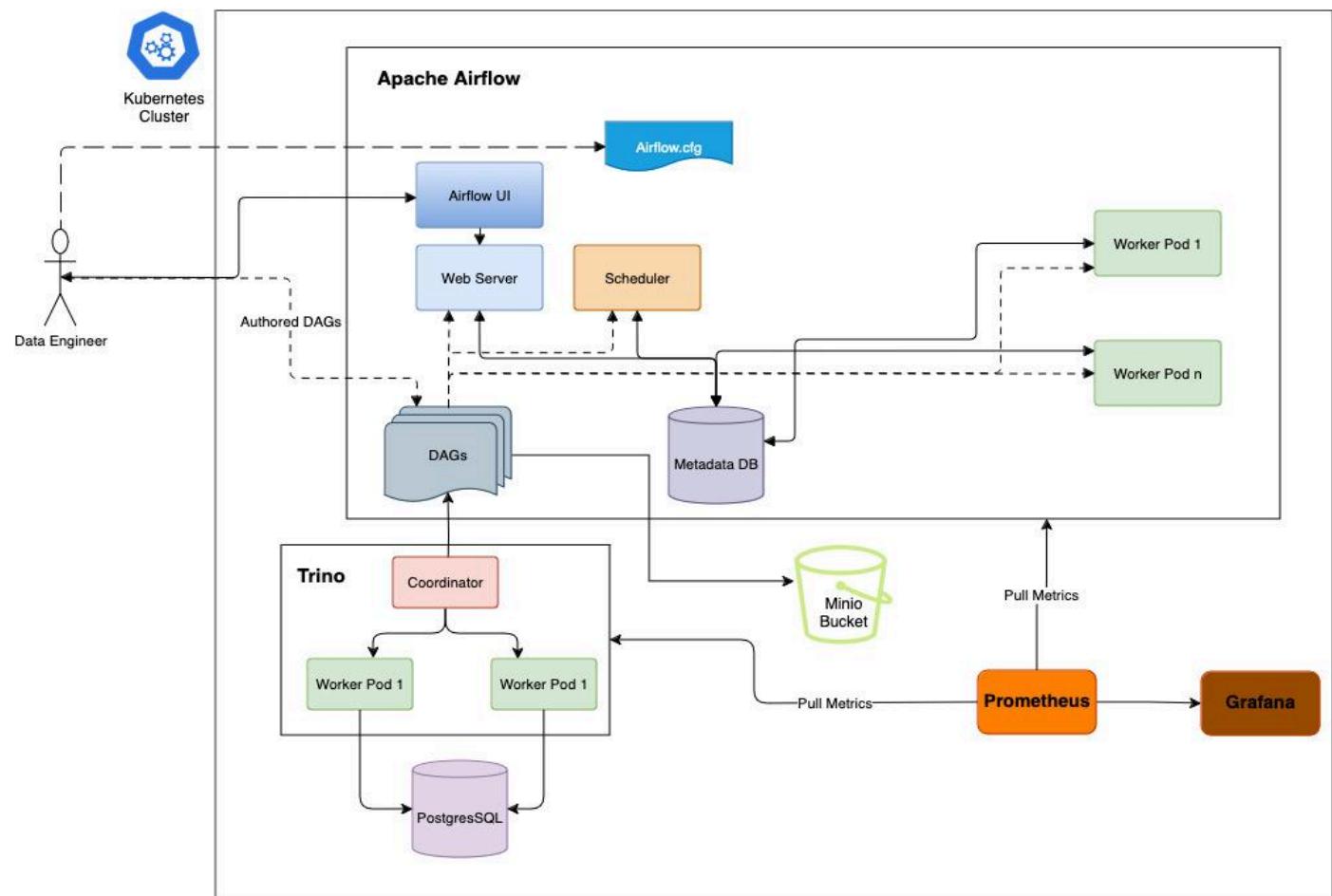


# Platform | Hiring | Quidditch

## Overview

This document provides detailed instructions for deploying a scalable Apache Airflow and Starburst Presto setup on Kubernetes. The goal is to demonstrate the orchestration of complex workflows with Airflow and the execution of distributed SQL queries on Presto, while ensuring scalability and efficient resource management on Kubernetes.

## Architecture Diagram:



# Prerequisites

- **Kubernetes Cluster:** Minikube.
- **kubectl:** Command-line tool to interact with your Kubernetes cluster.
- **Helm:** Kubernetes package manager for managing applications.

## 1. Kubernetes Setup

### 1.1 Minikube Setup

#### 1. Start Minikube with Multiple Nodes:

- minikube start --nodes=3

#### 1.2 Challenges:

Minikube wasn't starting as it was giving **Docker is out of disk space** error. Also upon checking minikube status (minikube status -p minikube), it was returning **host: InsufficientStorage**, so cleaned up docker by running the following commands:

- docker builder prune
- docker system prune
- docker image prune
- docker rm -f \$(docker ps -a -q)

Then it started giving another error which is **Enabling 'default-storageclass' returned an error**. After doing some digging came up with the following commands that actually fixed all the errors above:

- rm -rf ~/.minikube
- minikube delete --all --purge
- docker system prune
- minikube start --nodes=3

Output of minikube status -p minikube:

```
minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

## 2. Apache Airflow Deployment

### 2.1 Deployment with Helm

1. Pull custom docker image:
  - o docker pull pratishmahajan26/airflow-trino:latest
2. Install Airflow using Helm:
  - o cd quidditch/helm-chart
  - o helm install airflow ./airflow \  
--set images.airflow.repository=pratishmahajan26/airflow-trino \  
--set images.airflow.tag=latest \  
--set images.airflow.pullPolicy=Always
3. Access Airflow UI:
  - o Airflow Webserver: kubectl port-forward svc/airflow-webserver 8080:8080  
--namespace airflow
  - o Visit http://127.0.0.1:8080 to use the application
  - o Default Webserver (Airflow UI) Login credentials:
    - username: admin
    - password: admin
  - o Default Airflow Postgres connection credentials:
    - username: postgres
    - password: postgres
    - port: 5432
4. To check pods:
  - o kubectl get pods -o wide

### 2.2 Scaling Airflow

1. Horizontal Pod Autoscaler (HPA):
  - o kubectl autoscale deployment airflow-worker --namespace airflow --min=3  
--max=10 --cpu-percent=80
2. Manual Scaling:
  - o kubectl scale deployment airflow-worker --replicas=5 --namespace airflow

### 2.3 Secure Airflow Web Interface

#### 1. Basic Authentication:

```
web:  
  auth:  
    basicAuth:  
      enabled: true  
      username: admin  
      password: admin
```

## 3. PostgreSQL Deployment

### 3.1 Deployment with Helm

1. **Install Postgres Using Helm:**
  - helm install my-postgres oci://registry-1.docker.io/bitnamicharts/postgresql
2. **Get Password:**
  - export POSTGRES\_PASSWORD=\$(kubectl get secret --namespace default my-postgres-postgresql -o jsonpath="{.data.postgres-password}" | base64 -d)
  - echo \$POSTGRES\_PASSWORD
3. **Expose Postgres port for connecting it with SQL client:**
  - kubectl port-forward --namespace default svc/my-postgres-postgresql 5432:5432

## 4. Trino Deployment

### 4.1 Deployment with Helm

1. **Install Trino Using Helm:**
  - helm install -f trino/custom-values.yml trino ./trino
  - In custom-values.yml, port value has been changed to 8085 as Airflow is utilizing 8080. Also, a connector for postgresql is added under additionalCatalogs as rdbms. Only these values are overridden with the default values in values.yaml.
2. **Access Trino Dashboard:**
  - export POD\_NAME=\$(kubectl get pods --namespace default --selector "app.kubernetes.io/name=trino,app.kubernetes.io/instance=example-trino-cluster, app.kubernetes.io/component=coordinator" --output name)
  - echo \$POD\_NAME
  - kubectl port-forward \$POD\_NAME 8085:8085
3. **Run local Trino server for executing queries on shell:**
  - brew install trino
  - trino --server http://localhost:8080
  - select count(\*) from rdbms.public.sales;
4. **Reference to create your own yaml file:**  
<https://trino.io/docs/current/installation/kubernetes.html#creating-your-own-yaml>

## 5. Challenges Faced

5.1 Running Trino and Airflow pods:

- One of the biggest challenges I faced was running Airflow and Trino pods simultaneously. When Trino was being deployed after Airflow and PostgreSQL, the pods couldn't initialize, and the Airflow scheduler also started failing.
- After struggling and experimenting with the configurations, I finally figured out that the issue was due to insufficient resources allocated to Minikube. It wasn't able to provide enough memory to both Airflow and Trino. All pods are dependent on the resources (CPU and memory) that Minikube has.
- To resolve this, I restarted Minikube after deleting the existing Minikube cluster, allocating more resources by providing 6 CPUs and 8192 MB of memory, by running the following command:  
`minikube start --cpus=6 --memory=8192`
- Additionally, the minikube dashboard was very handy for checking error logs when deploying pods, which can be activated by running the following command:  
`minikube dashboard`
- I also had to change the port of Trino, as both Airflow and Trino were initially configured to run on the same port, 8080. Because of this conflict, the Airflow webserver couldn't start since the port was already bound to Trino.

## 5.2 Connect Trino to Postgres

The problem was addressed as follows:

### Add additional catalog:

- Create custom-values.yaml file in trino helm-chart directory

- Add the following:

```

○ additionalCatalogs:
  rdbms: |-
    connector.name=postgresql
    connection-url=jdbc:postgresql://my-postgres-postgresql.default.svc.cluster
    .local:5432/postgres
    connection-user=postgres
    connection-password=<postgres password>
  
```

### Resolve connection failed error:

- I was using the wrong host, i.e., `localhost` and the IP address of the PostgreSQL pod, in the connection-url above, for creating the connection with PostgreSQL. After digging into the issue and checking the output logs of the `helm install postgres` command, I discovered that PostgreSQL can be accessed via port 5432 on the following DNS name from within the cluster:  
`my-postgres-postgresql.default.svc.cluster.local.`
- After adding this DNS name as the host, Trino was able to connect with PostgreSQL and fetch the data successfully. Additionally, it's important to ensure that both pods are in the same Kubernetes namespace for seamless communication.

### Upgrade trino with reinstalling:

- `helm upgrade -f trino/custom-values.yaml trino ./trino`

## 5.3 Connect Airflow to Trino

### Installation of trino provider for apache airflow:

- After trying to install it using `pip install apache-airflow-providers-trino`, directly on the running container, it was not getting active and couldn't see Trino in connection type dropdown in Airflow Connections. So I created the docker file for building the custom docker image for Airflow having this provider installed, which can be found here:  
<https://github.com/pratishmahajan26/quidditch/blob/main/airflow.Dockerfile>
- Docker build command:
  - `cd quidditch && docker build -t airflow-trino:latest -f airflow.Dockerfile .`
- But I couldn't figure out how to use this image with helm charts, so I came up with the following solution, after trying out multiple ways and it took me a lot of time to figure this out as I couldn't create helm charts locally to use the custom image.
- Solution:
  - Check the latest version of apache/airflow which is pulled through helm install, by running docker images command.
  - Update the tag of the custom image to the latest version of apache/airflow, so next time whenever helm installs airflow, it will automatically use the updated tag and it will deploy airflow using the custom image.
  - `docker tag airflow-custom:latest apache/airflow:2.9.2`  
`helm install airflow apache-airflow/airflow`  
`kubectl exec -ti airflow-scheduler-88575d458-k5ckv -- bash`
  - As both are using same namespace, so airflow can only connect to trino using trino-coordinator ip which could be get using:  
`kubectl get pods -o wide`
  - Also this had to be added in Extra when adding the connection for trino  
`{ "impersonate_as_owner": "true" }`

## 5.4 Helm charts on local:

### Installation of Airflow using helm charts on local:

- Push docker image to docker hub
  - Build docker image  
`cd quidditch && docker build -t airflow-trino:latest -f airflow.Dockerfile .`
  - Login to docker hub:  
`docker login -u "<user_name>" -p '<password>'`
  - Tag docker image to repository on docker hub:  
`docker tag airflow-trino:latest pratishmahajan26/airflow-trino:latest`

- Push docker image

```
docker push pratishmahajan26/airflow-trino:latest
```

- **Install airflow:**

- cd quidditch/helm-chart

- Change the following properties in airflow/values.yaml:

```
airflow:
  repository: pratishmahajan26/airflow-trino
  tag: latest
```

- Helm install by passing values in the command:

```
helm install airflow ./airflow \
--set images.airflow.repository=pratishmahajan26/airflow-trino \
--set images.airflow.tag=latest \
--set images.airflow.pullPolicy=Always
```

- **Managing dags:**

- If there is any change in the dag or in the custom operator, then the above procedure can be followed, as it is required to deploy airflow in order to reflect the changes.

- For development purposes the dag changes can be copied directly in all the containers running for Scheduler, Webserver & Worker:

```
kubectl cp ~/quidditch/airflow-dags/trino_minio.py -n default
airflow-scheduler-98f9f996f-9fg8n:/opt/airflow/dags
```

```
kubectl cp ~/quidditch/airflow-dags/trino_minio.py -n default
airflow-webserver-5644b4bb95-nkzdm:/opt/airflow/dags
```

```
kubectl cp ~/quidditch/airflow-dags/trino_minio.py -n default
airflow-worker-0:/opt/airflow/dags
```

## 6. MiniO Deployment

### 6.1 Deployment with Helm

#### 1. Install Minio Using Helm:

- helm install --set resources.requests.memory=512Mi --set replicas=1 --set persistence.enabled=false --set mode=standalone --set rootUser=rootuser,rootPassword=rootpass123 --generate-name minio/minio

#### 2. Access MinIO server:

- export POD\_NAME=\$(kubectl get pods --namespace default -l "release=minio-1721640558" -o jsonpath="{.items[0].metadata.name}")
- kubectl port-forward \$POD\_NAME 9000:9001 --namespace default
- Access MinIO server on <http://localhost:9000>

#### 3. Connect Airflow to MiniO:

- Add connection on Airflow by selecting Amazon Web Services in Connection Type and provide the following details:
  - Conn Id: `my_s3_conn`
  - Get the Ip of minio pod and add it to the below config
  - Extra:
 

```
{
    "aws_access_key_id": "rootuser",
    "aws_secret_access_key": "rootpass123",
    "endpoint_url": "http://10.244.0.136:9000"
}
```
  - Create bucket by accessing MinIO server on <http://localhost:9000>

## 7. Scalability and Monitoring

### 7.1 Scaling Airflow and Trino

- **Airflow Manual Scaling:**
  - kubectl scale statefulset airflow-worker --replicas=1 -n default
- **Trino Scaling:**
  - kubectl scale deployment trino-woker --replicas=6 --namespace trino
  - kubectl autoscale deployment trino-woker --namespace default --min=1 --max=10 --cpu-percent=80
- **Auto Scaling in Airflow Workers:**
  - In airflow values.yaml enable :
 

```
# Allow HPA (KEDA must be disabled) .
hpa:
  enabled: true

  # Minimum number of workers created by HPA
  minReplicaCount: 1

  # Maximum number of workers created by HPA
  maxReplicaCount: 3

  # Specifications for which to use to calculate the desired
  replica count
  metrics:
    - type: Resource
      resource:
        name: cpu
      target:
        type: Utilization
        averageUtilization: 80
```

- Upgrade Airflow to apply settings above:  

```
helm upgrade airflow ./airflow \
--set images.airflow.repository=pratishmahajan26/airflow-trino \
--set images.airflow.tag=latest \
--set images.airflow.pullPolicy=Always
```
- Using kubectl get pods, check if airflow-worker pod is running
- Now to check if everything is in place for Auto Scaling, set **minReplicaCount to 2** and upgrade airflow again. A new airflow worker will be added having the name **airflow-worker-1**.
- Set **minReplicaCount to 1** again and upgrade the Airflow. Ideally **airflow-worker-1** added in the previous step, should be scaled down (delete automatically based on the hpa configuration). But it won't as there are some issues which have to be fixed first.
- To check the issue run:
  - i. kubectl get hpa -n default
  - ii. kubectl get hpa airflow-worker

pratish.mahajan@Pratishs-MacBook-Pro ~ % kubectl get hpa
NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE
airflow-worker StatefulSet/airflow-worker cpu: <unknown>/80% 1 3 2 2m53s

  - iii. kubectl describe hpa airflow-worker -n default  

```
failed to get cpu utilization: unable to get metrics for resource cpu: no
metrics returned from resource metrics API
invalid metrics (1 invalid out of 1), first error is: failed to get cpu resource
metric value: failed to get cpu utilization: did not receive metrics for
targeted pods (pods might be unready)
invalid metrics (1 invalid out of 1), first error is: failed to get cpu resource
metric value: failed to get cpu utilization: unable to get metrics for
resource cpu: unable to fetch metrics from resource metrics API: the
server could not find the requested resource (get pods.metrics.k8s.io)
failed to get cpu utilization: missing request for cpu in container
worker-log-groomer of Pod airflow-worker-0
```
  - iv. There could be issues related to fetching of metrics which can be resolved by performing various steps.
- Steps to resolve the above issues:
  - i. Enable kubernetes metrics-server:  

```
[kubectl apply -f
https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml]
minikube addons enable metrics-server
Reference:
Hello Minikube | Kubernetes
How to fix failed get resource metric in Kubernetes HPA
```
  - ii. To Verify:  

```
kubectl get deployment metrics-server -n kube-system
kubectl logs deployment/metrics-server -n kube-system
```

```
kubectl top pods
```

iii. Update airflow values.yaml:

Under workers, set resources to:

```
resources:  
  requests:  
    cpu: 100m  
    memory: 500Mi
```

And resource under `workers.logGroomerSidecar` to:

```
logGroomerSidecar:  
  # Whether to deploy the sidecar  
  enabled: true  
  # Command to use when running the sidecar  
  command: ~  
  # Args to use when running the sidecar  
  args: ["bash", "/clean-logs"]  
  # Number of days to retain logs  
  retentionDays: 15  
  resources:  
    # limits:  
    #   cpu: 100m  
    #   memory: 128Mi  
    requests:  
      cpu: 100m  
      memory: 128Mi  
  # Detailed default security contexts for the container  
  securityContexts:  
    container: {}
```

Explanation of this is given in

[How to fix failed get resource metric in Kubernetes HPA](#)

- Upgrade airflow again, and check pods, the worker pds will scale down to 1.
- Run: kubectl get hpa airflow-worker

```
[pratish.mahajan@Pratish-MacBook-Pro Documents % kubectl get hpa airflow-worker  
NAME          REFERENCE          TARGETS      MINPODS   MAXPODS   REPLICAS   AGE  
airflow-worker  StatefulSet/airflow-worker  cpu: 25%/80%  1         3          1          87m
```

- Autoscaling with KEDA:
  - [Autoscaling with KEDA — helm-chart Documentation](#)
- Additional Commands:
  - minikube addons disable metrics-server
  - kubectl top pods
  - kubectl get pod,svc -n kube-system
  - minikube delete
  - kubectl get --raw /apis/[metrics.k8s.io/v1beta1](#)
  - kubectl hpa delete airflow-worker
  - kubectl get deployment metrics-server -n kube-system
  - kubectl get statefulset
  - kubectl get statefulset <statefulset-name> -n <namespace>

## 7.2 Monitoring with Prometheus and Grafana

### 1. Deploy Prometheus and Grafana:

```
## Prometheus
- helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts
- helm repo update
- helm install prometheus prometheus-community/prometheus
- export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/name=prometheus,app.kubernetes.io/instance=prometheus"
-o jsonpath=".items[0].metadata.name")
- kubectl --namespace default port-forward $POD_NAME 9090
- Visit http://127.0.0.1:9090 to use the application

## Grafana
- helm repo add grafana https://grafana.github.io/helm-charts
- helm repo update
- helm install grafana grafana/grafana
- Prometheus Server for Grafana:
http://prometheus-server.default.svc.cluster.local
- Get your 'admin' user password by running:
  - kubectl get secret --namespace default grafana -o
jsonpath=".data.admin-password" | base64 --decode ; echo
- export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/name=grafana,app.kubernetes.io/instance=grafana" -o
jsonpath=".items[0].metadata.name")
- kubectl --namespace default port-forward $POD_NAME 3000
- Visit http://127.0.0.1:3000 to use the application

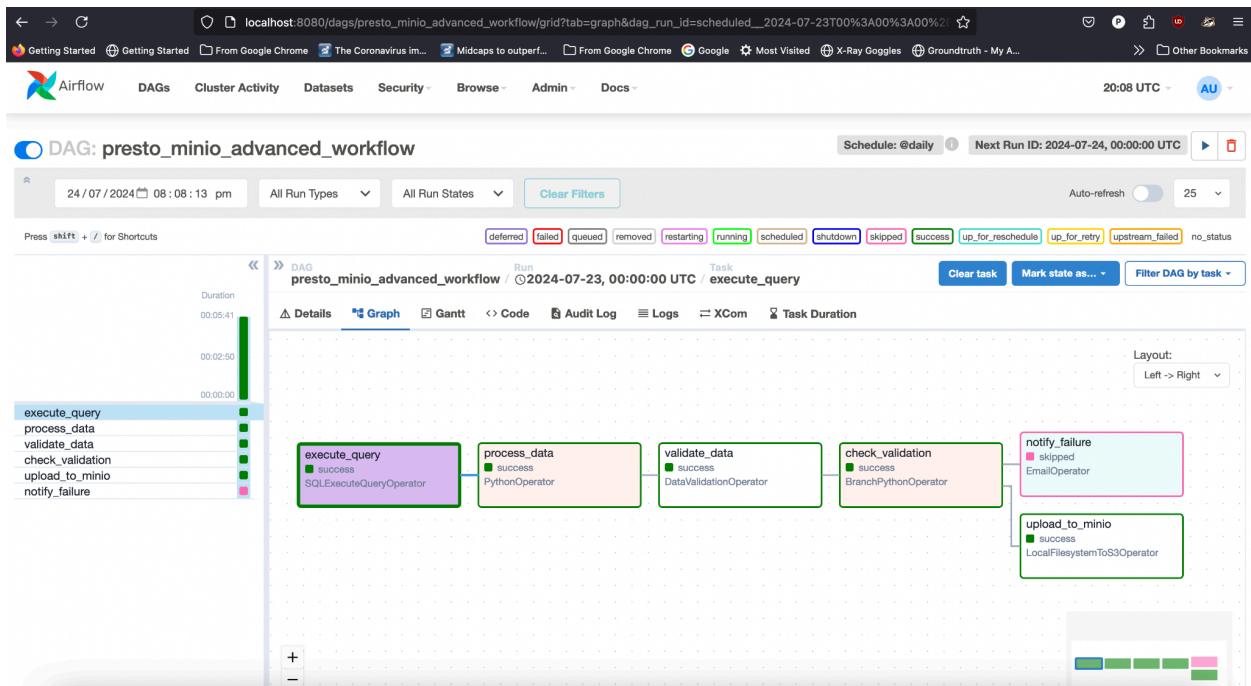
## Minikube Dashboard
- minikube dashboard
- It will automatically open a new tab on the browser
- Enable & disable Metrics Server:
- minikube addons enable metrics-server
- minikube addons disable metrics-server
```

## 8. Screenshots

```
pratish.mahajan@Pratishs-MacBook-Pro h1-data % trino --server http://localhost:8085
trino> select * from rdbms.public.sales;
+-----+-----+-----+-----+-----+-----+-----+
| sale_id | sale_date | customer_name | product_name | quantity | price | total |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 2023-07-01 | John Doe | Laptop | 1 | 1200.00 | 1200.00 |
| 2 | 2023-07-02 | Jane Smith | Smartphone | 2 | 600.00 | 1200.00 |
| 3 | 2023-07-03 | Alice Johnson | Tablet | 1 | 300.00 | 300.00 |
| 4 | 2023-07-04 | Bob Brown | Smartwatch | 3 | 150.00 | 450.00 |
| 5 | 2023-07-05 | Charlie Davis | Headphones | 5 | 80.00 | 400.00 |
+-----+-----+-----+-----+-----+-----+-----+
(5 rows)
```

```
Query 20240724_153853_00002_8cs77, FINISHED, 1 node
Splits: 1 total, 1 done (100.00%)
0.76 [5 rows, 0B] [6 rows/s, 0B/s]
```

```
trino>
```



localhost:8080/home

Getting Started | Getting Started | From Google Chrome | The Coronavirus in... | Midcaps to oupt... | From Google Chrome | Google | Most Visited | X-Ray Goggles | Groundtruth - My A... | Other Bookmarks

20:08 UTC AU

**DAGs**

All 2 Active 2 Paused 0

Running 0 Failed 0

Filter DAGs by tag

Search DAGs

Auto-refresh C

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
presto_minio_advanced_workflow	airflow	1	@daily	2024-07-23, 00:00:00	2024-07-24, 00:00:00	1, 2, 3, 4, 5, 6, 7, 8	▶ ⚡ ...	
presto_minio_workflow	airflow	1	@daily	2024-07-23, 00:00:00	2024-07-24, 00:00:00	1, 2, 3, 4, 5, 6, 7, 8	▶ ⚡ ...	

Showing 1-2 of 2 DAGs

Version: v2.9.2  
Git Version: .release:f56f13442613912725d307aafc537cc76277c2d1

localhost:8080/dags/presto\_minio\_workflow/grid?tab=graph&dag\_run\_id=scheduled\_2024-07-23T00%3A00%3A00%2B00%3A00

Getting Started | Getting Started | From Google Chrome | The Coronavirus in... | Midcaps to oupt... | From Google Chrome | Google | Most Visited | X-Ray Goggles | Groundtruth - My A... | Other Bookmarks

20:09 UTC AU

**DAG: presto\_minio\_workflow**

Schedule: @daily | Next Run ID: 2024-07-24, 00:00:00 UTC | ⚡ ⚡

24 / 07 / 2024 08 : 08 : 58 pm All Run Types All Run States Clear Filters Auto-refresh 25

Press shift + / for Shortcuts

Duration: 00:00:06

Graph Gantt Code Audit Log Logs XCom Task Duration

Details Graph Gantt Code Audit Log Logs XCom Task Duration

Layout: Left -> Right

execute\_query process\_data upload\_to\_minio

execute\_query SQLExecuteQueryOperator process\_data PythonOperator upload\_to\_minio LocalFilesystemToS3Operator

localhost:9001/browser/minio-airflow/cHJvY2zc2VkX2RhdGEv

Getting Started Getting Started From Google Chrome The Coronavirus im... Midcaps to outperf... From Google Chrome Google Most Visited X-Ray Goggles Groundtruth - My A... Other Bookmarks

MINIO OBJECT STORE LICENSE

User Object Browser Access Keys Documentation

Administrator Buckets Policies Identity Monitoring Events Tiering Site Replication Configuration

Object Browser

minio-airflow

Created on: Thu, Jul 25 2024 01:31:21 (GMT+5:30) Access: PRIVATE 308.0 B - 1 Object

Rewind Refresh Upload

minio-airflow / processed\_data

Name Last Modified Size

processed\_data.csv Today, 01:38 308.0 B

Create new path

This screenshot shows the MinIO Object Browser interface. On the left, there's a sidebar with navigation links for User, Object Browser, Access Keys, Documentation, and Administrator sections. Under the Administrator section, Buckets is selected. The main area displays a table of objects in the 'minio-airflow / processed\_data' bucket. There is one object named 'processed\_data.csv' with a size of 308.0 B and a last modified date of Today, 01:38.

localhost:9001/buckets

Getting Started Getting Started From Google Chrome The Coronavirus im... Midcaps to outperf... From Google Chrome Google Most Visited X-Ray Goggles Groundtruth - My A... Other Bookmarks

MINIO OBJECT STORE LICENSE

User Object Browser Access Keys Documentation

Administrator Buckets Policies Identity Monitoring Events Tiering Site Replication Configuration

Buckets

Search Buckets Create Bucket +

minio-airflow

Created: Thu Jul 25 2024 01:31:21 GMT+0530 (India Standard Time) Access: R/W

Usage Objects  
308.0 B 1

This screenshot shows the MinIO Buckets page. It features a sidebar with User, Object Browser, Access Keys, Documentation, and Administrator sections. The Buckets section is selected under Administrator. The main part of the screen shows the 'minio-airflow' bucket details. It was created on Thu Jul 25 2024 01:31:21 GMT+0530 (India Standard Time) and has R/W access. The usage is listed as 308.0 B and the number of objects as 1.

localhost:8090/ui/sql\_client.html

CLUSTER OVERVIEW RESOURCE GROUPS SQL CLIENT

VERSION 0.288-15F14BB ENVIRONMENT DEVELOPMENT UPTIME 5.91m

**SQL** Session Properties

Catalog Schema Run

```
SELECT * FROM rbms.public.sales limit 100
```

**Results**

sale_id	sale_date	customer_name	product_name	quantity	price	total
1	2023-07-01	John Doe	Laptop	1	1200.00	1200.00
2	2023-07-02	Jane Smith	Smartphone	2	600.00	1200.00
3	2023-07-03	Alice Johnson	Tablet	1	300.00	300.00
4	2023-07-04	Bob Brown	Smartwatch	3	150.00	450.00
5	2023-07-05	Charlie Davis	Headphones	5	80.00	400.00

Rows per page: 10 1-5 of 5 |< < > >|

localhost:8090/ui/sql\_client.html#

CLUSTER OVERVIEW RESOURCE GROUPS SQL CLIENT

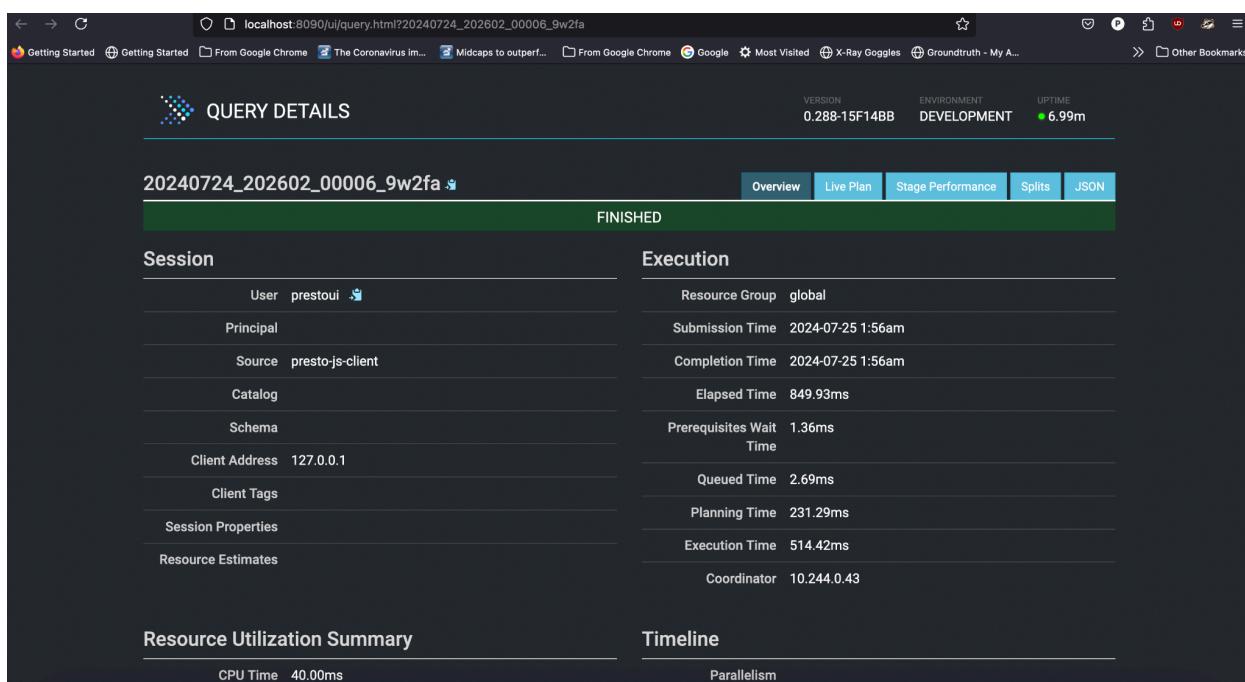
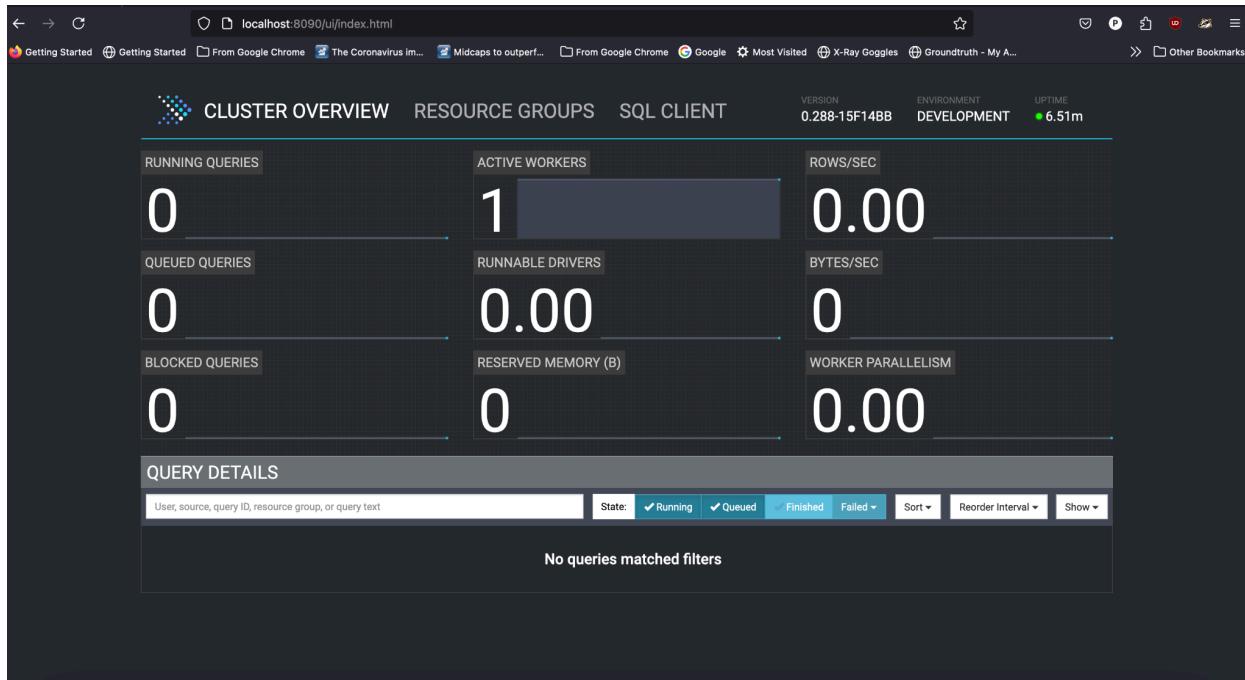
VERSION 0.288-15F14BB ENVIRONMENT DEVELOPMENT UPTIME 6.24m

**SQL** Session Properties

Filter By Name X

Name	Current Value	Default Value	Date Type	Description
adaptive_partial_aggregation	false	false	boolean	Enable adaptive partial aggregation
adaptive_partial_aggregation_unique_rows_ratio_threshold	0.8	0.8	double	Rows reduction ratio threshold at which to adaptively disable partial aggregation
add_partial_node_for_row_number_with_limit	true	true	boolean	Add partial row number node for row number node with limit
aggregation_if_to_filter_rewrite_strategy	DISABLED	DISABLED	varchar	Set the strategy used to rewrite AGG IF to AGG FILTER. Options are DISABLED,FILTER_WITH_IF,UNWRAP_IF_SAFE,UNWRAP_AP_IF
aggregation_operator_unspill_memory_limit	4MB	4MB	varchar	Experimental: How much memory can should be allocated per aggregation operator in unspilling process
aggregation_partitioning_merging_strategy	LEGACY	LEGACY	varchar	Strategy to merge partition preference in aggregation node. Options are LEGACY, TOP_DOWN,BOTTOM_UP
aggregation_spill_enabled	true	true	boolean	Enable aggregate spilling if spill_enabled
allow_window_order_by_literals	true	true	boolean	Allow ORDER BY literals in window functions
check_access_control_on_utilized_columns_only	false	false	boolean	Apply access control rules on only those columns that are required to produce the query output
check_access_control_with_subfields	false	false	boolean	Apply access control rules with subfield information from columns containing row types

Rows per page: 10 1-10 of 226 |< < > >|



## Minikube Dashboard (Addon: metrics-server)

