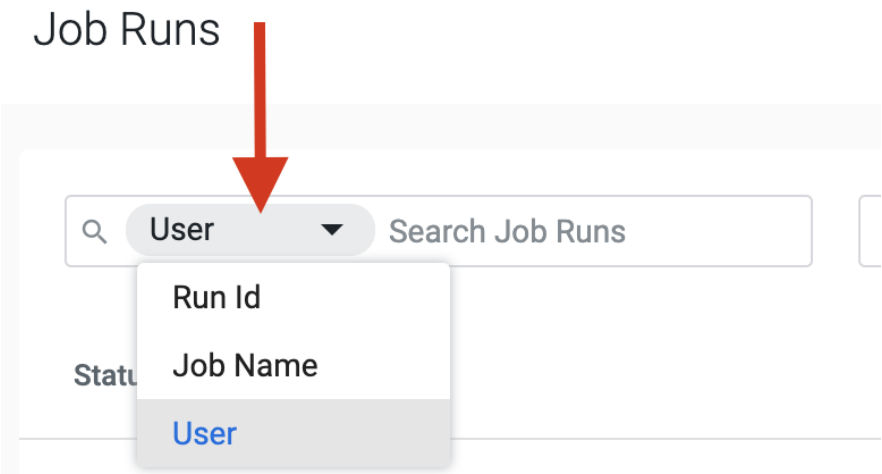
- To check the job logs, click on **Job Runs** and select the **ID** against the job that you have triggered.



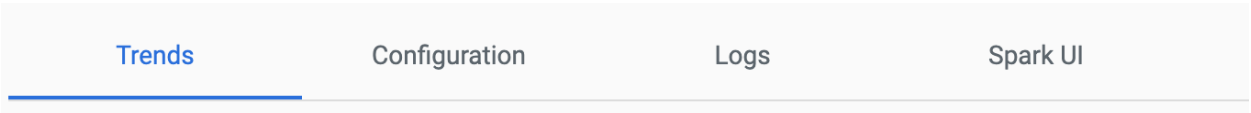




- For simplifying the job selection, you can choose the **User** filter and add your username and hit enter. You will see the list of jobs triggered by you.



- 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.

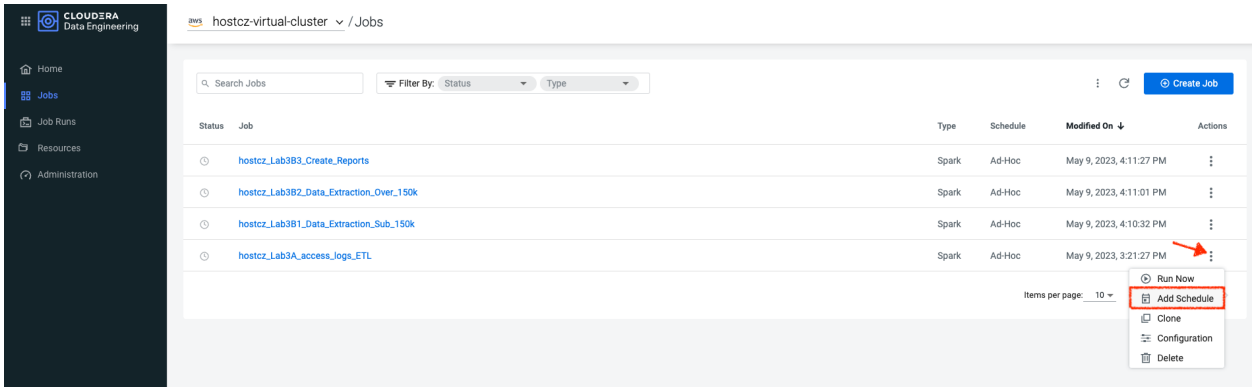


Lab 3 - Add schedule to the ad-hoc Spark jobs

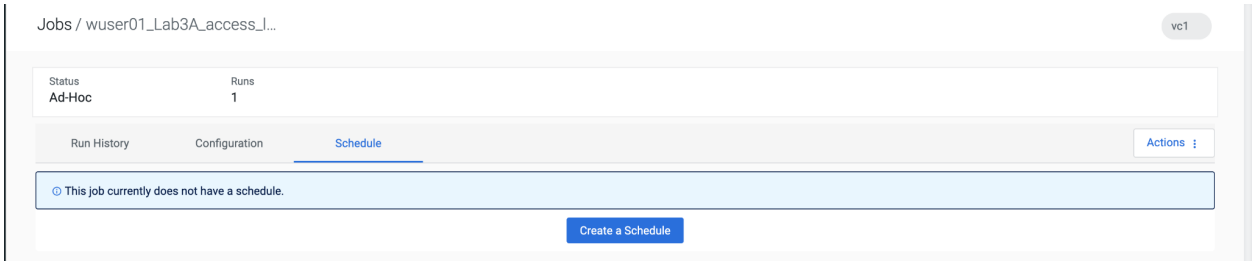
In this lab, we will add a schedule to a job created as part of the previous lab.

- 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**.

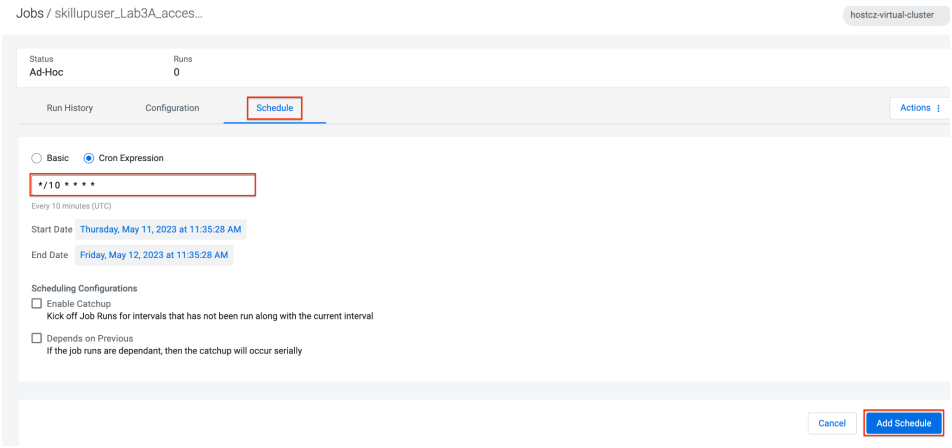


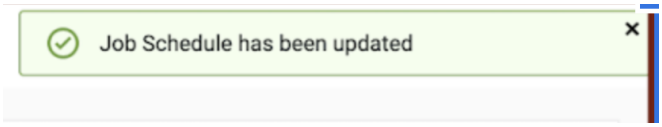
- You will land in the **Job Schedule** page. Click on **Create a Schedule**.



- 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.

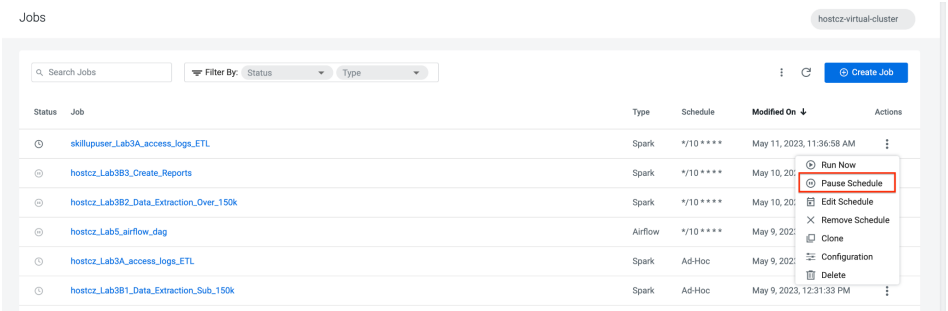


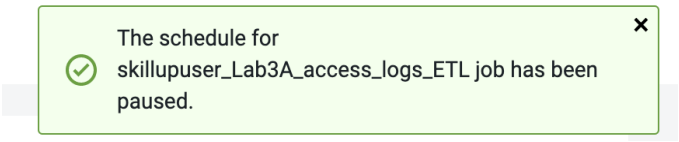


- You can repeat the same process for the other jobs as well.
  - JOB 1 : Run every 10 mins
  - JOB 2 : Run every 10 mins
  - JOB 3 : Run every 10 mins
  - JOB 4 : Run every 30 mins

- We do not have to wait for the jobs to get triggered as per the schedule. The idea was to understand how Ad-Hoc jobs are scheduled. We can continue with the next steps

- **Please PAUSE 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 and select **Pause schedule**. [Do this for all jobs]





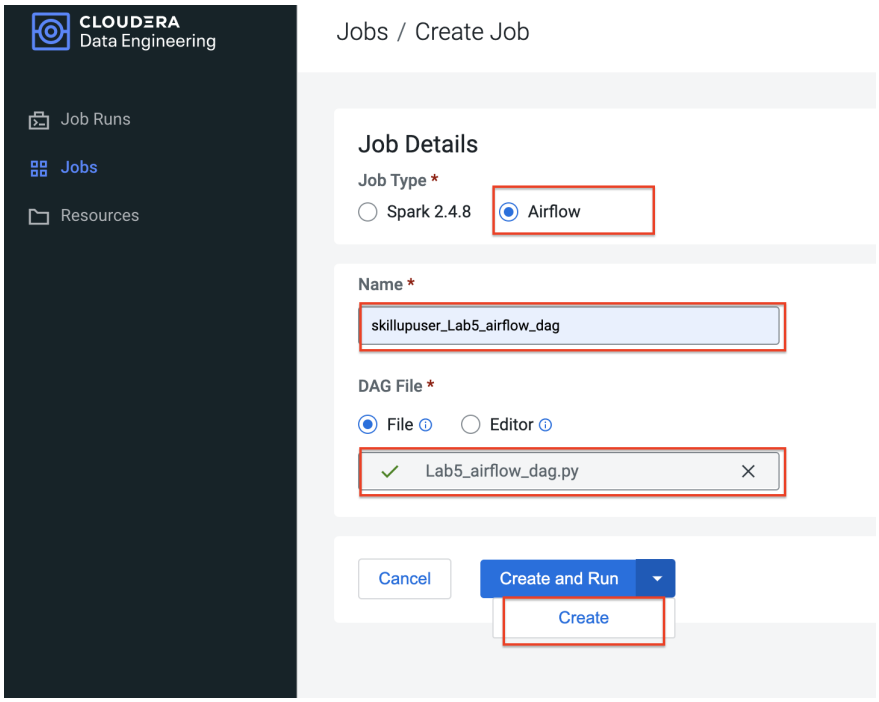
Lab 4 - Orchestrate a set of jobs using Airflow

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.

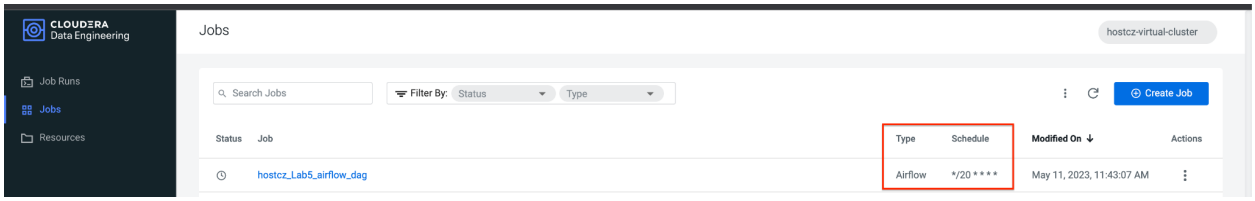
  JOB NAME : <username>_Lab5_airflow_dag
  Example : For user ***apac01*** the job name will be, ***apac01_Lab5_airflow_dag***

- Click on **Create.**



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

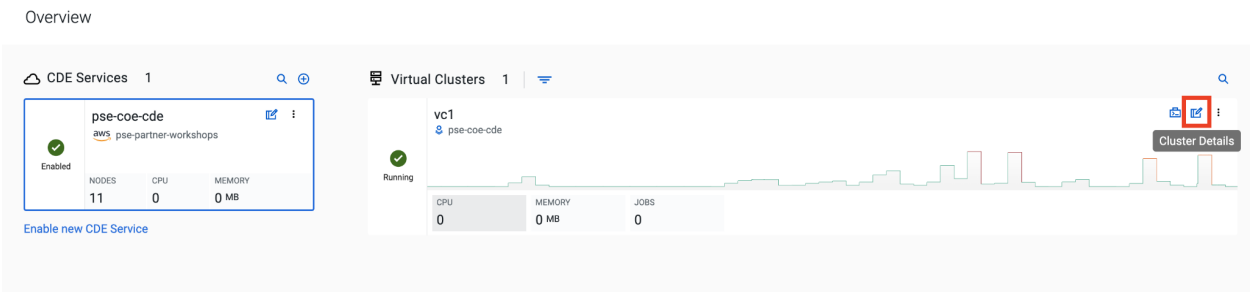
  Job



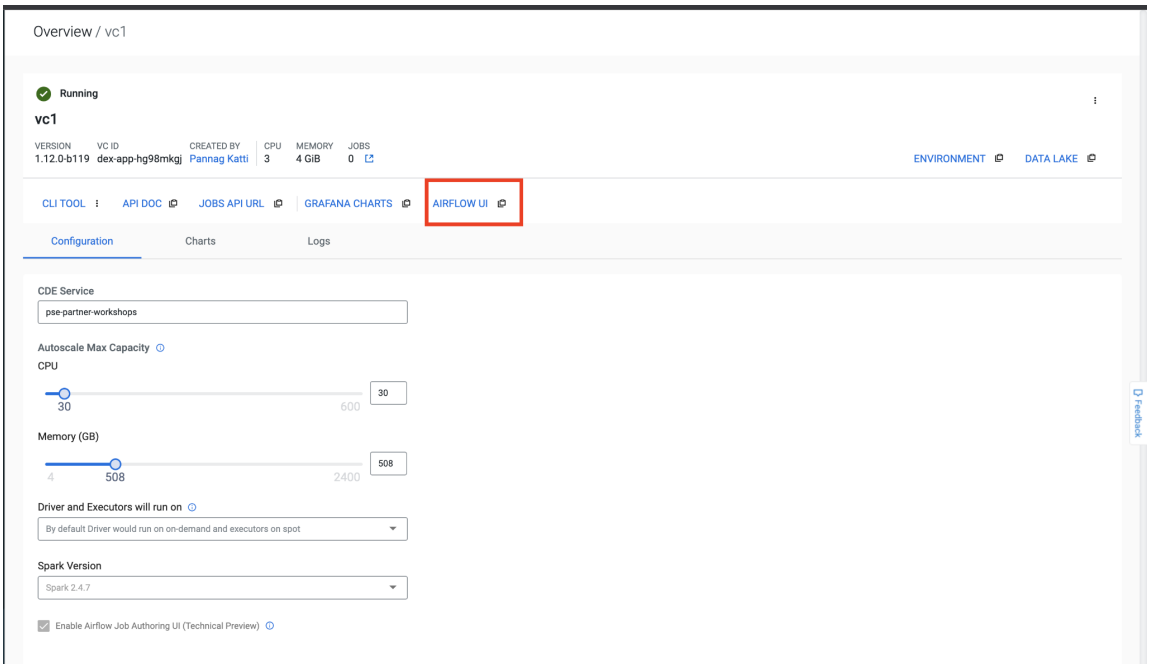
DAG File

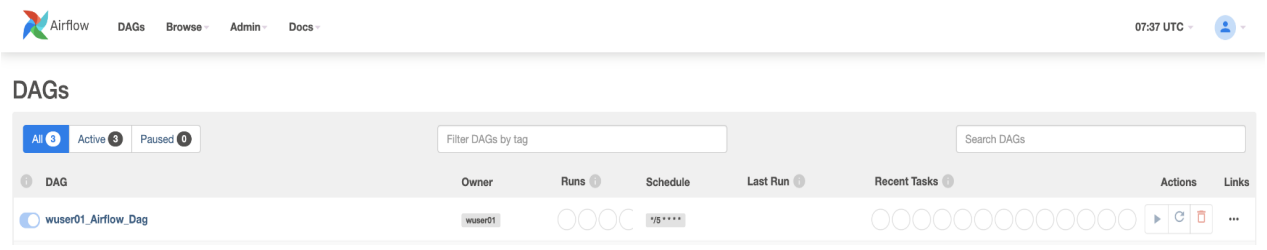
```
## Update the username
owner = "<ENTER YOUR USER NAME HERE>" # Example: "apac01"

DAG_name = owner + "_Airflow_Dag"
job_name_1 = owner + "_Lab3B1_Data_Extraction_Sub_150k"
job_name_2 = owner + "_Lab3B2_Data_Extraction_Over_150k"
job_name_3 = owner + "_Lab3B3_Create_Reports"

default_args = {
    'owner': owner,
    'retry_delay': timedelta(seconds=5),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 0
}

dag = DAG(
    DAG_name,
    default_args=default_args,
    start_date=datetime(2023,5,3),
    end_date=datetime(2023,5,18),
    schedule_interval='*/20 * * * *',
    catchup=False,
    is_paused_upon_creation=False
)


start = DummyOperator(task_id='start', dag=dag)

Data_Extraction_Sub_150k = CDEJobRunOperator(
    task_id=job_name_1,
    retries=3,
    dag=dag,
    job_name=job_name_1
```

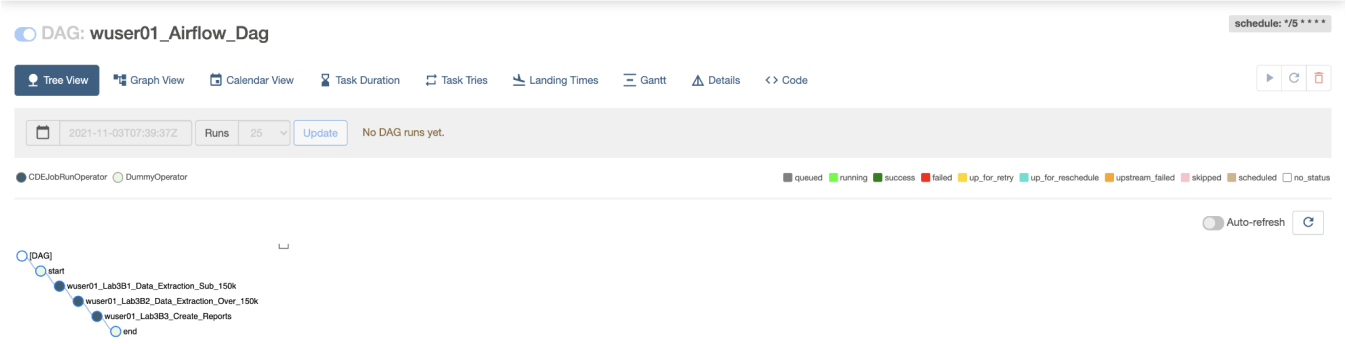
- Go to the Virtual Cluster you are using and click on **Cluster Details**.



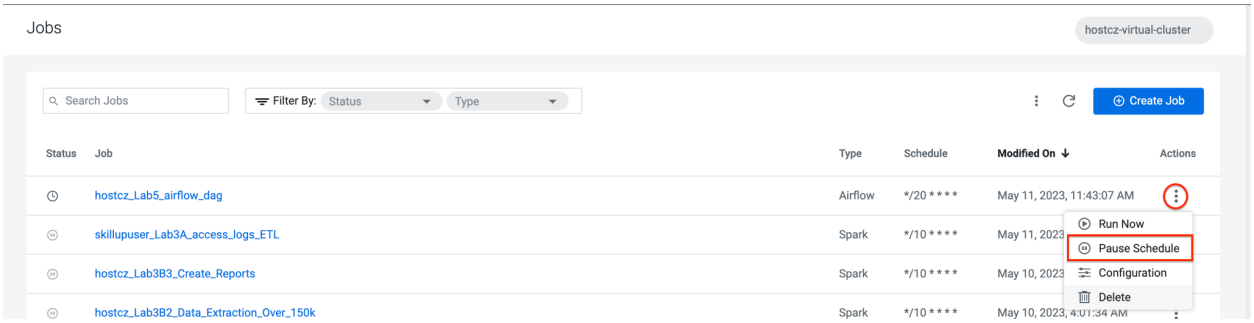
- Click on **Airflow UI** and observe the schedule created for your job.

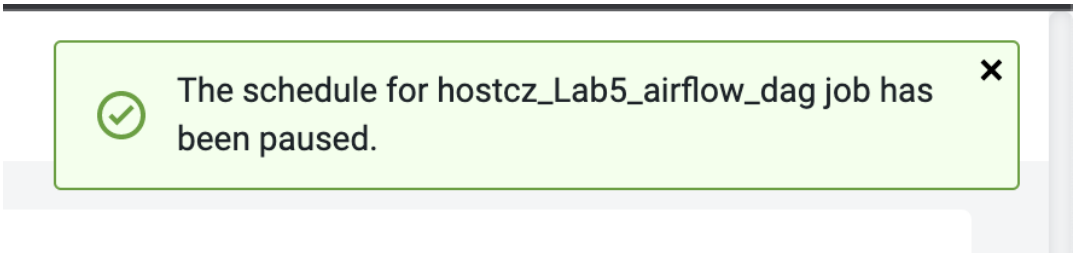




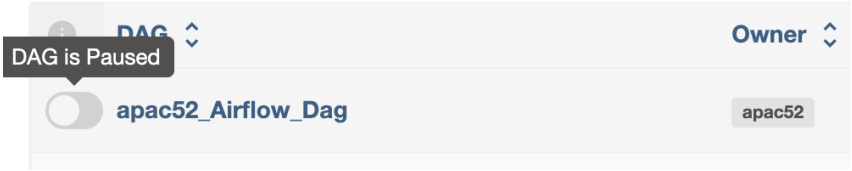


- Once the job has run successfully, we need to edit the job to **pause** the schedule.
- Click on the Jobs tab and locate the airflow job that you have just created.
- Next to the job, click on the 3 dots and click on **Pause Schedule**.



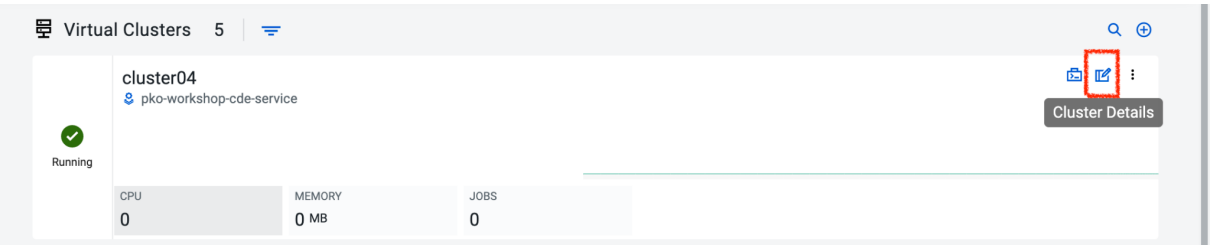


- You can go to the AirFLow UI again and see that the Job is now in Paused State

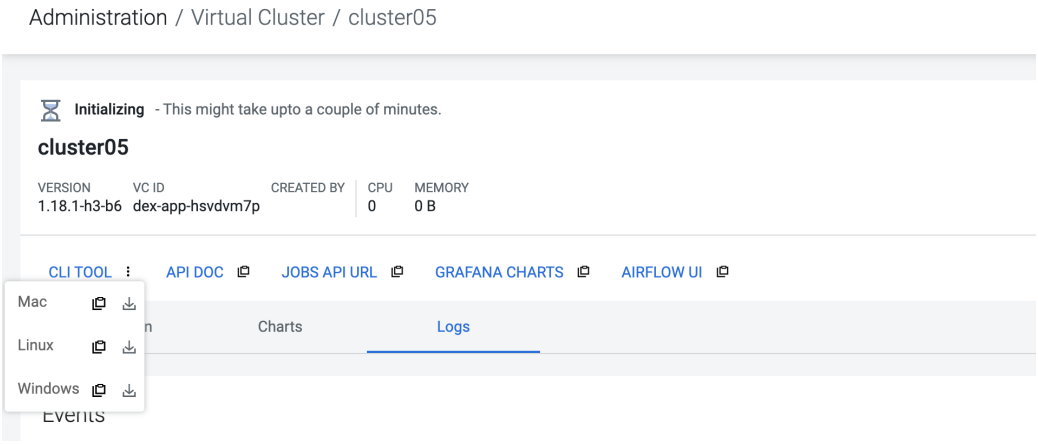


Lab 5 - Install and Configure CDE CLI

- 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.
- The CLI executable can be downloaded from the virtual cluster.
    - **Step 1** : Go to the **Cluster Details** of the virtual cluster where you are creating your job



    - **Step 2** : Click on CLI TOOL to download the executable based on your operating system.

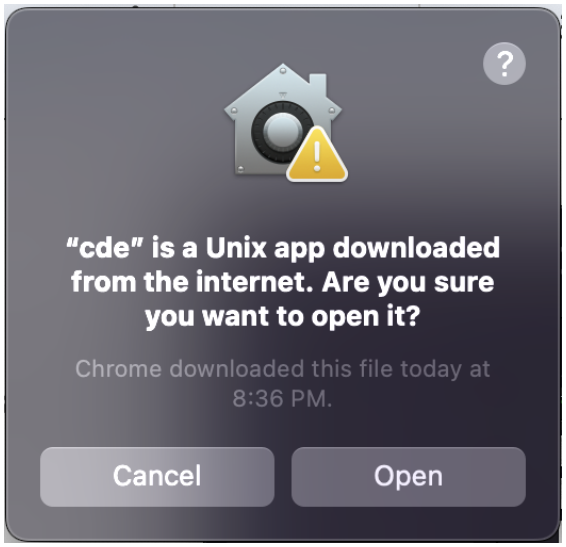


For Mac users:

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

```
chmod +x /path/to/cde
```

- Go to the folder where the executable is present. Right click and select "Open with" -> Terminal . You will get the below message



- Click on **Open**
- Once done, you will get the following window and message

```
      --credentials-profile string   CDP credentials profile name (default "defa
ult")
  -h, --help                         help for cde
      --hide-progress-bars           hide progress bars for file uploads
      --region string                CDP Control Plane region ("us-west-1", "eu-
1" or "ap-1") (default "us-west-1")
      --skip-credentials-file        skip CDP credentials file discovery
      --tls-ca-certs string          additional PEM-encoded CA certificates
      --tls-insecure                 skip verification of API server TLS certifi
cate
      --user string                  CDP user to authenticate as
      --vcluster-endpoint string     CDE virtual cluster endpoint
  -v, --verbose                      verbose logging
      --version                      version for cde

Use "cde [command] --help" for more information about a command.

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

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

```
[mmehra@ Downloads % ./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
      --cdp-endpoint string          CDP API endpoint (default depends on CDP Control Plane region)
      --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
      --region string                CDP Control Plane region ("us-west-1", "eu-1" or "ap-1") (default "us-west-1")
      --skip-credentials-file        skip CDP credentials file discovery
      --tls-ca-certs string          additional PEM-encoded CA certificates
      --tls-insecure                 skip verification of API server TLS certificate
      --user string                  CDP user to authenticate as
      --vcluster-endpoint string     CDE virtual cluster endpoint
  -v, --verbose                      verbose logging
      --version                      version for cde

Use "cde [command] --help" for more information about a command.
mmehra@ Downloads %
```

- 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
```

```
mmehra@MacBook-Pro workshop % touch config.yaml
mmehra@MacBook-Pro workshop % ls
cde             config.yaml
mmehra@MacBook-Pro workshop %
```

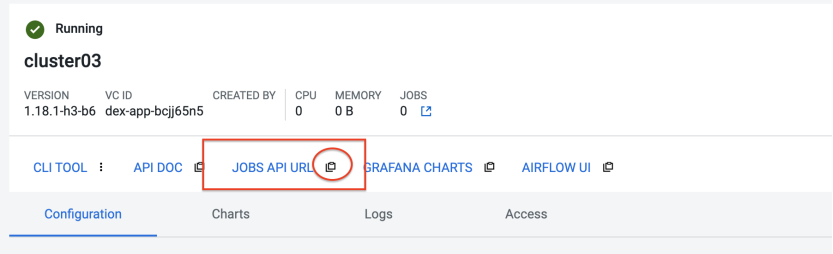
```
vi config.yaml
```

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

Here, **user** is the username you have been mapped in the excel sheet.

vcluster-endpoint can be obtained from the Virtual Cluster that is assigned to you. Go to the Virtual Cluster “Cluster Details”

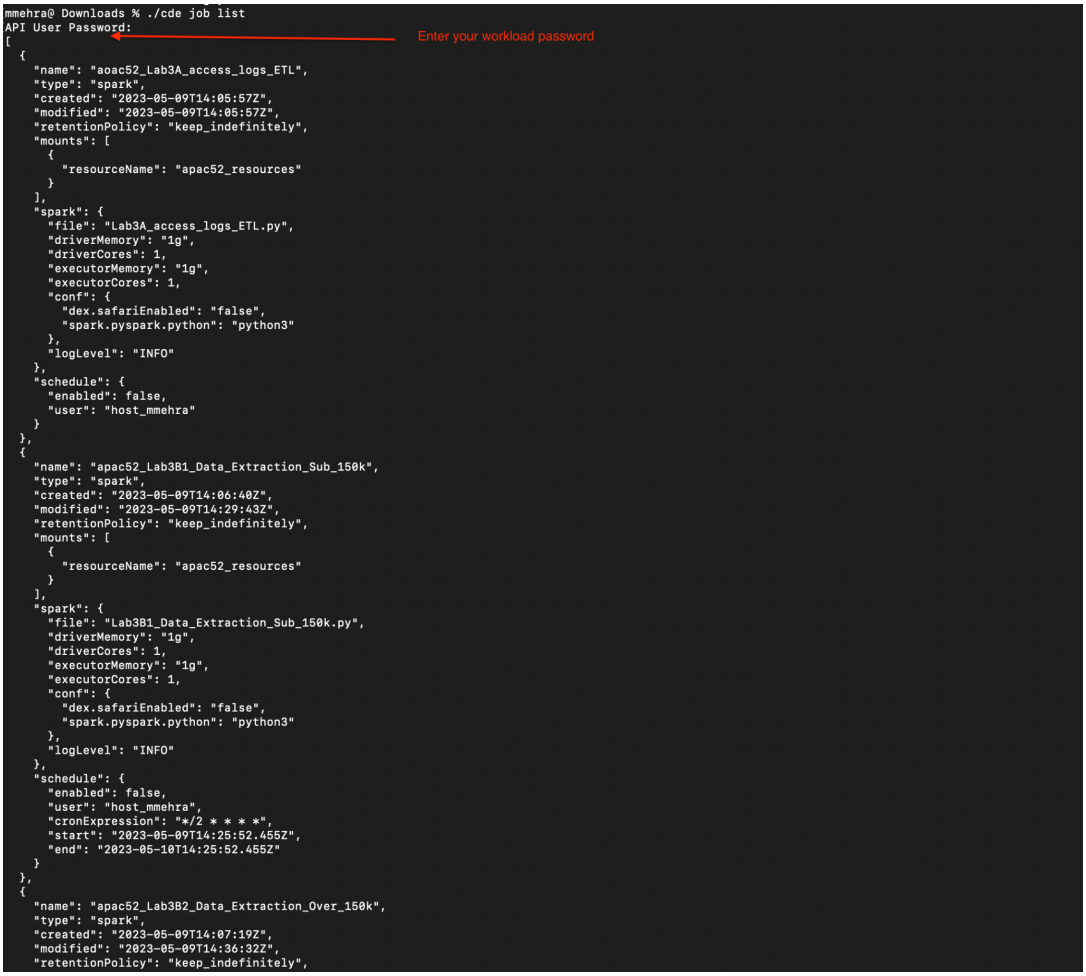
Click on the copy icon next to JOBS API URL to copy the **vcluster-endpoint**



- Save config.yaml
- 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
```

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



- 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.

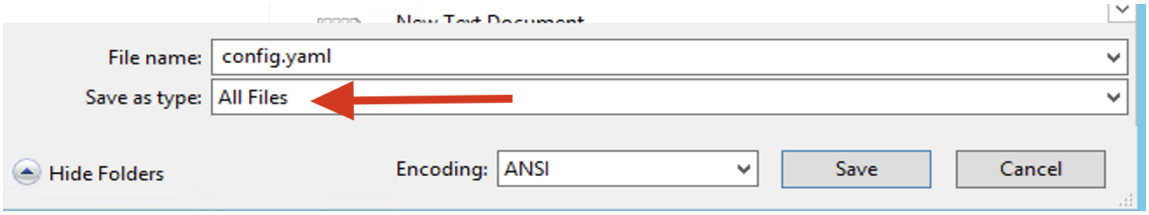
```
cd C:\Users\<path-to-cde.exe folder>
```

- 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. For the **vcluster-endpoint** get in touch with the instructor.

```
[Can be obtained from your virtual cluster]
```

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

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

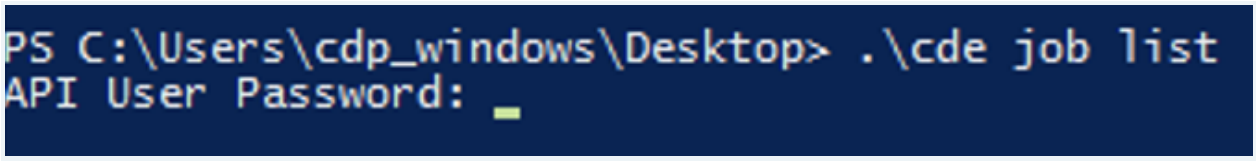
- Run the below command for validation. You should see the path-to-config.yaml as the output.

```
ls env:CDE_CONFIG
```

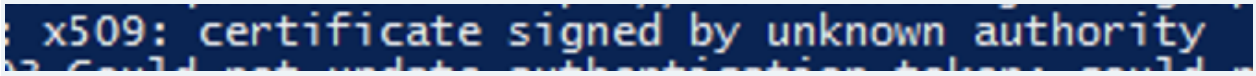
```
PS C:\Users\cdp_windows\Desktop> start .\cde.exe
PS C:\Users\cdp_windows\Desktop> $env:CDE_CONFIG =  "C:\Users\cdp_windows\Desktop\config.yaml"
PS C:\Users\cdp_windows\Desktop> ls env:CDE_CONFIG

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
```



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



- 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.

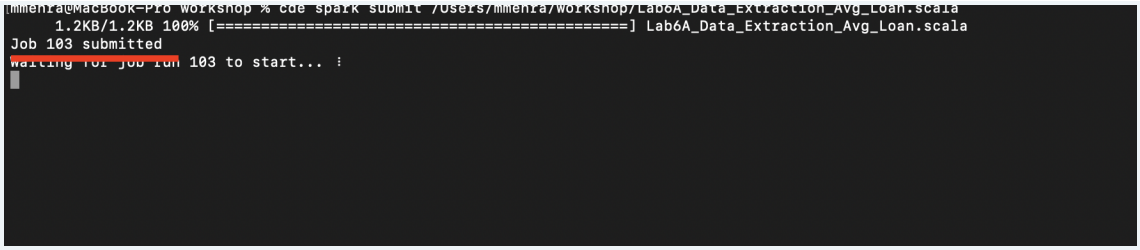
https://docs.cloudera.com/data-engineering/cloud/cli-access/topics/cde-cli-manage-jobs.html

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
```



- 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

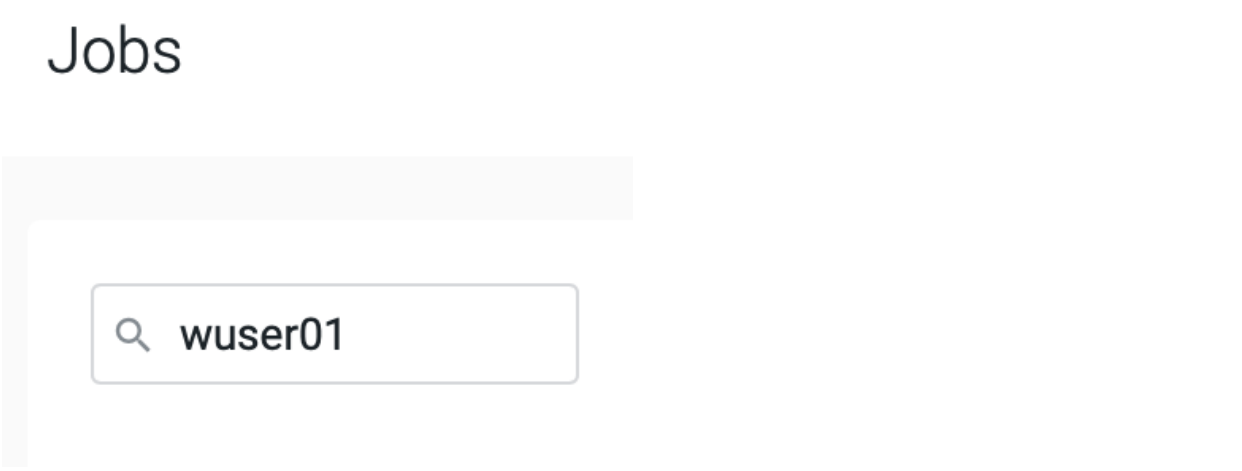
- You can observe the logs and SparkUI for this Job Run.
- Please note that you are not creating this as a job in CDE. It will be an ad-hoc run without the need of registering it as a job.

Lab 7 - Data Lineage and Auto-Scaling

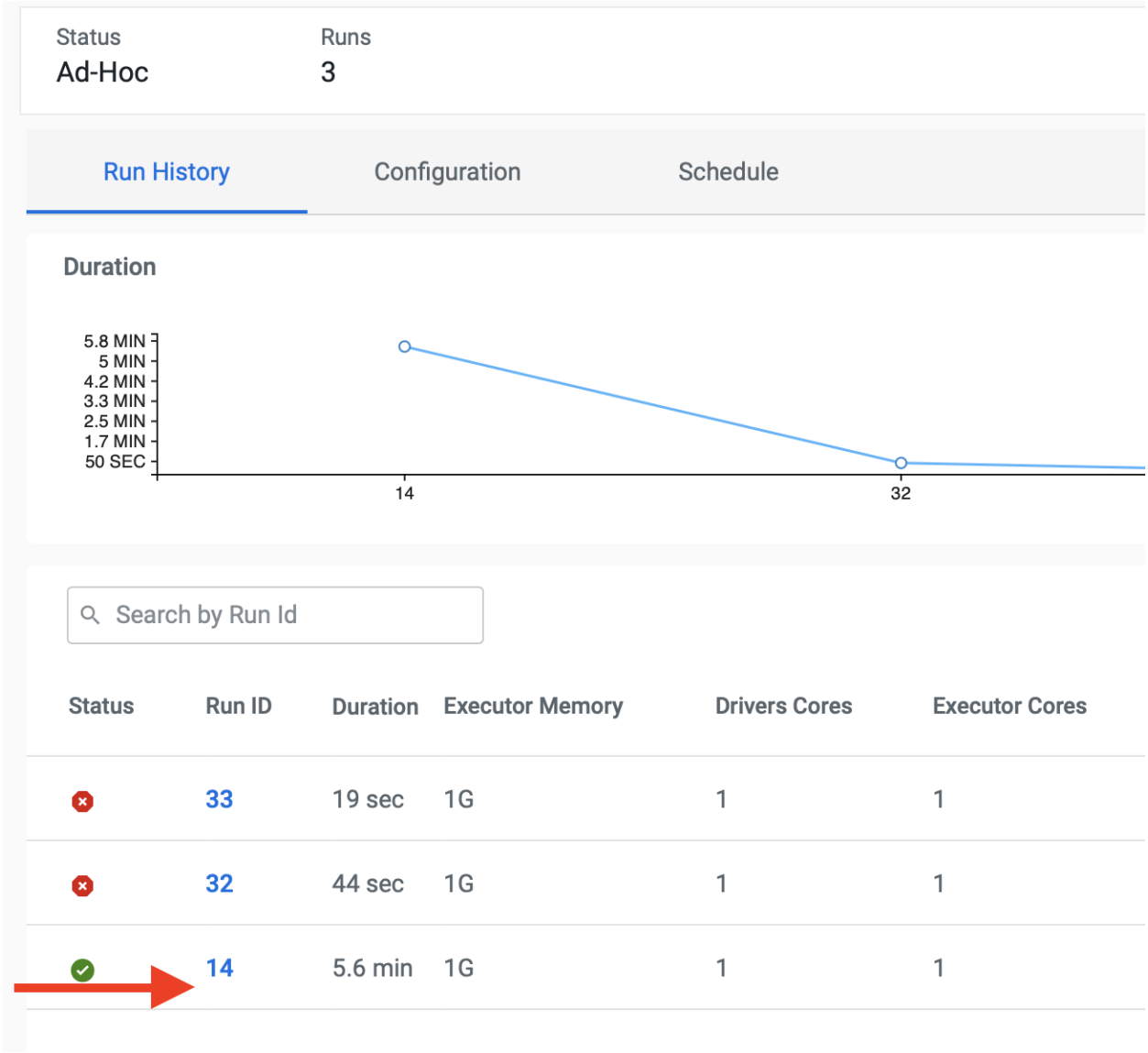
In this lab, you will go through the data lineage of the two use cases that we worked on. Additionally, you will also see the auto-scaling capabilities of CDE service with the rising demand for compute resources.

Data Lineage using Atlas

- In the CDE UI, click on the Jobs tab. Go to the job `<username>_Lab3B3_Create_Reports` that you have created in the Lab2.
- To get the jobs, please filter the jobs with your username.

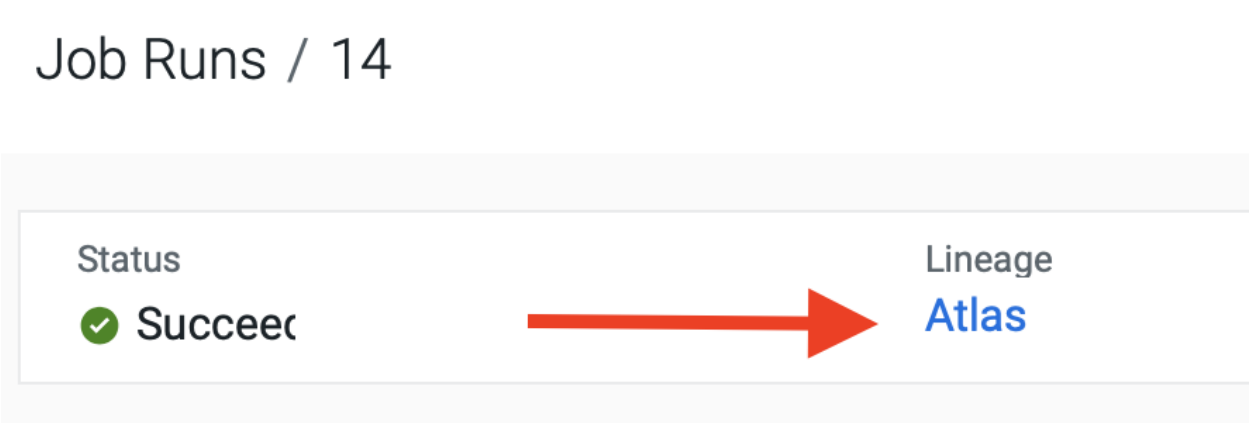


- In **Run History** tab, click on the successful Run ID i.e., the one with the green tick mark.

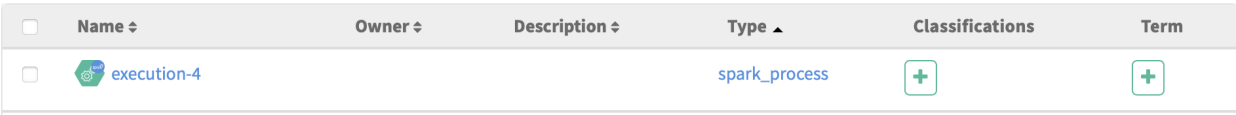


(Note the ID will be different for you from the one you see in the screenshot)

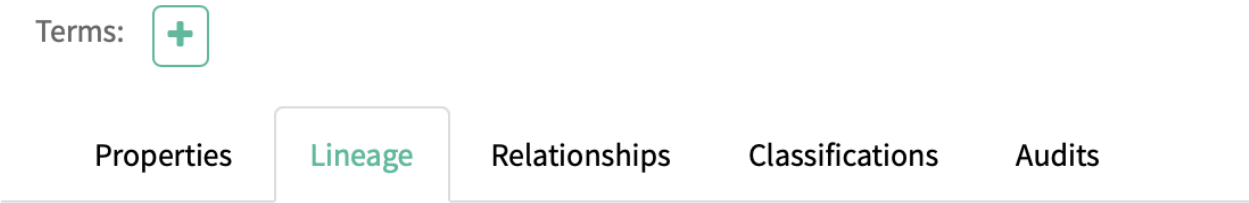
- Click on **Atlas** under Lineage.

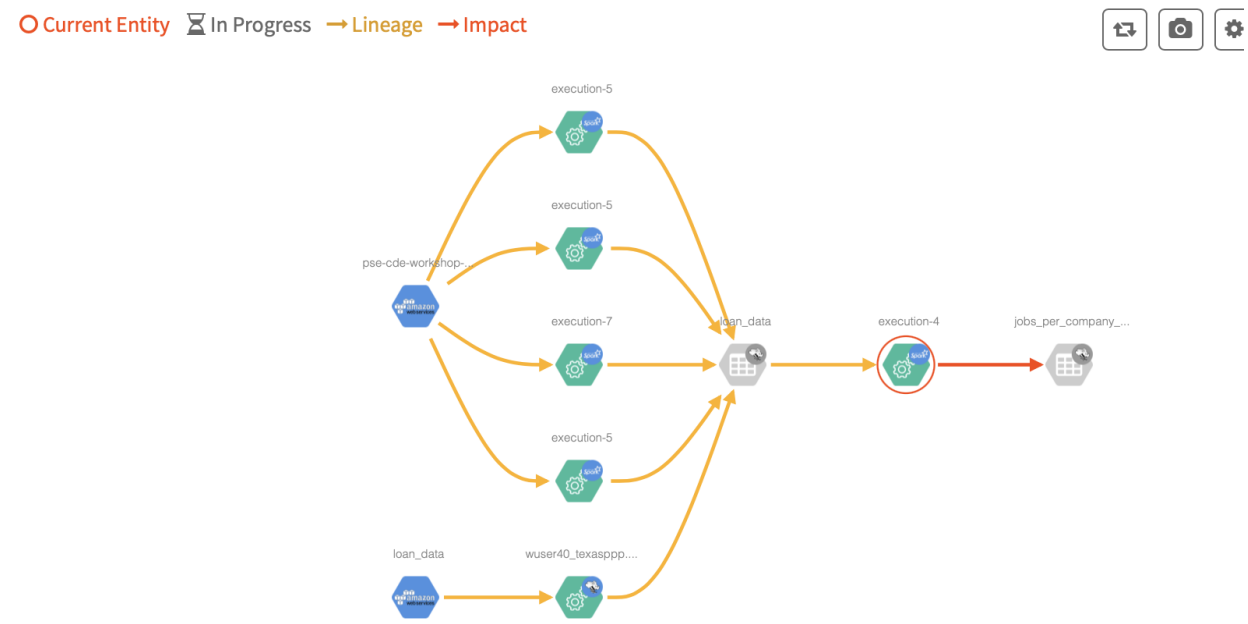


- Click on the execution that you see in the list.



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

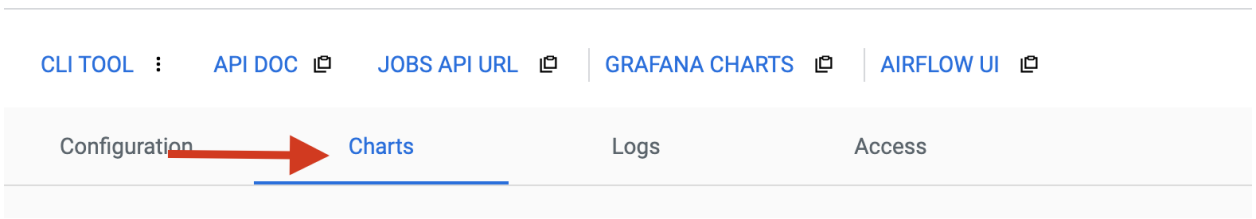




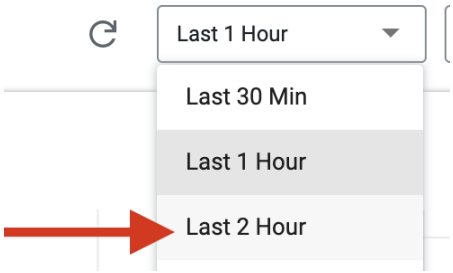
- Click on each entity to understand how the data is flowing from source to consumption.

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.



- 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.

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

THANK YOU VERY MUCH FOR YOUR PARTICIPATION