update code
Sai Maduri authored 6 days ago

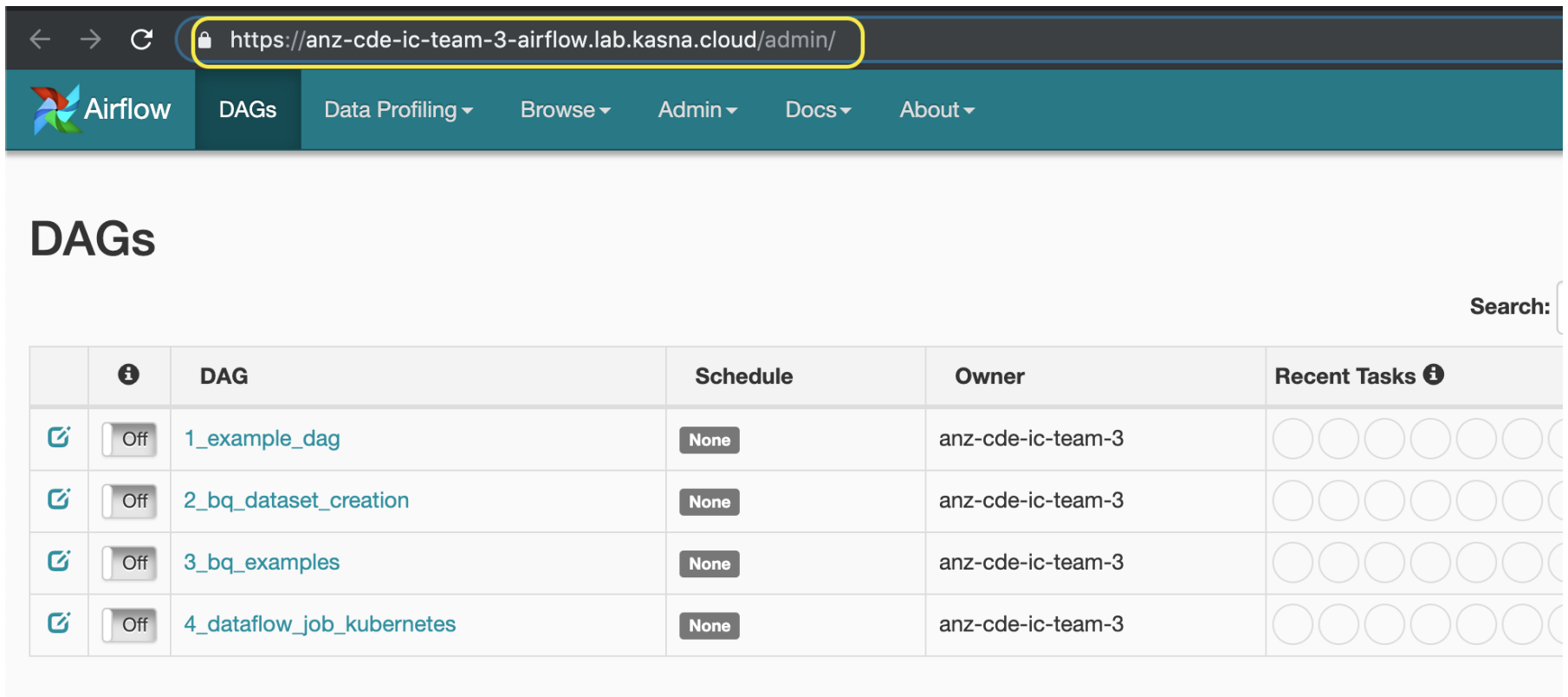
e467d41b

UserGuide.md 4.39 KB

Airflow UI Access

Every member of a team has access to a pre-created airflow instance and can log into the the airflow web-ui through https://{TEAM_ID}-airflow.lab.kasna.cloud. eg: for team1 https://anz-cde-ic-team-1-airflow.lab.kasna.cloud

The website is authenticated by google oauth credentials. users can login using their kasna google credentials



Airflow Jobs

Once users login to the web-ui there are some pre-defined sample jobs which are defined using job templates. Airflow has been pre-configured to periodically load the jobs(dags) uploaded to a google cloud storage bucket(gs://{TEAM_ID}-storage). The cloudbuild.yaml defines a cloud build job to upload the src directory containing job definitions to the storage bucket.

anz-cde-ic-team-3-storage

Objects Overview Permissions Bucket Lock

Upload files | Upload folder | Create folder | Manage holds | Delete

Filter by prefix...

Buckets / anz-cde-ic-team-3-storage / dags

Name	Size	Type	Storage class	Last modified
__init__.py	0 B	text/x-python	Regional	14/08/2019, 10:02:31 UTC+10
dag.py	494 B	text/x-python	Regional	14/08/2019, 10:02:31 UTC+10
schemas/	—	Folder	—	—
sql/	—	Folder	—	—
templates/	—	Folder	—	—

Yaml to Dags

By Convention airflow jobs are written in python. But we will be utilizing an existing library to dynamically create dags from yaml files.

For example to simple job to create two tasks with bash operator scheduled to run hourly. And task_2 depends on task_1

```
1_example_dag:
  default_args:
    start_date: 2019-08-10
    catchup: 'False'
    timezone: 'Australia/Melbourne'
  schedule_interval: '0 * * * *'
  description: 'this is example dag'
  tasks:
    task_1:
      operator: airflow.operators.bash_operator.BashOperator
      bash_command: 'echo 1'
    task_2:
      operator: airflow.operators.bash_operator.BashOperator
      bash_command: 'echo 2'
      dependencies: [task_1]
```

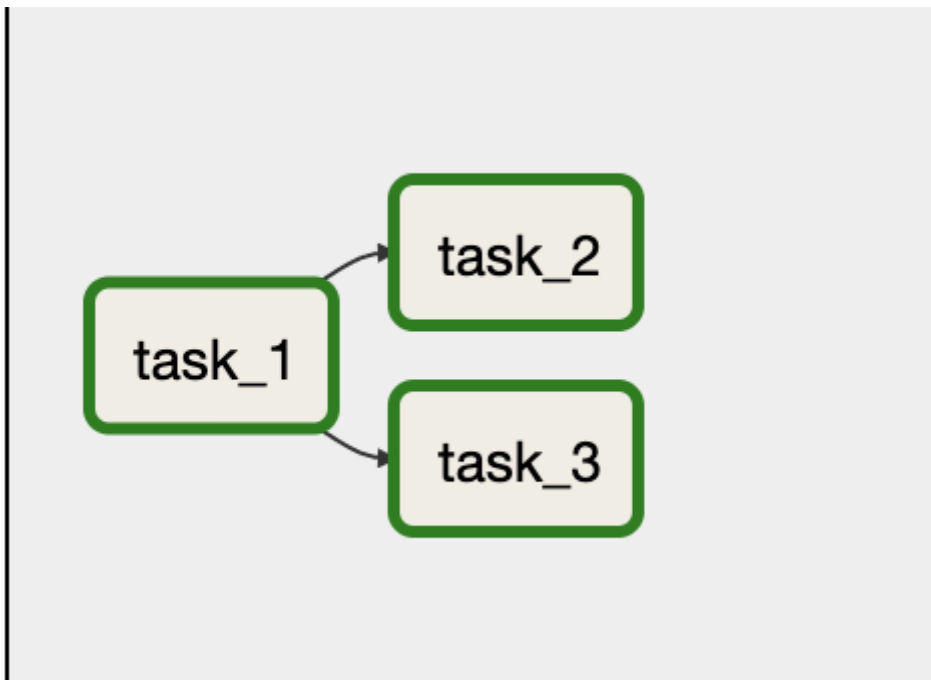
The dag.py file in the src will render dags from .yaml files in the templates directory. Can also customize timezone, schedule_interval and other airflow job options

Sample Jobs:

There are few sample jobs to start with

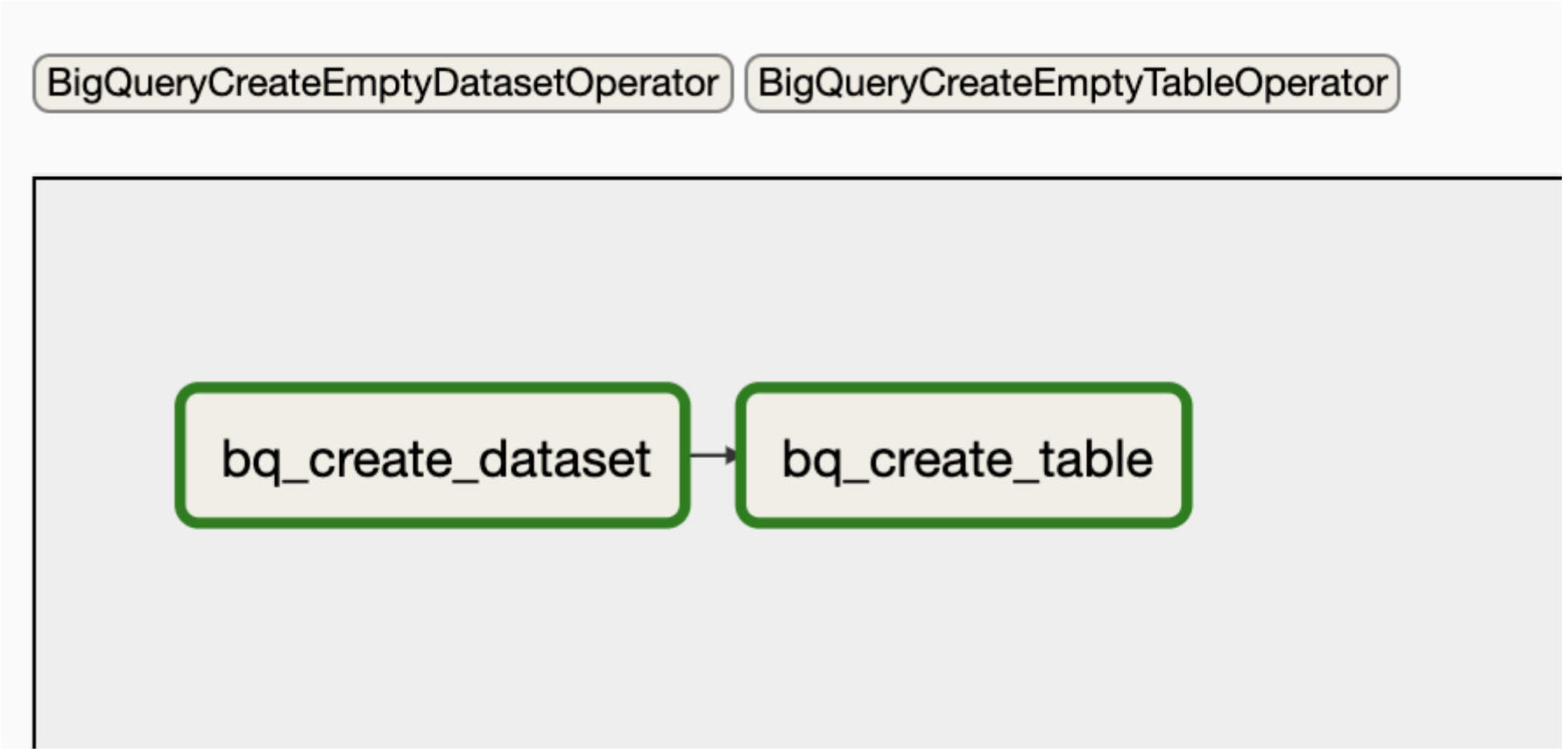
JOB1

This create a simple job with bash operators with three tasks, scheduled daily.

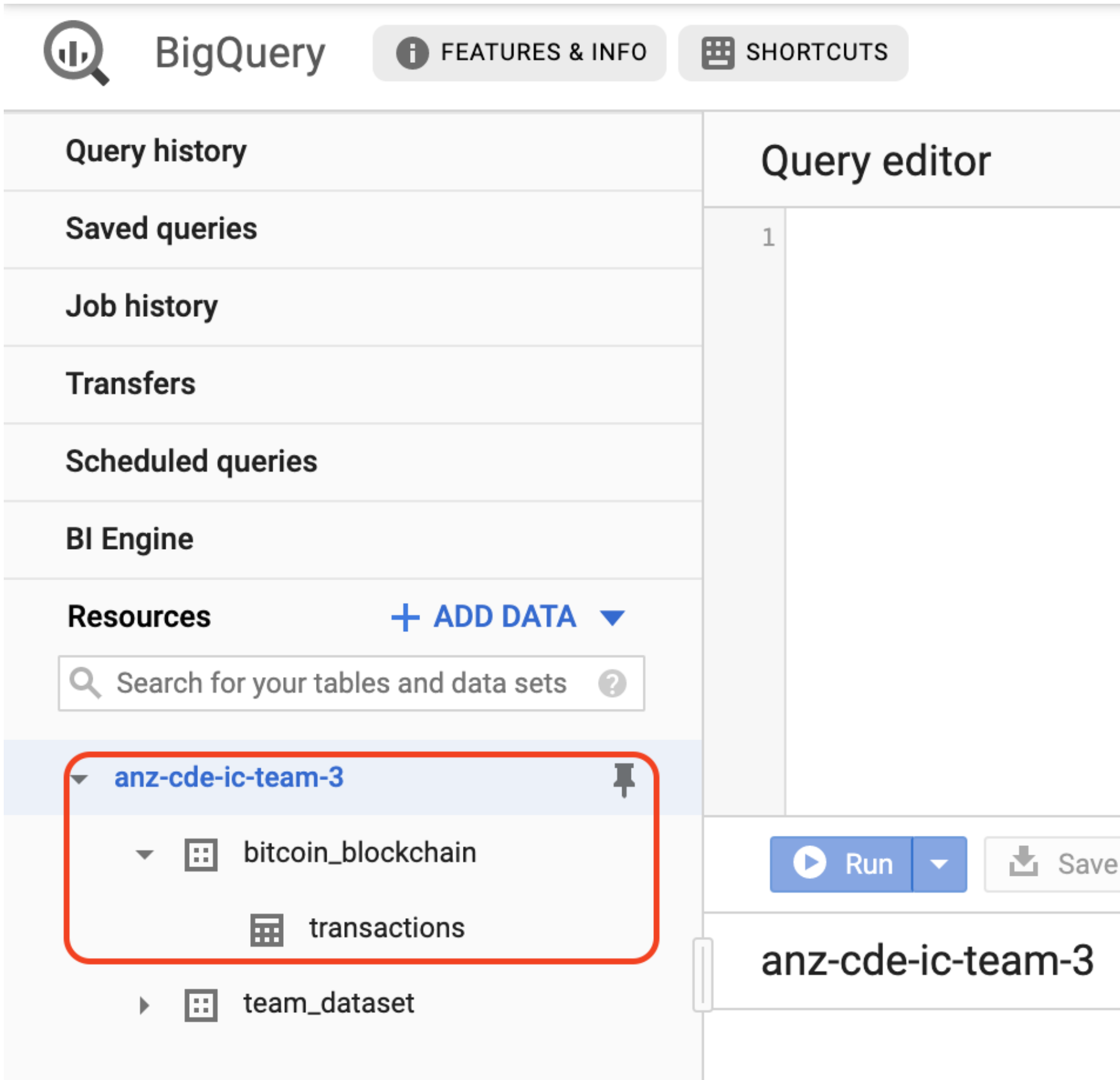


JOB2

This create a job using gcloud operators to create a bigquery dataset and a table



After the job is successful we should be able to see a biqquery dataset and a table are created utilizing the gcloud service credentials for the gke cluster



JOB3

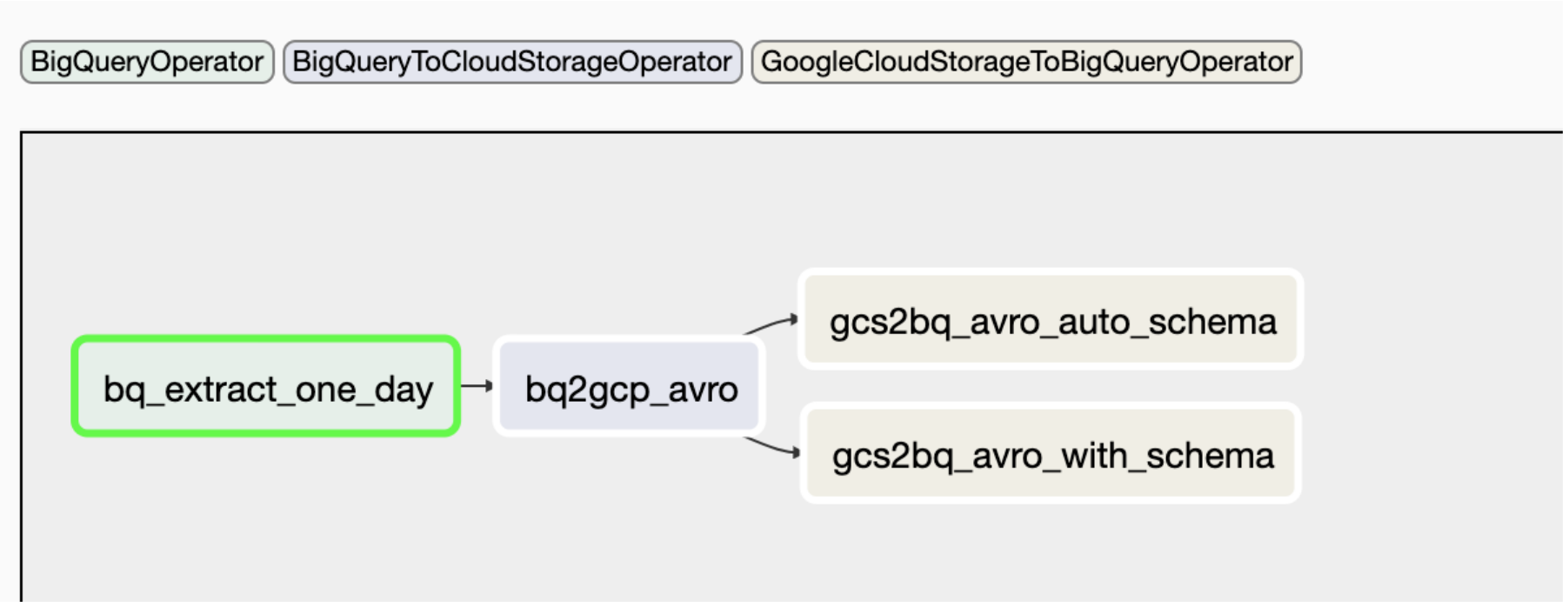
In this job we will utilize the above created dataset and table and explore more tasks we can do with gcloud operators. we will perform four task in this job.

The first task will run a sql query to query a public dataset and create a dump the data into the table we created in the earlier job. we run the sql query from a file uploaded to the cloud storage from the src/sql directory

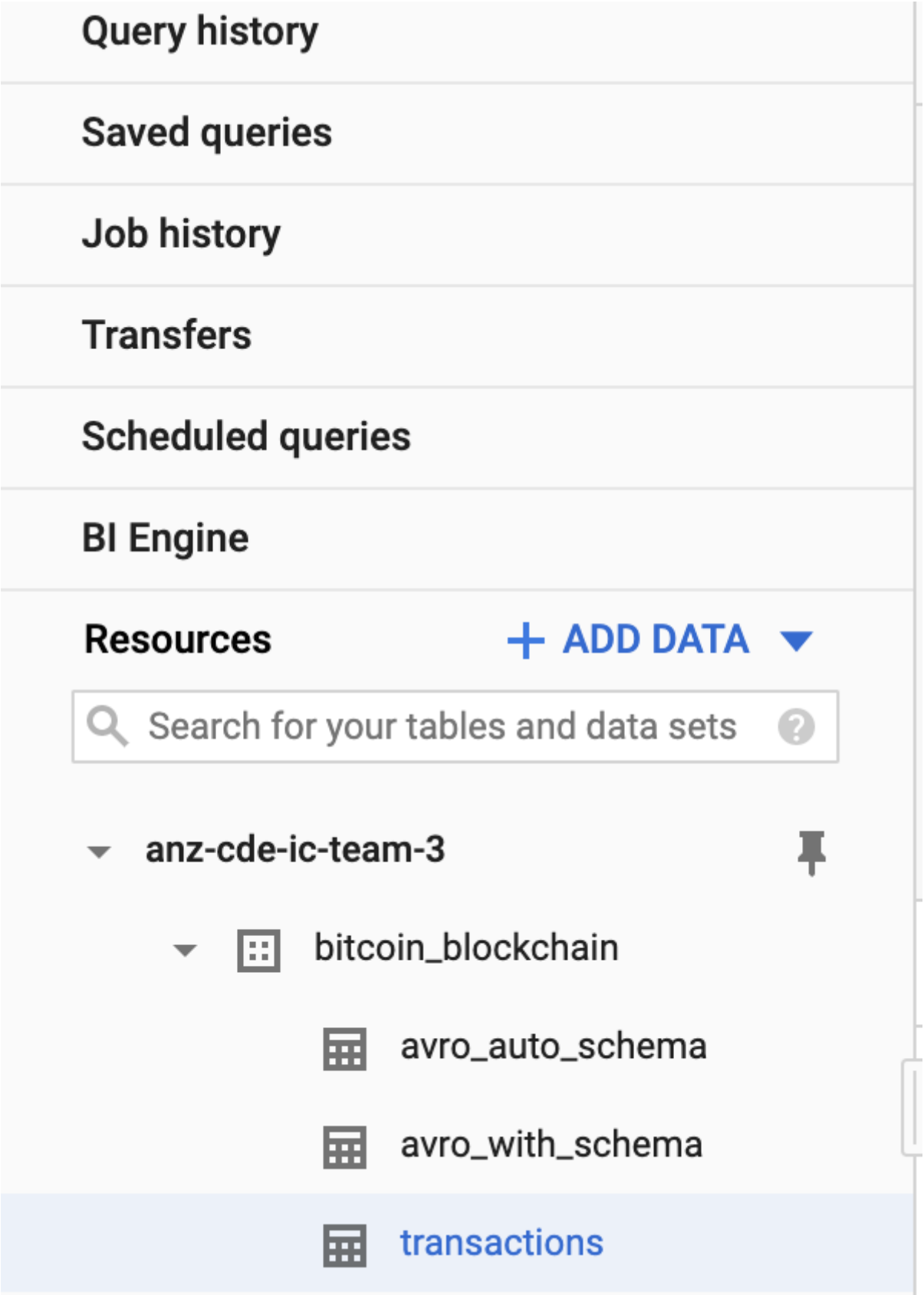
The second task will transform the bigquery table from first task to a cloud storage bucket in avro format

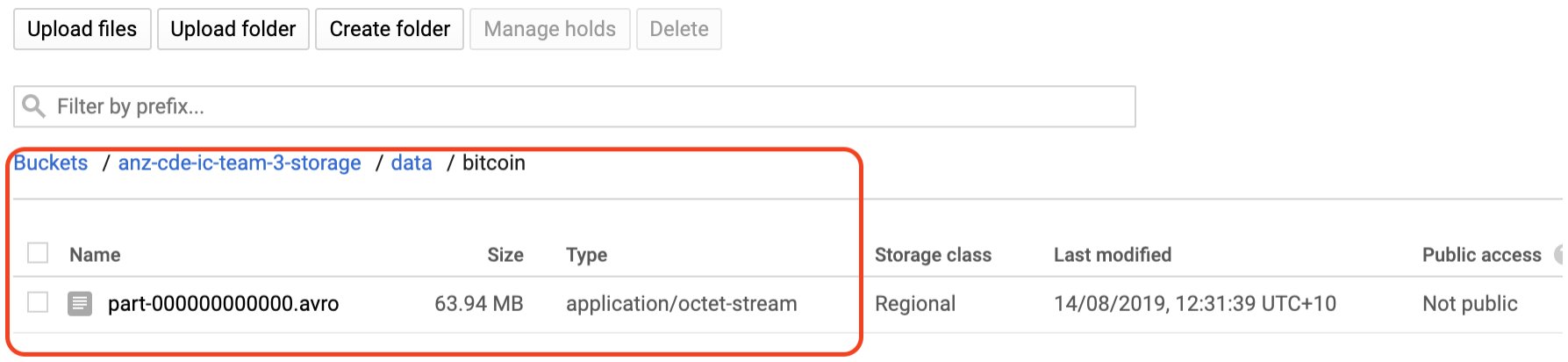
The third task will load the data from second task from cloud storage to another bigquery table using auto schema

The fourth task will load the data from second task from cloud storage to another bigquery table using the schema provided loaded into the cloud storage which is uploaded from the src/schemas folder



After the job is successful we should be able to see two additional bigquery tables loaded from the avro data in cloud storage both with schema and auto schema modes





JOB4

Airflow gcp operator provide different ways to orchestrate with gcp environments. But sometimes an operator cannot meet our requirements such as the gcp dataflow java operator does not load java classes that rely on reflection for injection. Can we run this someother way using docker where all dependencies can be resolved? KubernetesPodOperator can do this. gcp operator extended this to run pod in gke cluster using Operator GKE Pod Operator

Before we run this please build the dataflow job from the parent directory that build the docker image with the dataflow job artifacts.

After the job is successful we should be able to see the pod task in gke and also the dataflow job created by the pod

external-dns	OK	Deployment	1/1	default	anz-cde-ic-team-3-gke
hub	OK	Deployment	1/1	jhub	anz-cde-ic-team-3-gke
kafka-pipeline-d3a091cc	Succeeded	Pod	0/1	default	anz-cde-ic-team-3-gke
nginx-ingress-controller	OK	Deployment	1/1	airflow	anz-cde-ic-team-3-gke
nginx-ingress-default-backend	OK	Deployment	1/1	airflow	anz-cde-ic-team-3-gke

Dataflow | Jobs + CREATE JOB FROM TEMPLATE

Filter jobs

Name	Type	End time	Elapsed time	Start time	Status	SDK versior
kafka2avro-root-0814024857-19cd280d	Streaming	—	40 sec	14 Aug 2019, 12:49:13	Running	2.13.0