CDE Workshop Student Guide

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.

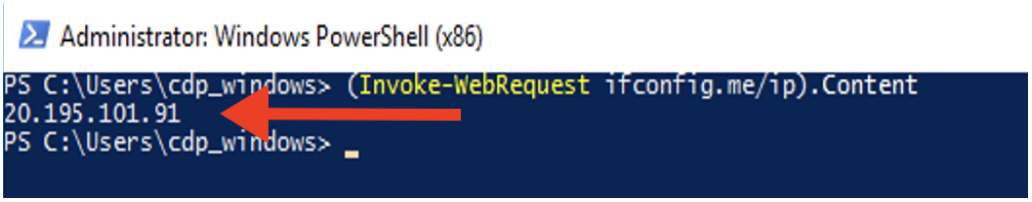
Using a Browser

- 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
```

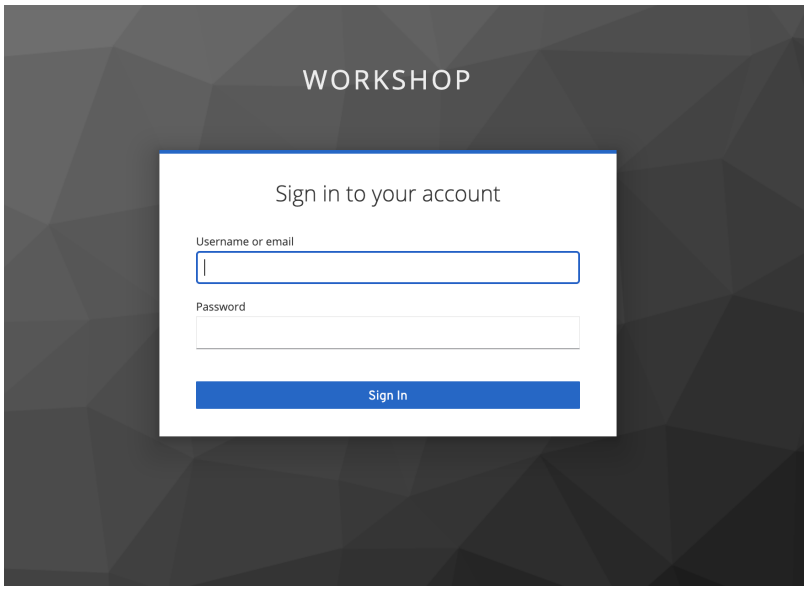


- **MAC:** Open Terminal and run the below command. You will get the public IP address as the output.

```
curl ifconfig.me
```

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

  http://13.234.35.193/auth/realms/workshop/protocol/saml/clients/samlclient

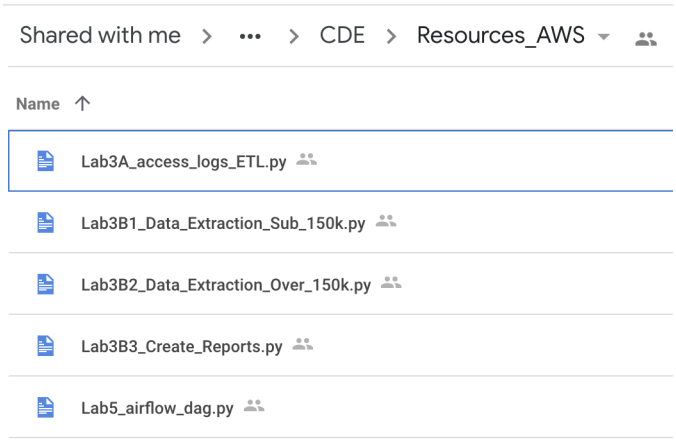


- You should land on the CDP Console as shown below.

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

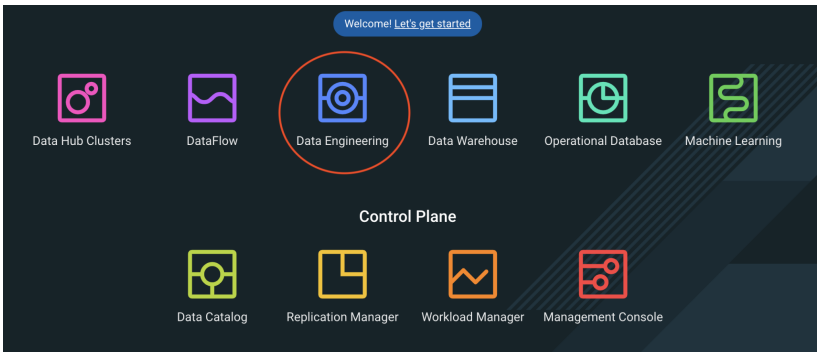
Cloudera Data Engineering (CDE) is a serverless service for Cloudera Data Platform that allows you to submit jobs to auto-scaling virtual clusters.

The CDE service involves several components:

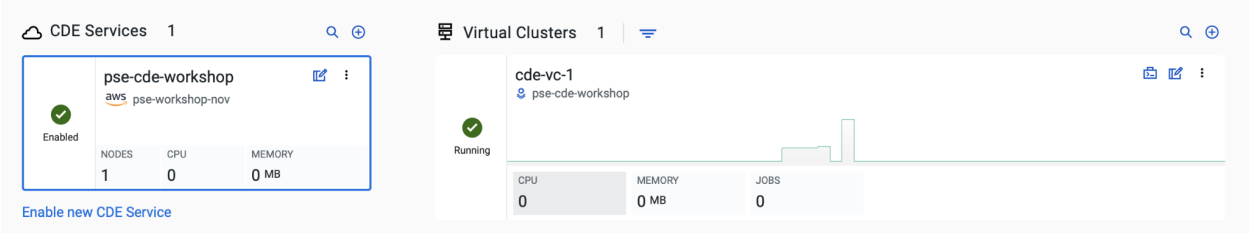
- **Environment**
  - A logical subset of your cloud provider account including a specific virtual network.
- **CDE Data Service**
  - The long-running Kubernetes cluster and services that manage the virtual clusters. The CDE service must be enabled in an environment before you can create any virtual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

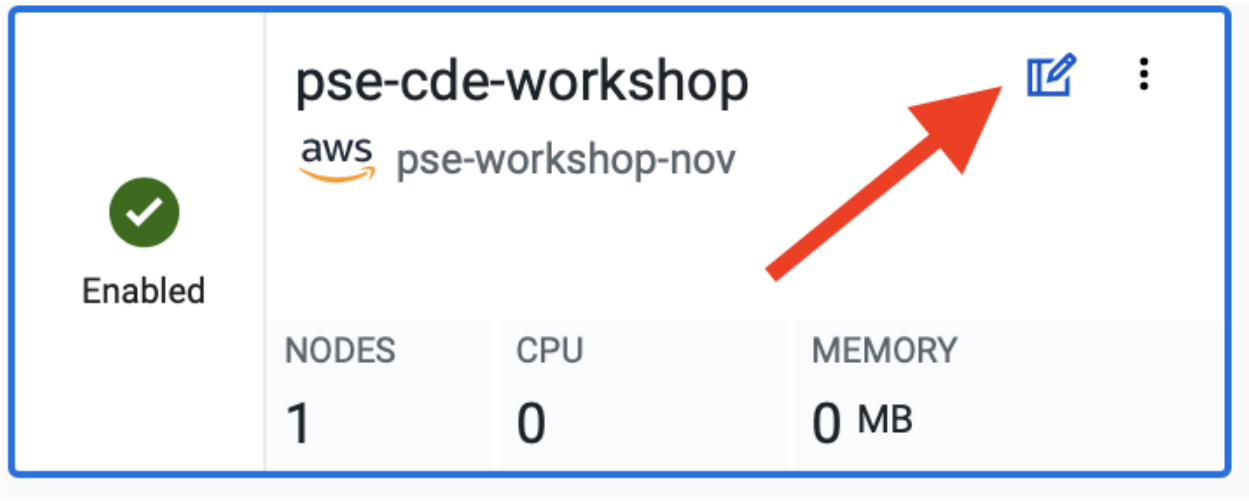
- Go to the CDP console and click on Data Engineering.

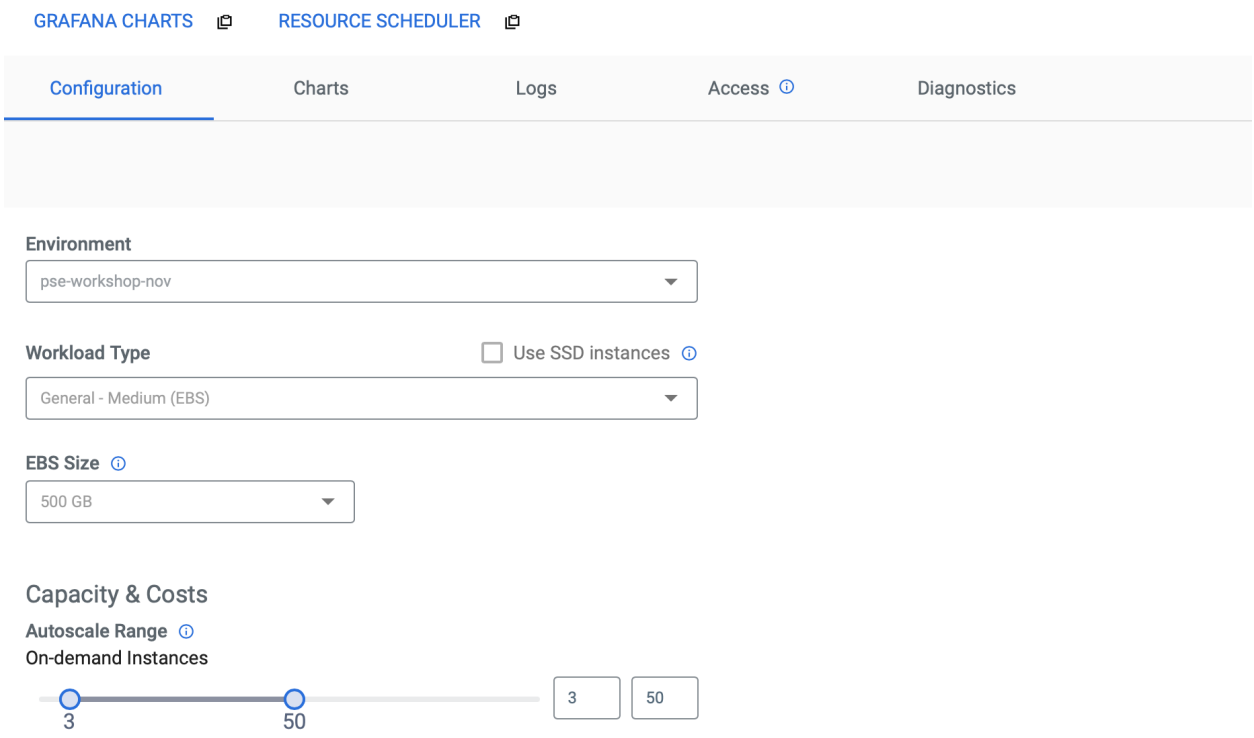


- You will see a CDE service and an associated virtual cluster.

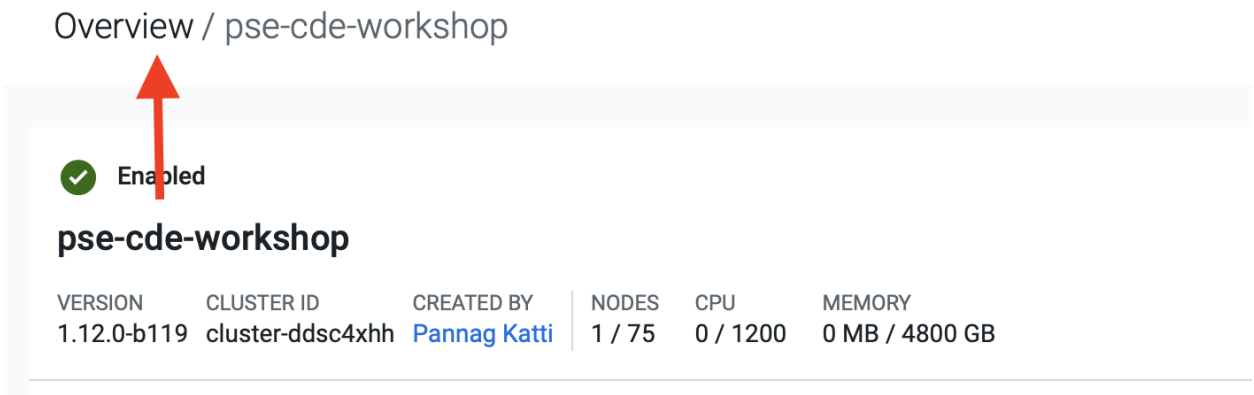


- On the **CDE service pse-cde-workshop,** click on the pencil icon and observe the configuration and other details related to the service.

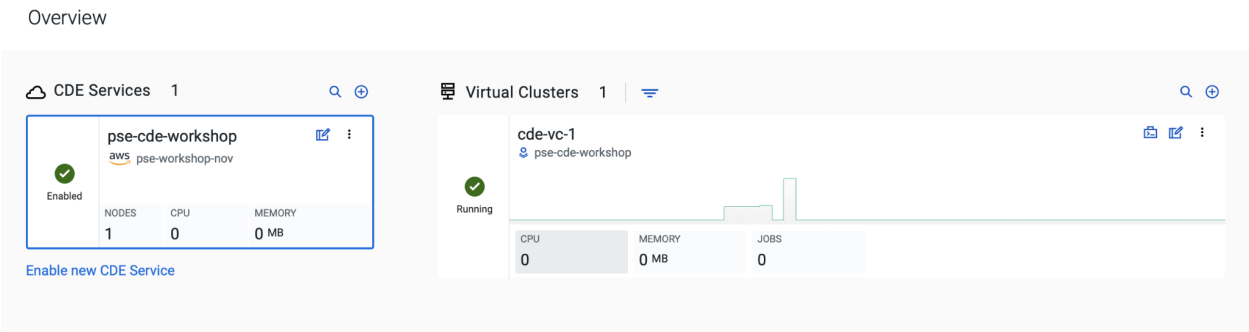




- Click on each tab and go through all the details related to the CDE service.
- Once done, click on the **Overview** to go back to the CDE home page.

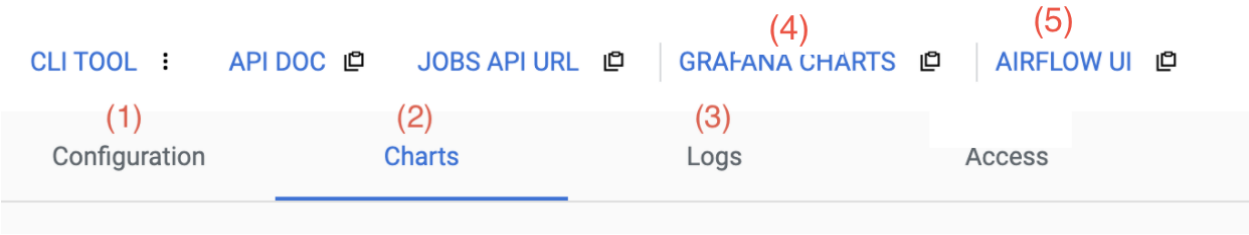


- You will land back on this page.

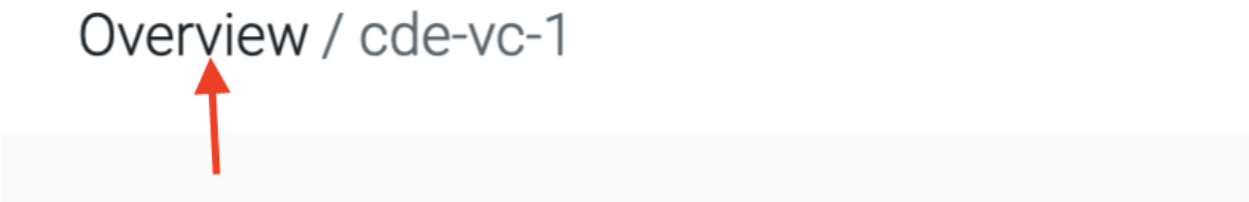


- On the right side, you will see the virtual cluster attached to this CDE service **pse-cde-workshop**. Click on the pencil icon on the **virtual cluster(VC**) **cde-vc-1** and observe the configuration details.

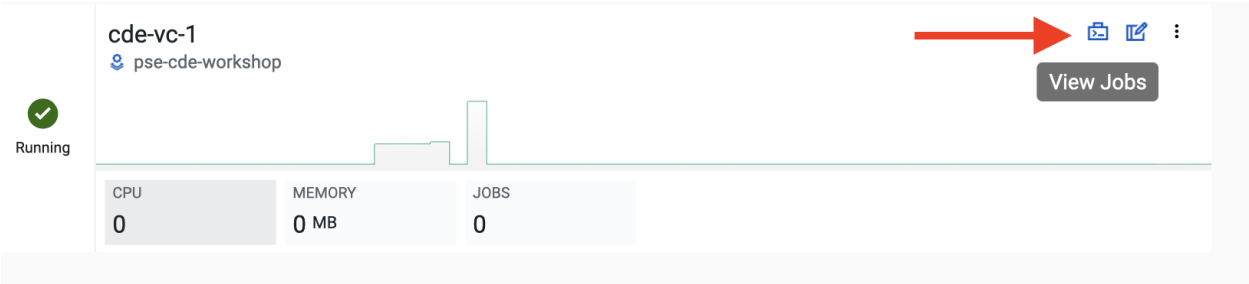
- Notice the compute limits that are set for this virtual cluster. When the demand increases, CDE will autoscale this cluster as per these values. At the end of this lab, we will see this with the help of dashboards.
- Navigate to all the other tabs numbered in the screenshot below and observe the details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



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



- 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.

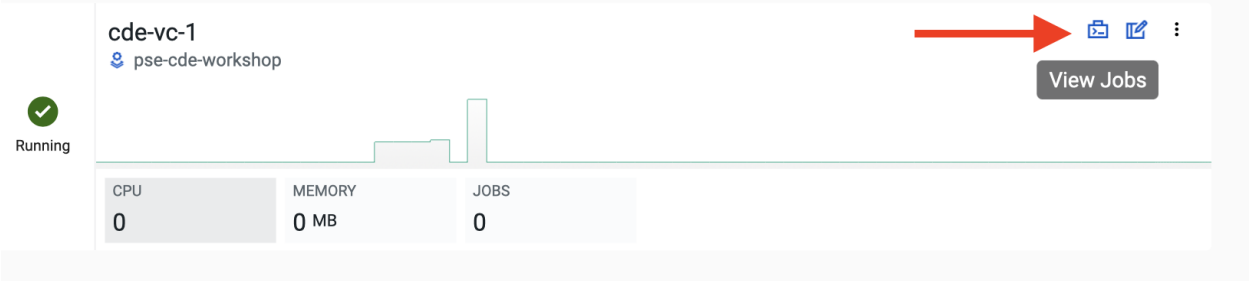
Lab 2 - Create and trigger ad-hoc Spark jobs

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.

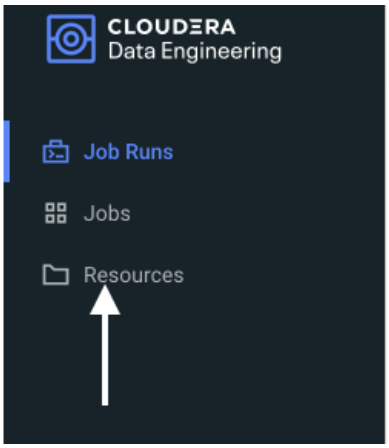
1. Log Data Cleansing using Spark
2. Analyze the Paycheck Protection Program Data
    a. Report 1: Breakdown of all cities in Texas that retained jobs
    b. Report 2: Breakdown of company type that retained jobs
3. PySpark job to enrich your data using an existing data warehouse

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.



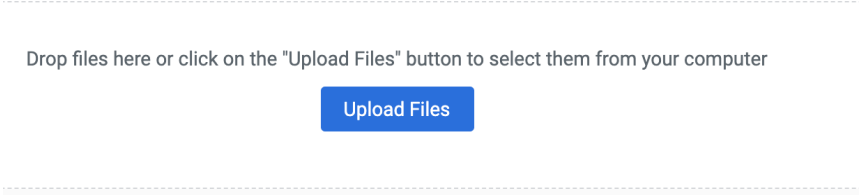
- In the left pane, click on the **Resources** tab.



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

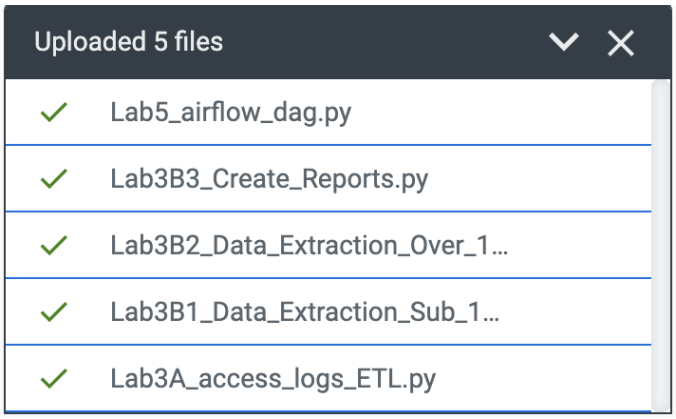
- Give a unique name(username-resources) and create the resource. This acts as your repository for storing all the scripts and dependencies.

- Once it is created, you will get an option to upload the files as shown below.

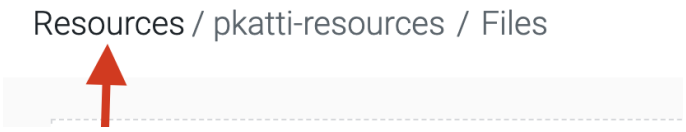


- Click on **Upload Files** and select all the scripts downloaded from the prerequisites step. (**Please upload only .py and .scala files**)

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

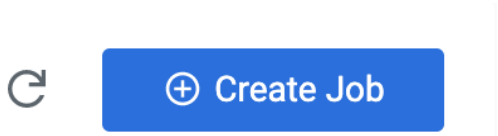


- Click on **Resources** at the top, select your resource and validate if all the seven *.py* files and one *.scala* are present in your resource. We are now ready to create jobs using these resources.



Job Creation

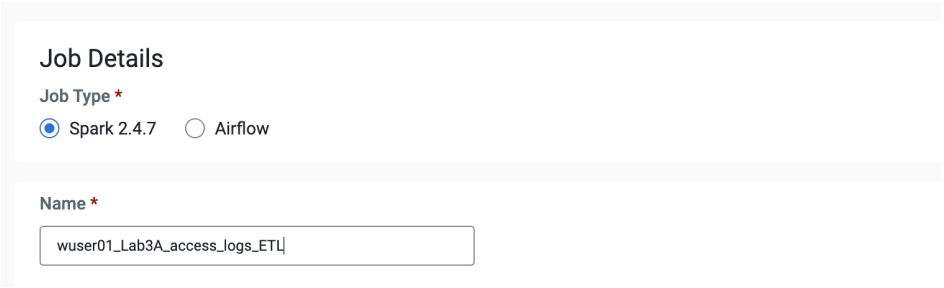
- We will now create the first job with the script ***Lab3A_access_logs_ETL.py***.
- In the left pane, click on **Jobs.**
- You will get the **Jobs** page to the right. Click on **Create Job**.



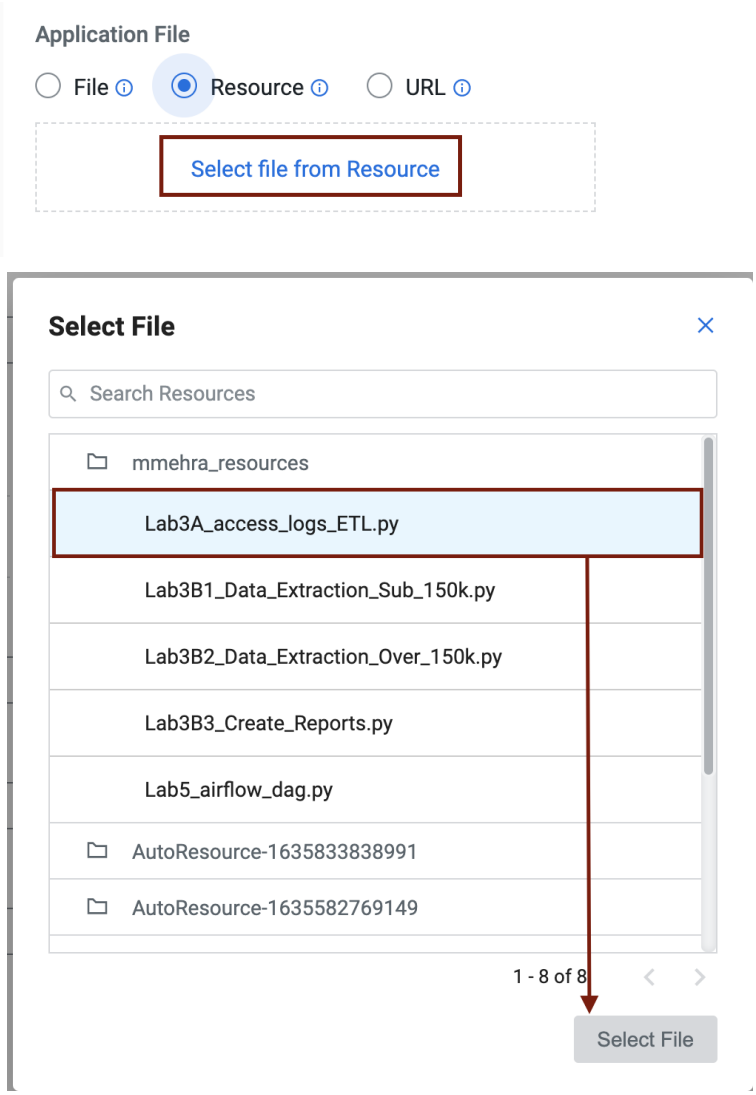
- Select job type as **Spark**.
- Please give the job names as mentioned below.

<username>_<script_name_without_py_extension>

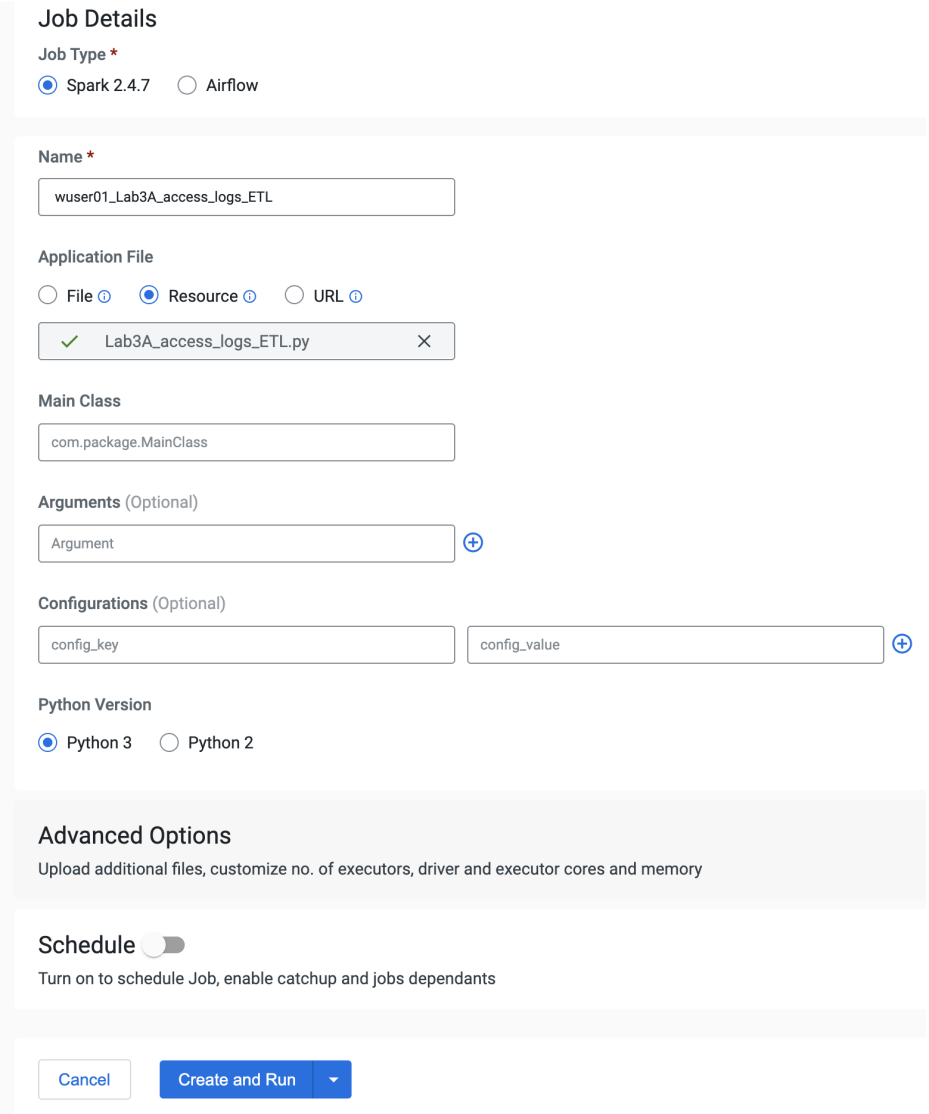
Eg:- For wuser1, job1 name would be **wuser1_Lab3A_access_logs_ETL**



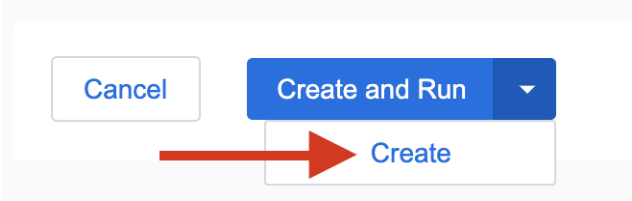
- **As this is a shared environment, please name the jobs with your username so that it helps in differentiating yours from others' jobs.**
- In **Application File**, click on **Resource** and select the file ***Lab3A_access_logs_ETL.py*** from your resource(<username>-resources).



- 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**.

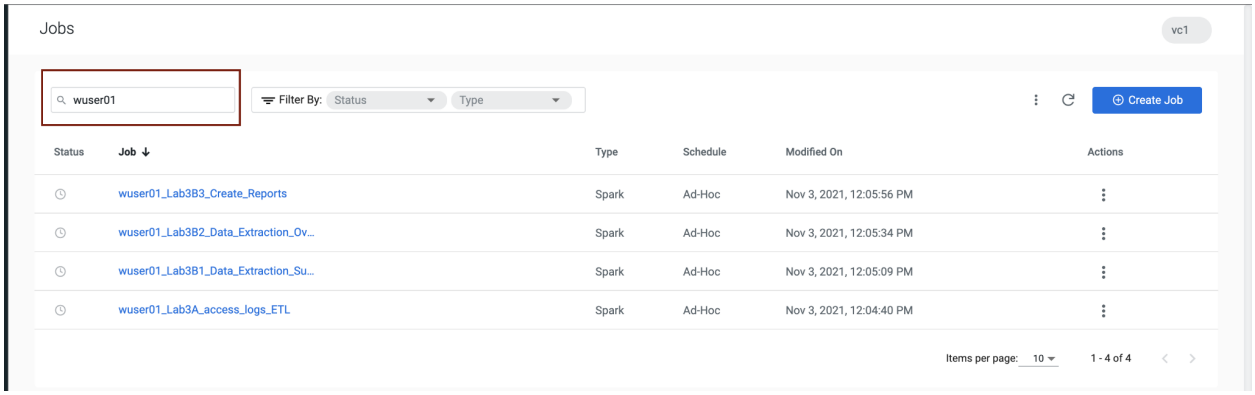


- Similarly, create four other jobs with the same naming conventions. Please refer to the table below to confirm you are creating exactly the same.

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_Reports	Lab3B3_Create_Reports.py

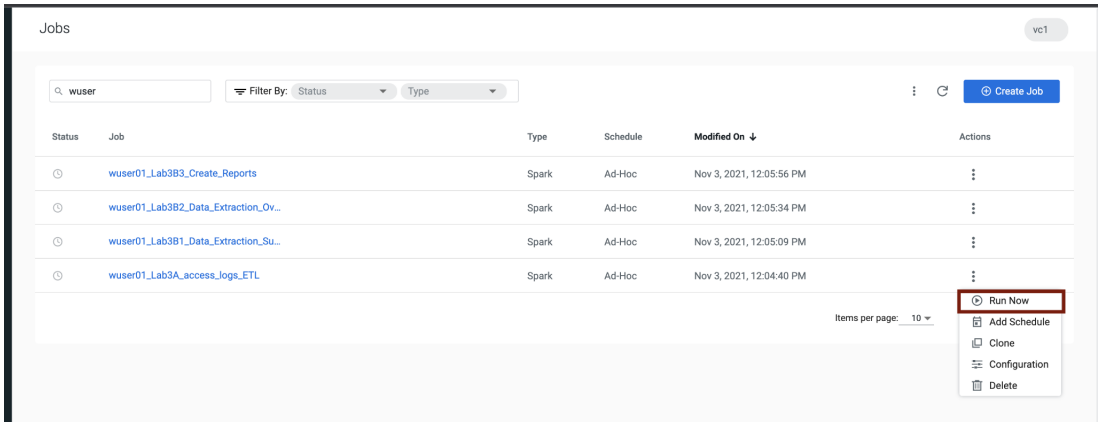
- Create these jobs as **ad-hoc** jobs i.e., without any schedule.
- Once done, click on the **Jobs** tab and enter your username in the search bar and press **ENTER.** You should see four jobs as shown below with your username.



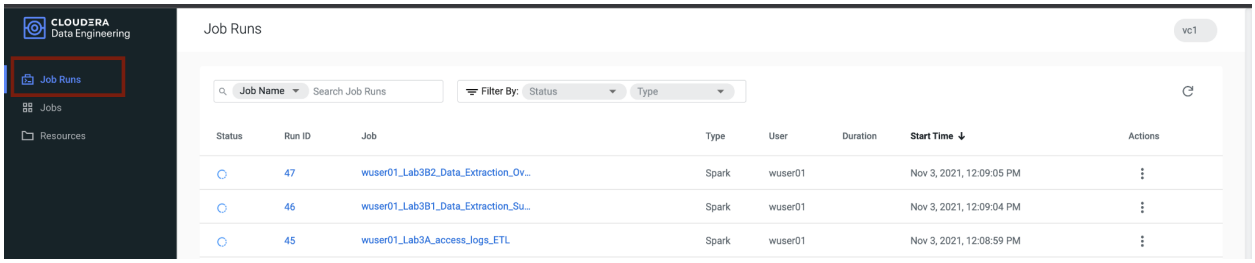
- Observe the type of the job is set to Spark and for schedule, it is Ad-hoc.

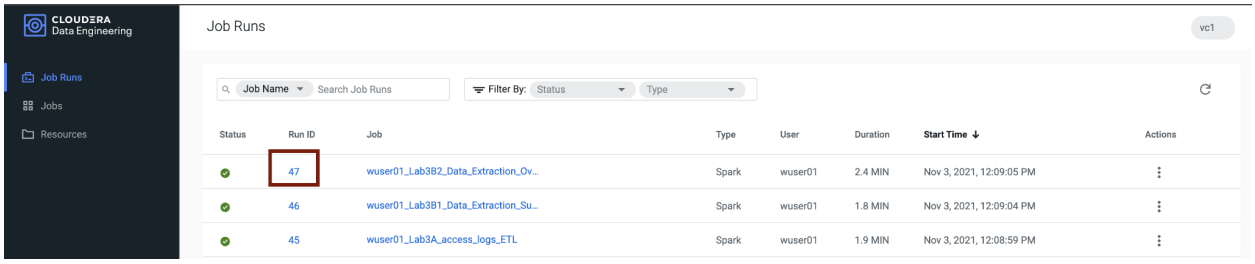
Triggering the jobs

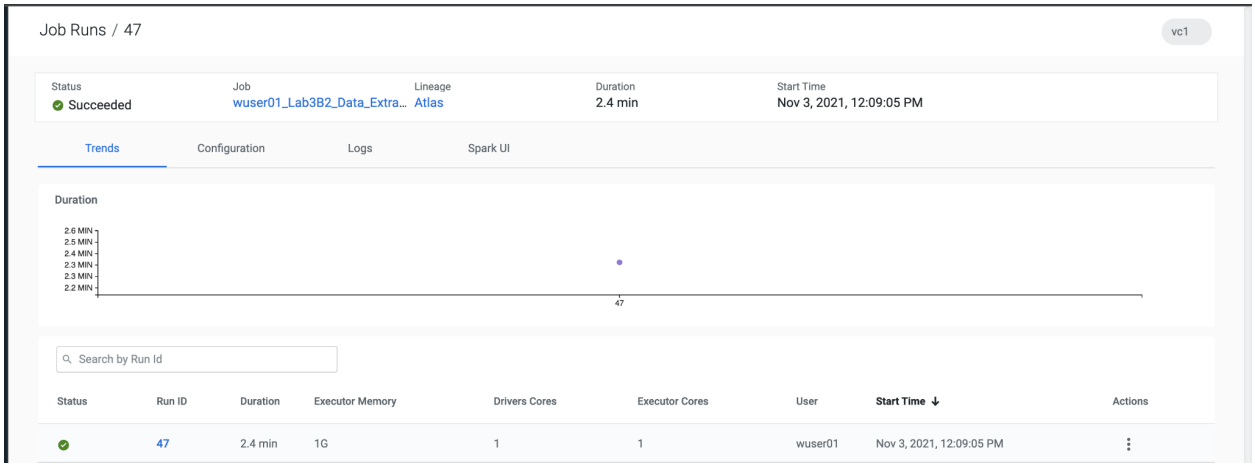
- You can trigger each job one by one and observe the different tabs under job run.
- To trigger the job, go to the **Jobs** tab, click on the 3-dotted icon, and click on **Run Now.**



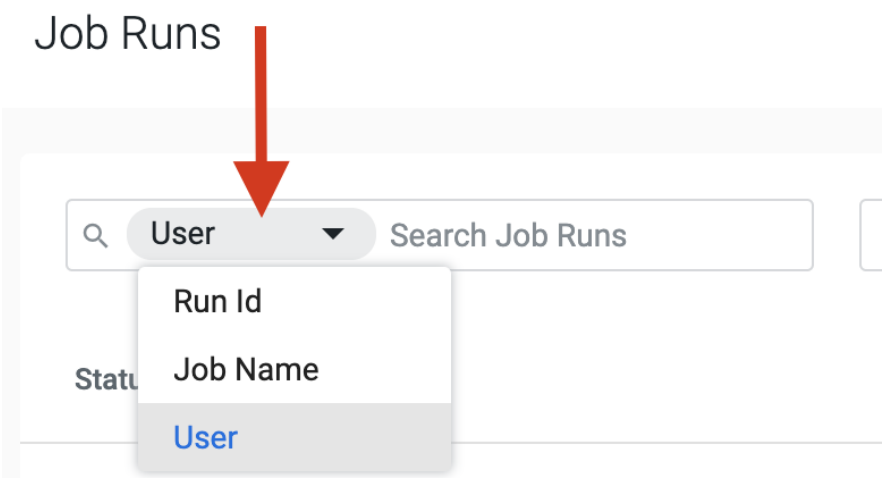
- To check the job logs, click on **Job Runs** and select the **ID** against the job that you have triggered.



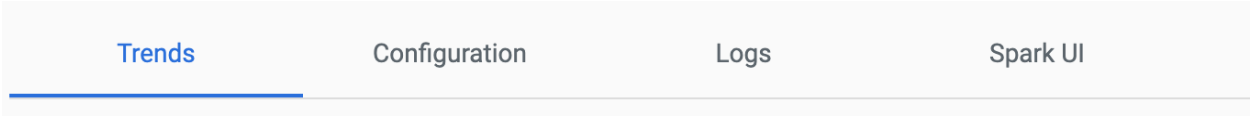




- For simplifying the job selection, you can choose the **User** filter and add your username and hit enter. You will see the list of jobs triggered by you.



- 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.

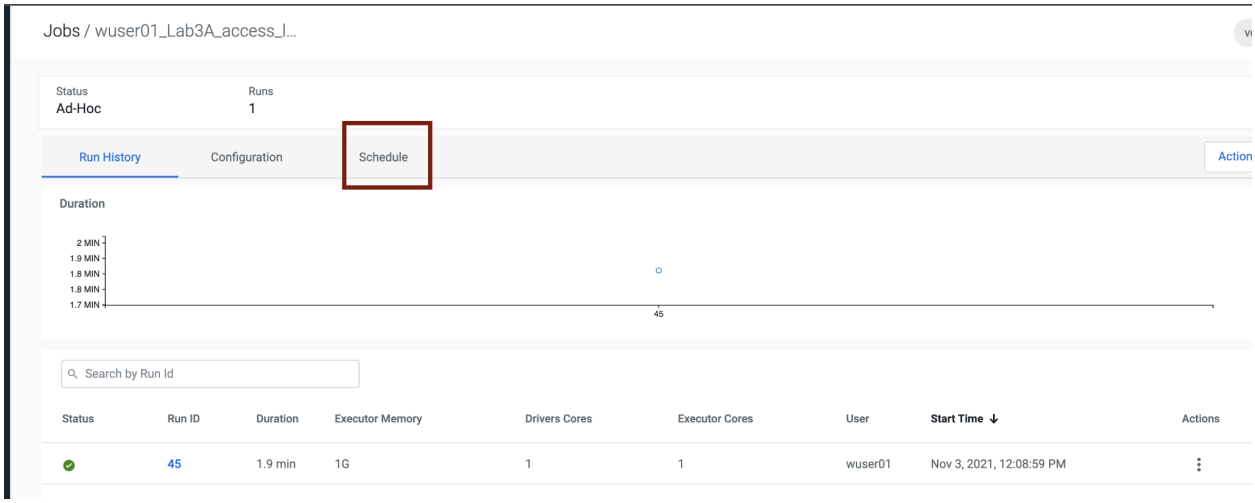


Lab 3 - Add schedule to the ad-hoc Spark jobs

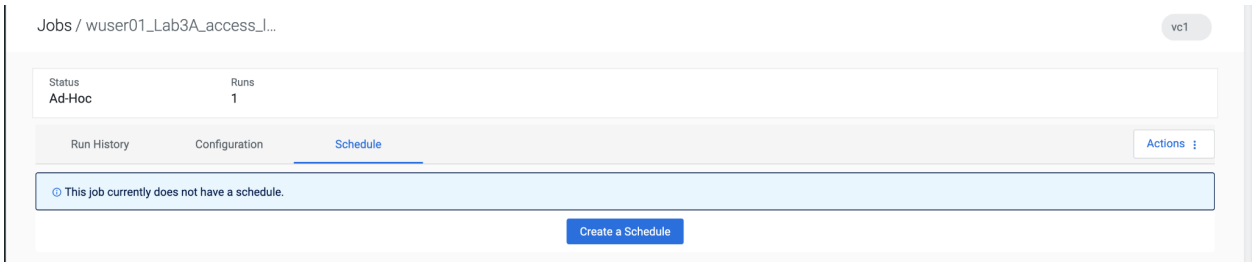
In this lab, we will add a schedule to a job created as part of the previous lab.

- 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**.

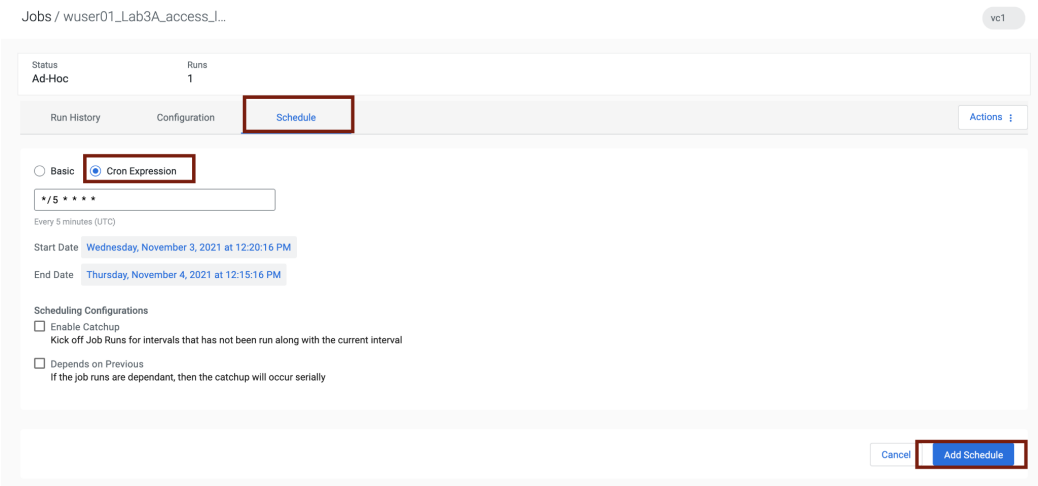


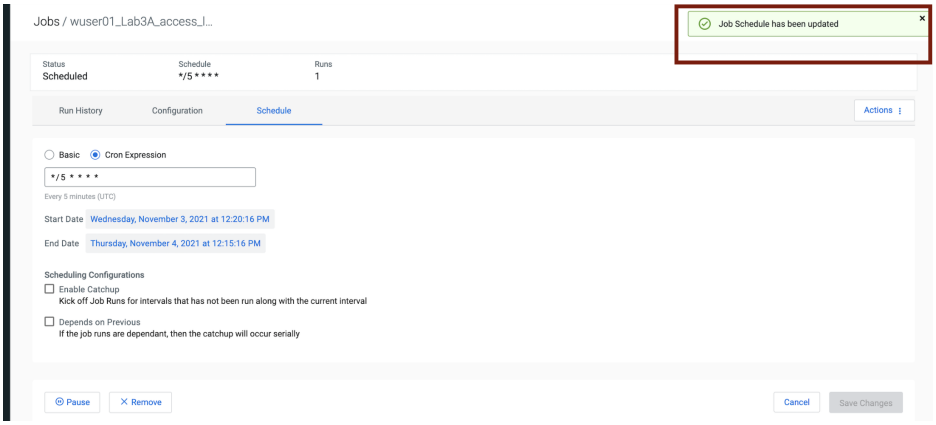
- You will land in the **Job Schedule** page. Click on **Create a Schedule**.



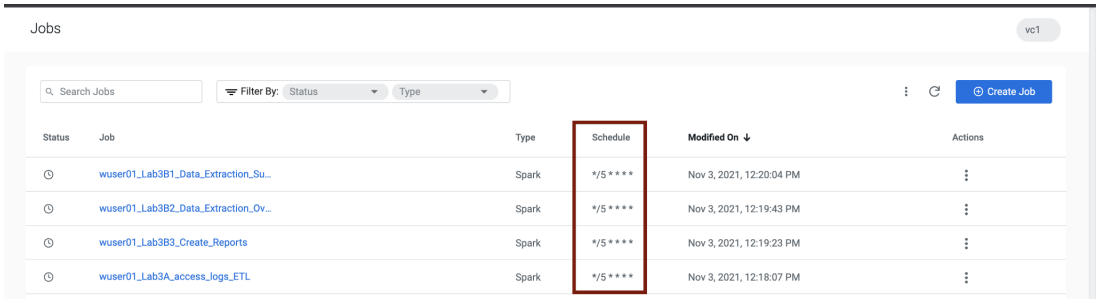
- Choose the **Cron Expression** option and enter the cron expression as given below.

  ***/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.

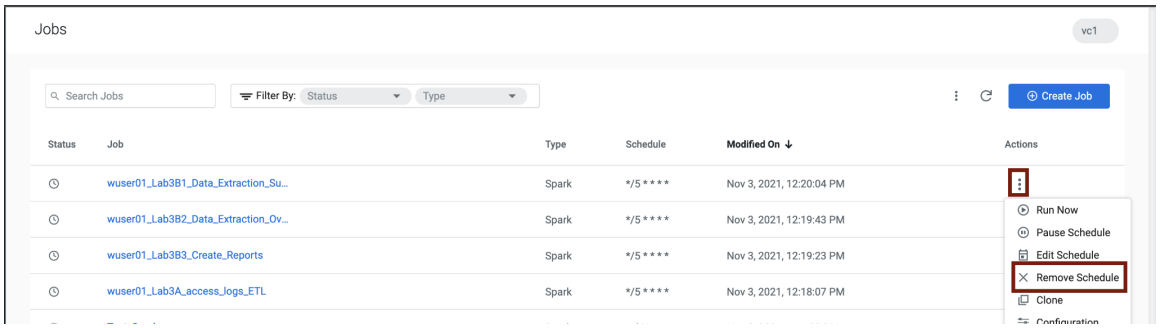


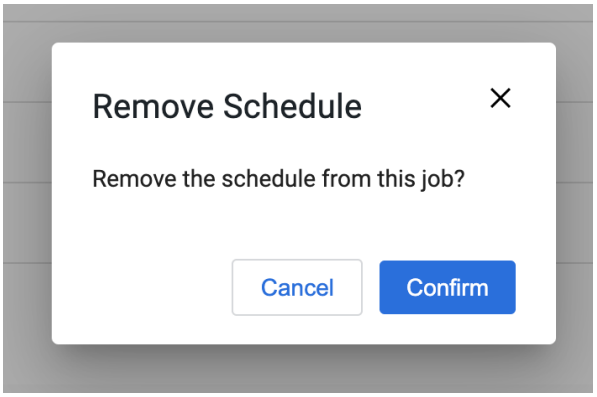
- You can modify the cron expression if you wish to. Refer to this to validate your cron expression.

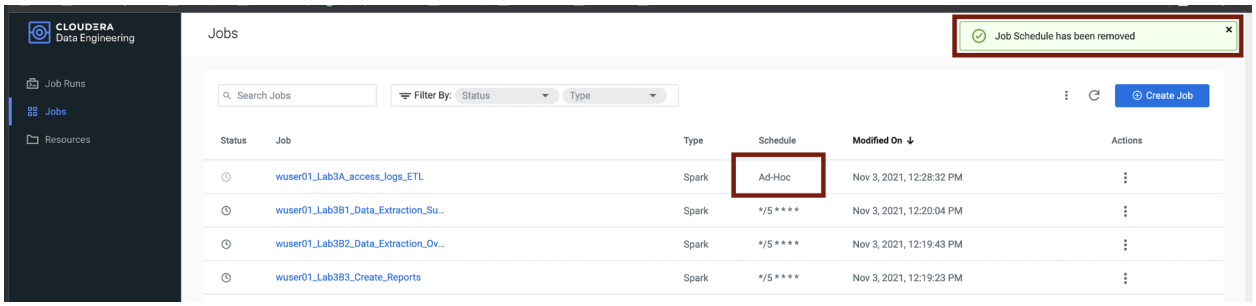
- Wait for the scheduler to trigger the jobs at the scheduled time and observe the results and logs for the same.

- **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**.







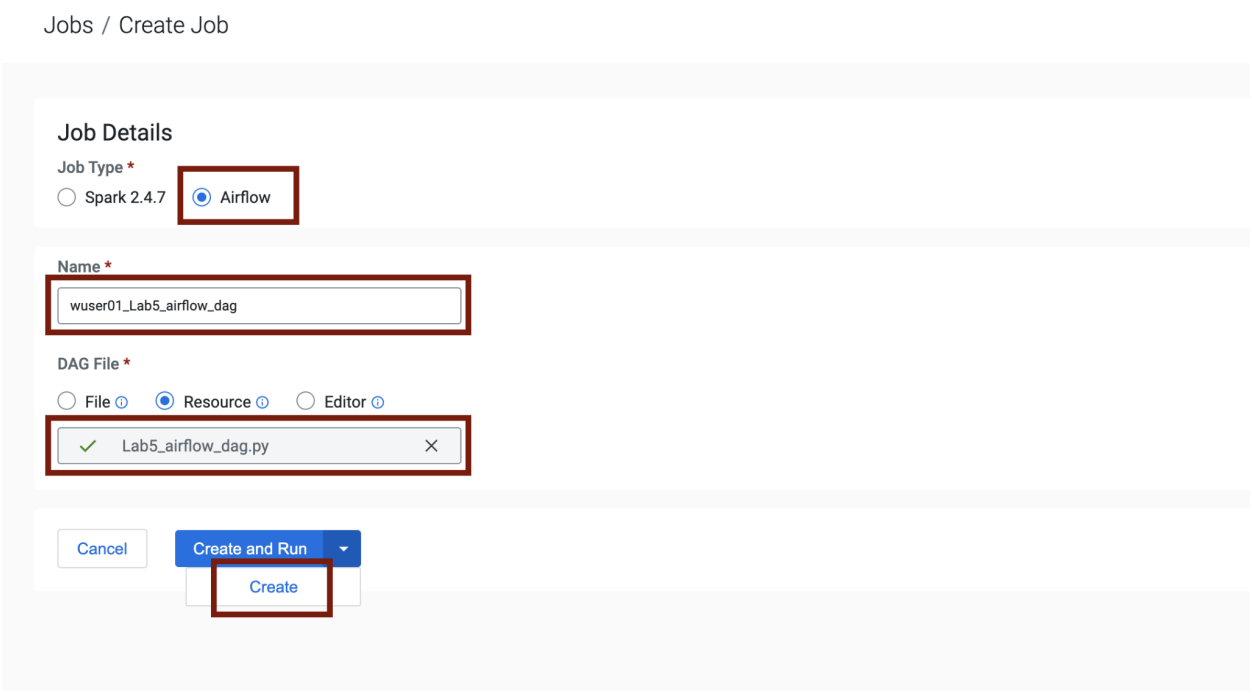
Lab 4 - Orchestrate a set of jobs using Airflow

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.

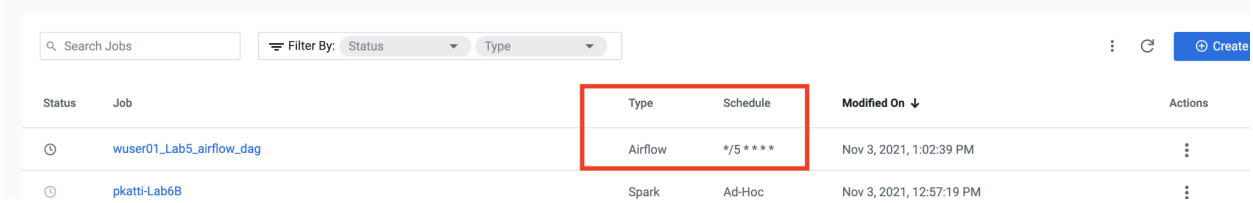
  JOB NAME : <username>_Lab5_airflow_dag
  Example : For user ***wuser01*** the job name will be, ***wuser01_Lab5_airflow_dag***

- Click on **Create.**



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

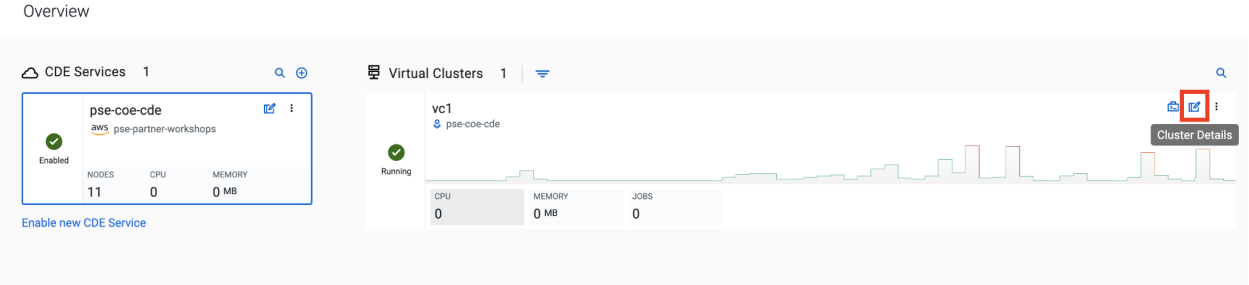
  Job



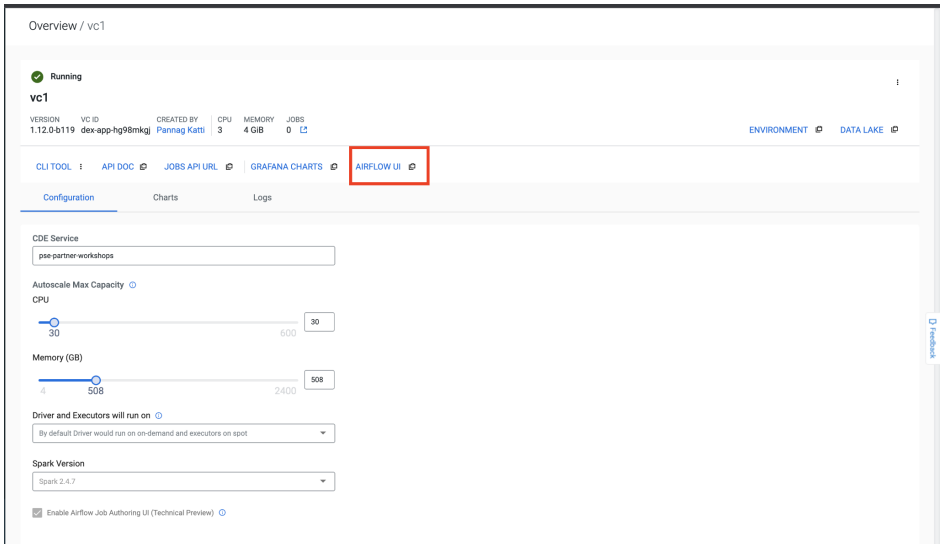
DAG File

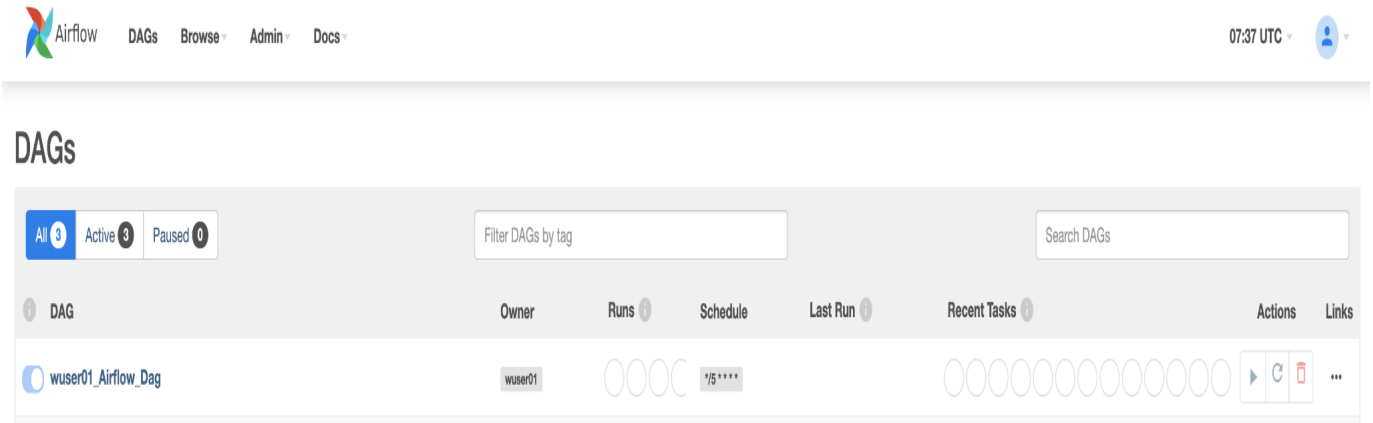
```
Lab5_airflow_dag.py
12 DAG_name = owner + "_Airflow_Dag"
13 job_name_1 = owner + "_Lab3B1_Data_Extraction_Sub_150k"
14 job_name_2 = owner + "_Lab3B2_Data_Extraction_Over_150k"
15 job_name_3 = owner + "_Lab3B3_Create_Reports"
16
17
18 default_args = {
19     'owner': owner,
20     'retry_delay': timedelta(seconds=5),
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
```

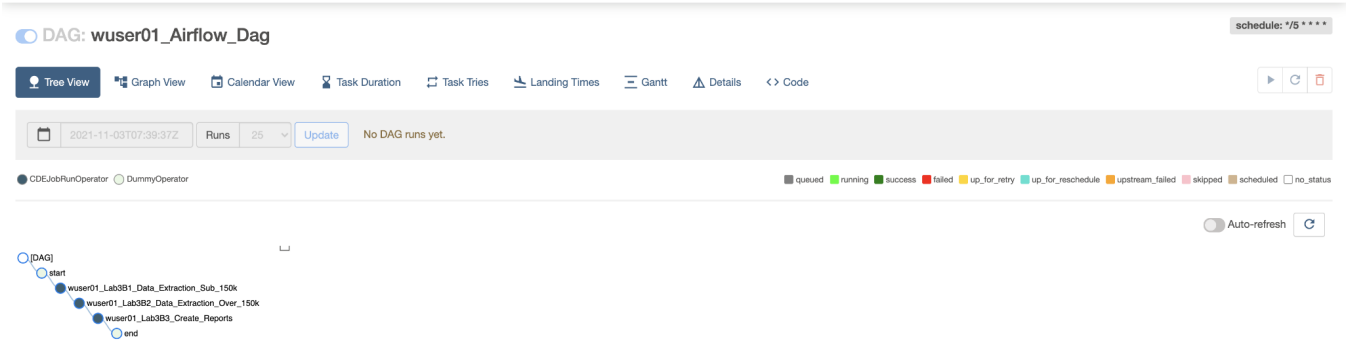
- Go to the Virtual Cluster <> and click on **Cluster Details**.



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

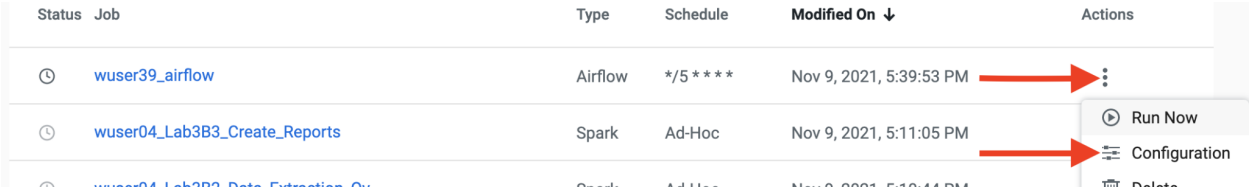




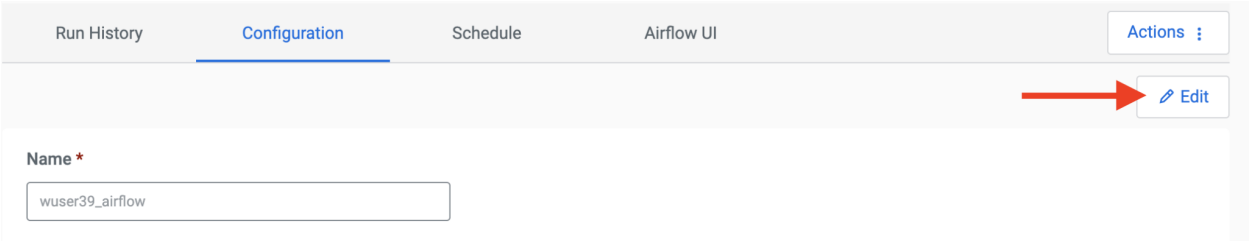


- Once the job has run successfully, we need to edit the job to pause the schedule.
- Click on the Jobs tab and locate the airflow job that you have just created.

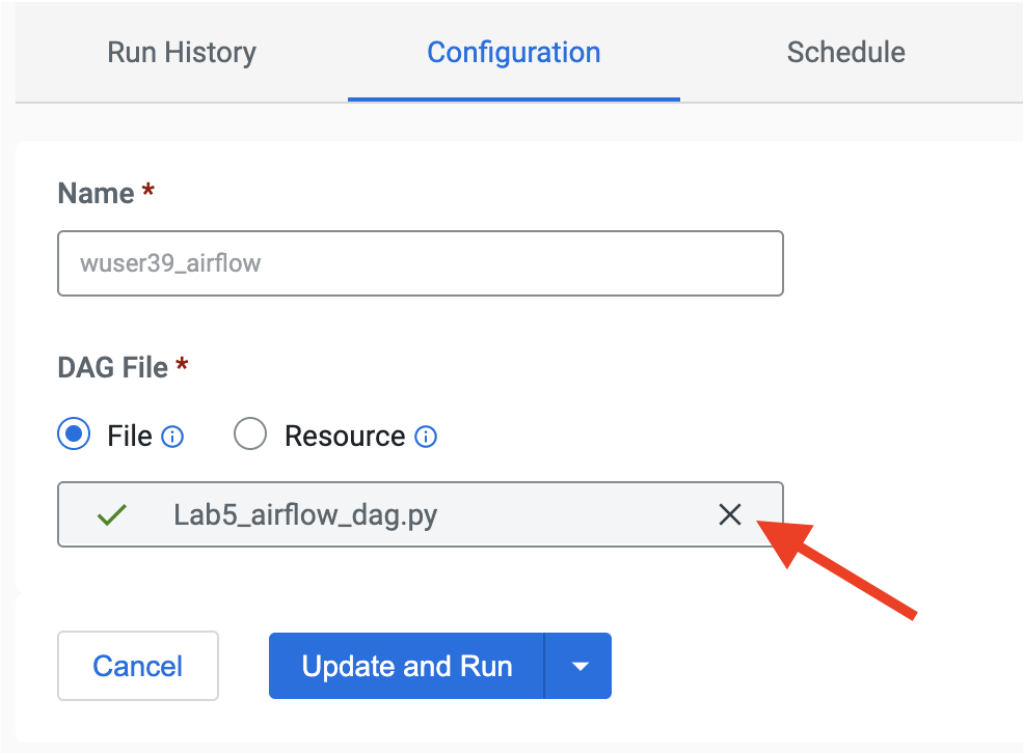
- Next to the job, click on the 3 dots and click on **Configuration**.



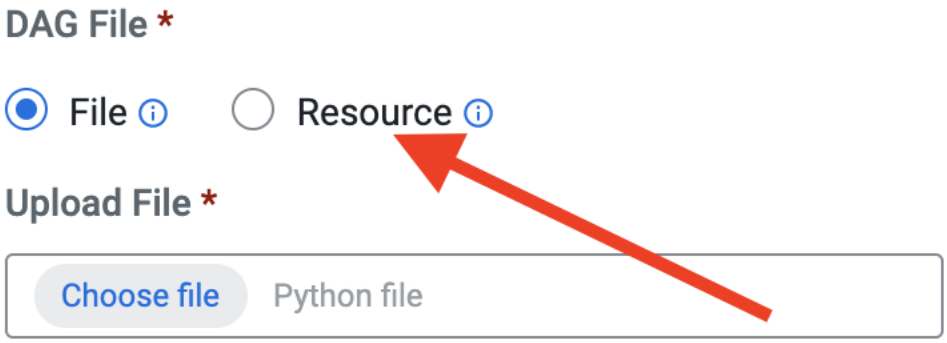
- Under Configuration, click on **Edit.**



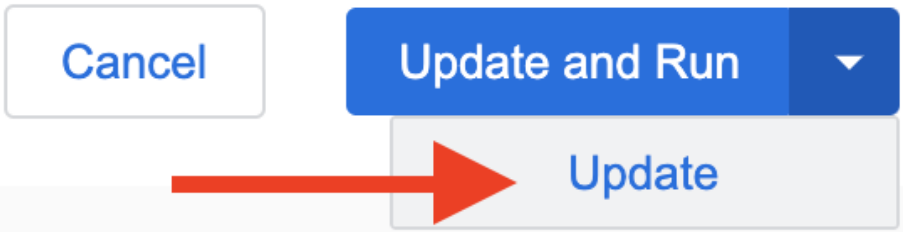
- In the **DAG File** section, click on the cross button to remove the dag file.



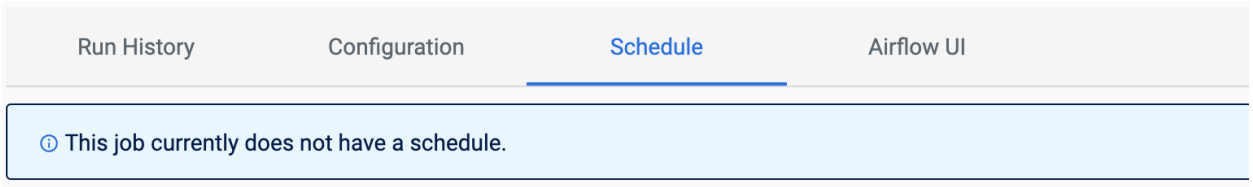
- Select **Resource** and upload the file **Lab5A_airflow_dag_STOP.py**.



- Click on **Update**.

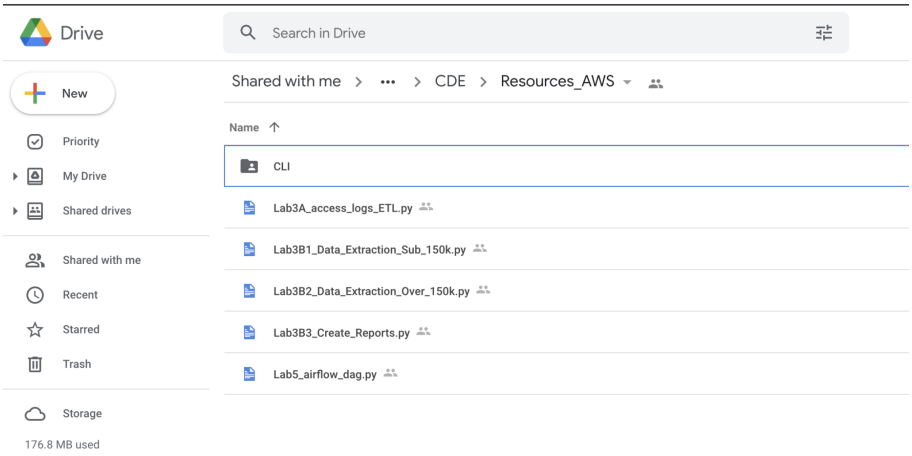


- Once done, go to the **Schedule** tab of the job and you should see the similar status as there is no schedule for the given job.



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.

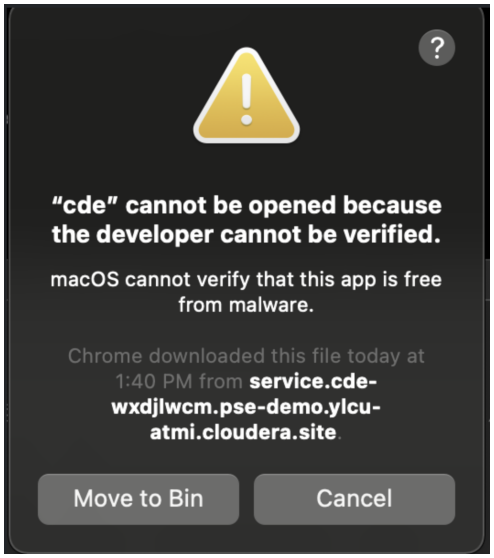


For Mac users:

- 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>"
```

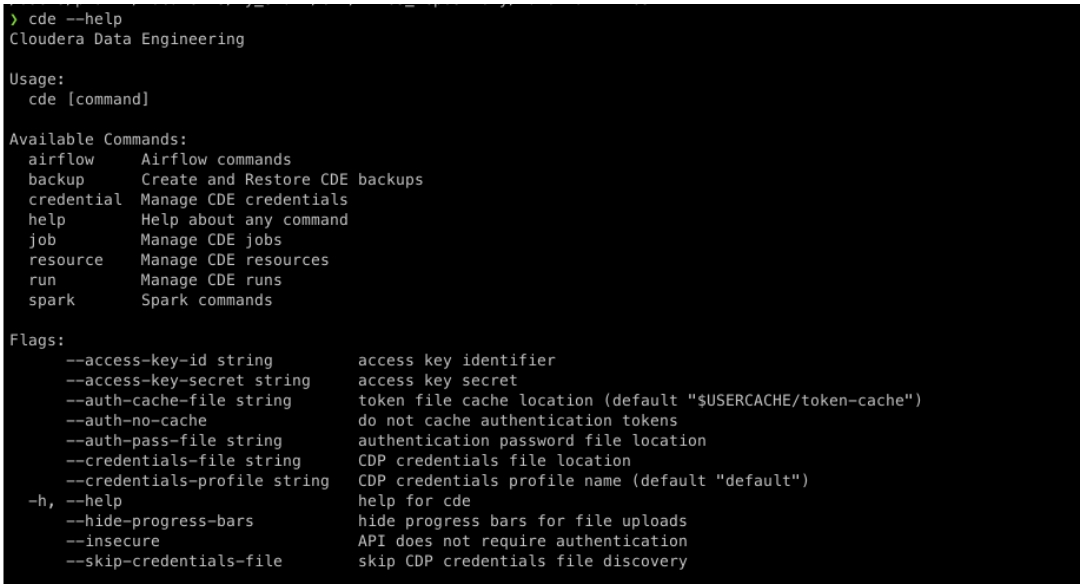
Example : `If the file is situated in the path /home/user/applications/cde`

`Then the export command will look something like this`

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

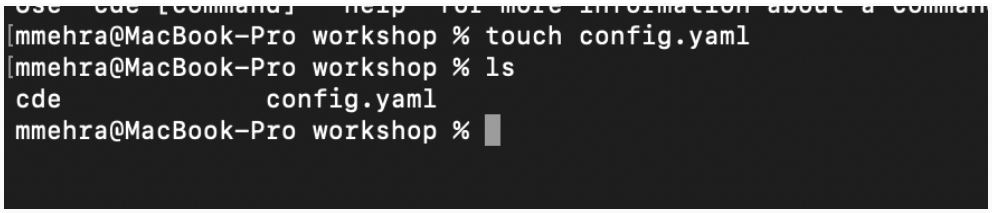
Note : Once the terminal is closed, the step to add the path to the PATH variable needs to be performed again.

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



- 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
```




```
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
```

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

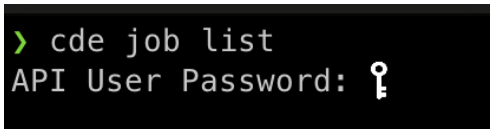
  ```
  export CDE_CONFIG=/path/to/config.yaml
  ```

- 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
  ```



- 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
  ```

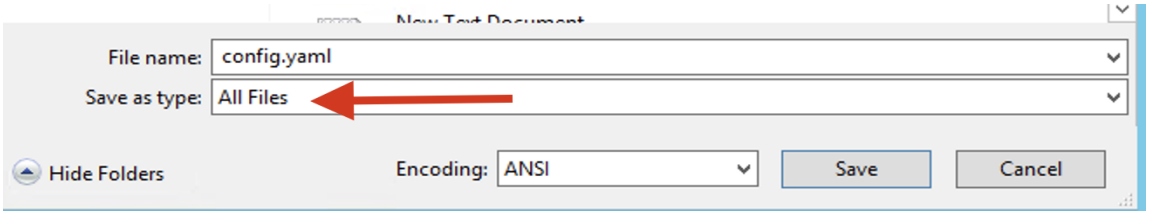
```
PS C:\Users\cdp_windows> cd C:\Users\cdp_windows\Desktop
PS C:\Users\cdp_windows\Desktop> ls


    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. 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
```

- 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

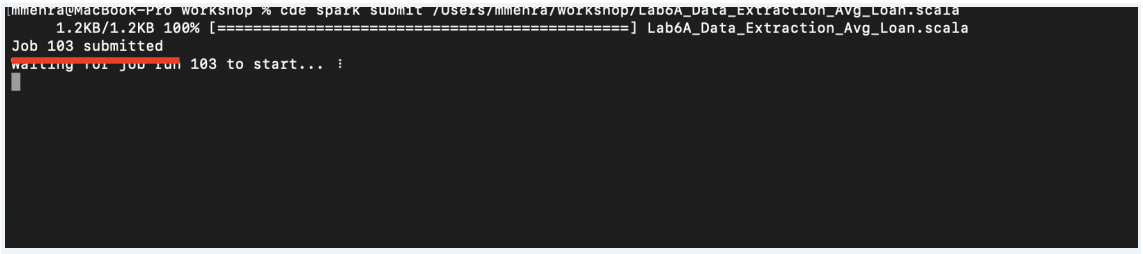
Run a spark-scala job using CLI

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.

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>
  ```

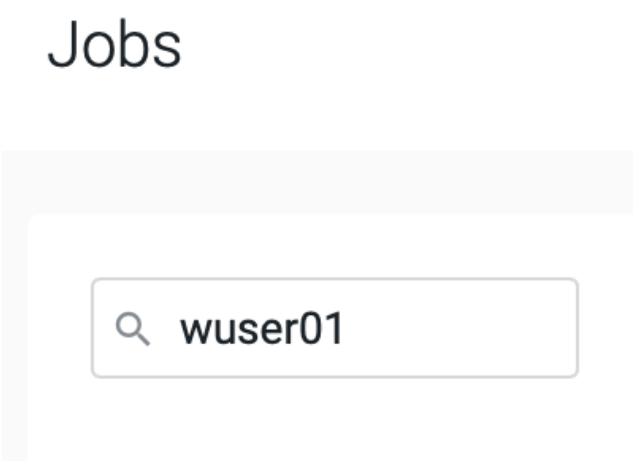
- This completes the exercise of Lab6.

Lab 7 - Data Lineage and Auto-Scaling

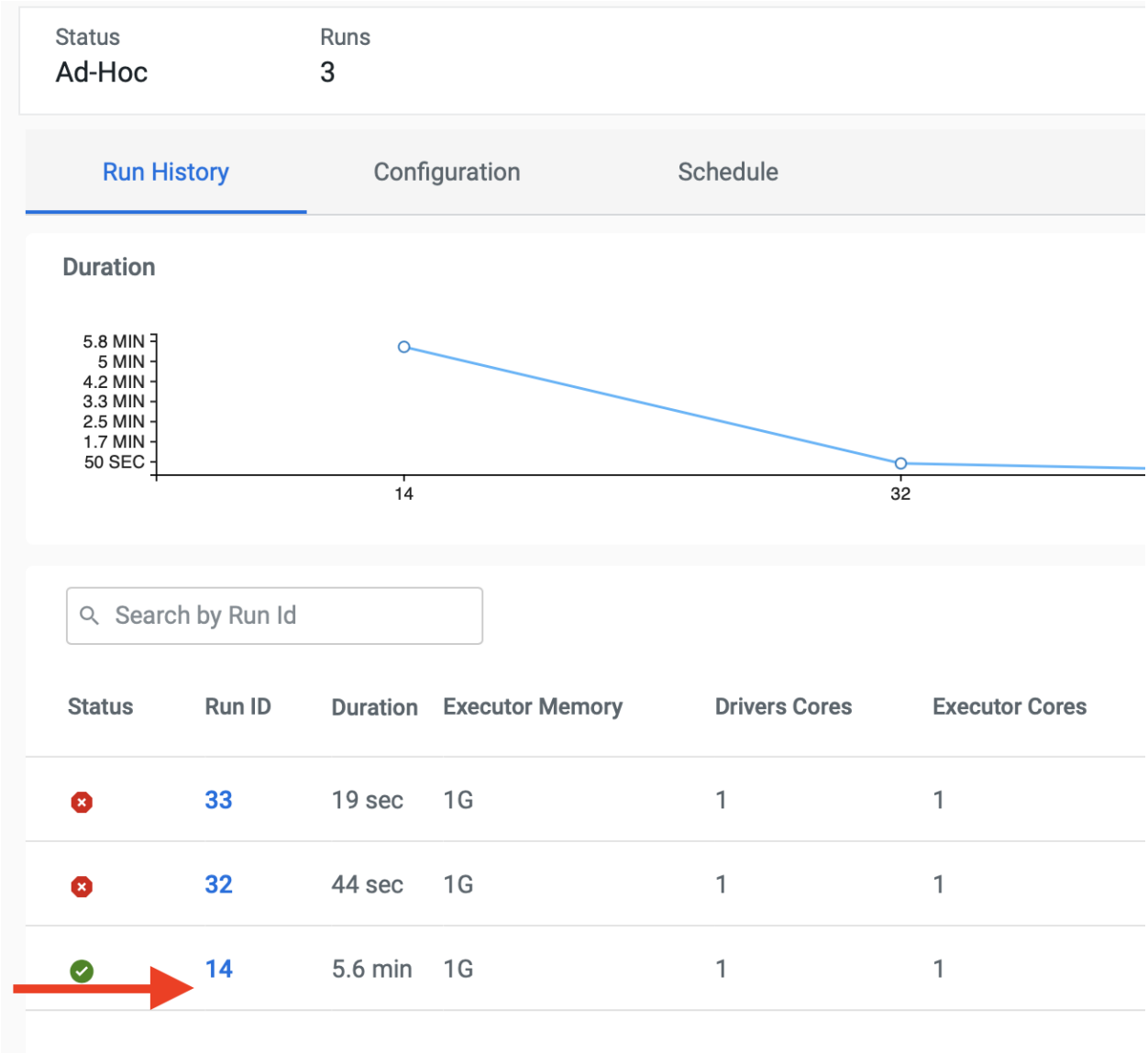
In this lab, you will go through the data lineage of the two use cases that we worked on. Additionally, you will also see the auto-scaling capabilities of CDE service with the rising demand for compute resources.

Data Lineage using Atlas

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

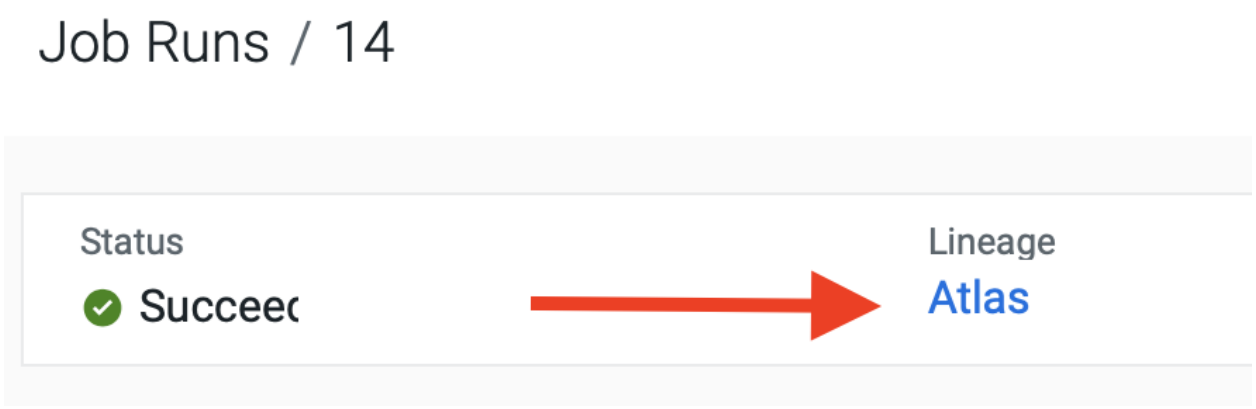


- In **Run History** tab, click on the successful Run ID i.e., the one with the green tick mark.

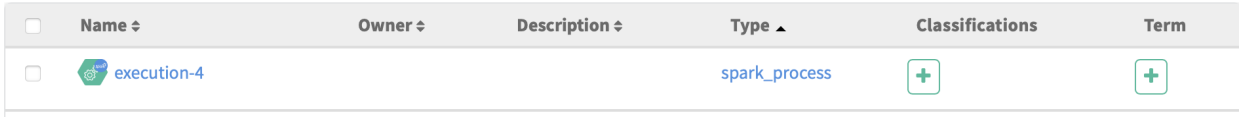


(Note the ID will be different for you from the one you see in the screenshot)

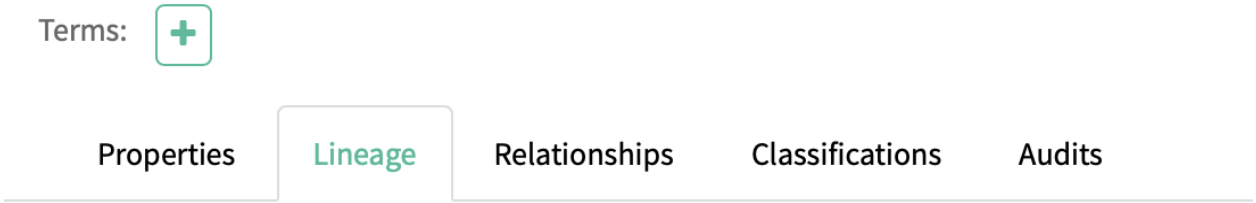
- Click on **Atlas** under Lineage.

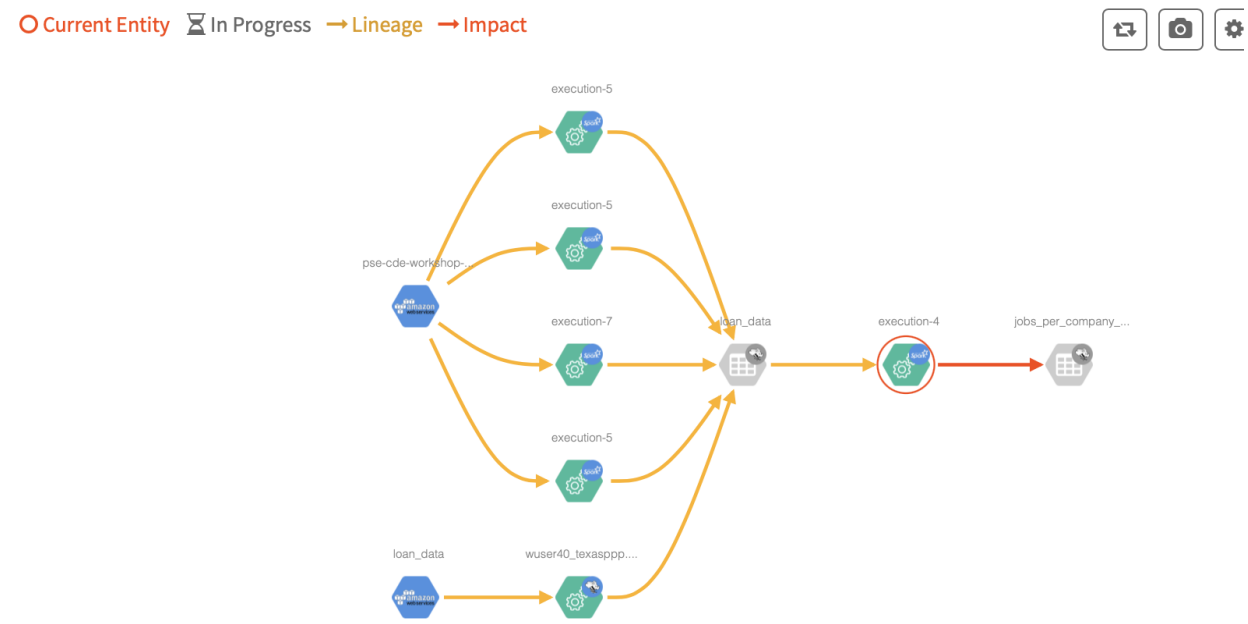


- Click on the execution that you see in the list.



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

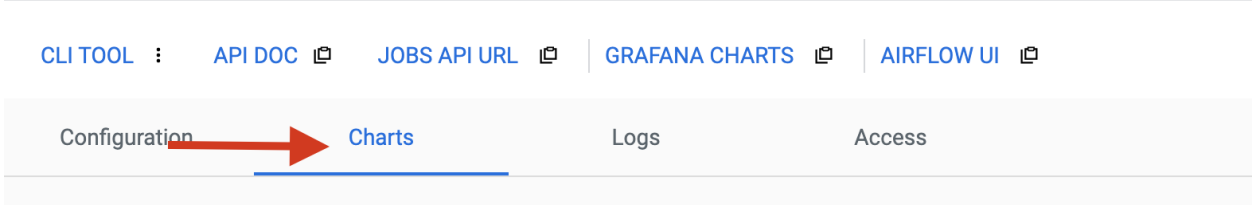




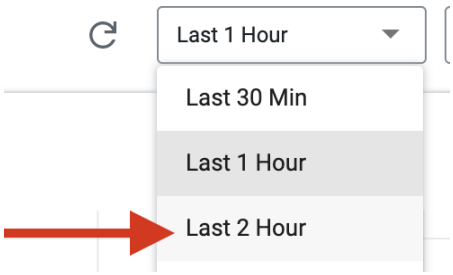
- Click on each entity to understand how the data is flowing from source to consumption.
- Follow the same steps for the last job you submitted using CLI.

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.



- 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

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