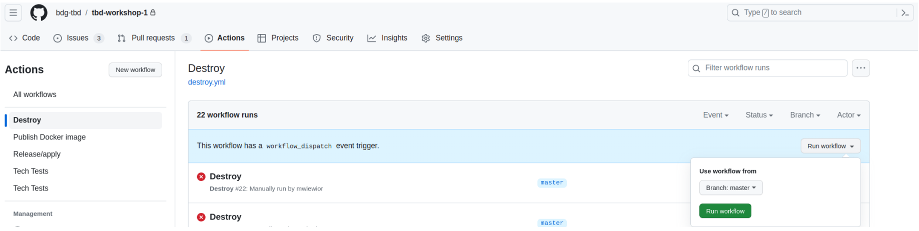
IMPORTANT Please remember to destroy all the resources after each work session. You can recreate infrastructure by creating new PR and merging it to master.



22 workflow runs

This workflow has a `workflow_dispatch` event trigger.

Destroy

Destroy #22: Manually run by mwiewior

Destroy

Figure 1: img.png

0. The goal of this phase is to create infrastructure, perform benchmarking/scalability tests of sample three-tier lakehouse solution and analyze the results using:

- TPC-DI benchmark
- dbt - data transformation tool
- GCP Composer - managed Apache Airflow
- GCP Dataproc - managed Apache Spark
- GCP Vertex AI Workbench - managed JupyterLab

Worth to read: * https://docs.getdbt.com/docs/introduction * https://airflow.apache.org/docs/apache-airflow/stable/index.html * https://spark.apache.org/docs/latest/api/python/index.html * https://medium.com/snowflake/loading-the-tpc-di-benchmark-dataset-into-snowflake-96011e2c26cf * https://www.databricks.com/blog/2023/04/14/how-we-performed-etl-one-billion-records-under-1-delta-live-tables.html

2. Authors:

- Zuzanna Górecka

- Adam Górski

- Michał Oracki

  Zespół nr 7

  Link to forked repo

3. Sync your repo with https://github.com/bdg-tbd/tbd-workshop-1.

4. Provision your infrastructure.

   a) setup Vertex AI Workbench `pyspark` kernel as described in point 8

   b) upload tpc-di-setup.ipynb to the running instance of your Vertex AI Workbench

5. In `tpc-di-setup.ipynb` modify cell under section ***Clone tbd-tpc-di repo***:

a)first, fork https://github.com/mwiewior/tbd-tpc-di.git to your github organization.

b)create new branch (e.g. ‘notebook’) in your fork of tbd-tpc-di and modify profiles.yaml by commenting following lines:

```
#"spark.driver.port": "30000"
#"spark.blockManager.port": "30001"
#"spark.driver.host": "10.11.0.5"  #FIXME: Result of the command (kubectl get node
#"spark.driver.bindAddress": "0.0.0.0"
```

This lines are required to run dbt on airflow but have to be commented while running dbt in notebook.

c)update git clone command to point to ***your fork***.

6. Access Vertex AI Workbench and run cell by cell notebook `tpc-di-setup.ipynb`.

   a) in the first cell of the notebook replace: `%env DATA_BUCKET=tbd-2023z-9910-data` with your data bucket.

   b) in the cell: `%%bash    mkdir -p git && cd git    git clone https://github.com/mwiewior/tbd-tpc-di.git    cd tbd-tpc-di    git pull` replace repo with your fork. Next checkout to ‘notebook’ branch.

   c) after running first cells your fork of `tbd-tpc-di` repository will be cloned into Vertex AI enviroment (see git folder).

   d) take a look on `git/tbd-tpc-di/profiles.yaml`. This file includes Spark parameters that can be changed if you need to increase the number of executors and

```
server_side_parameters:
  "spark.driver.memory": "2g"
  "spark.executor.memory": "4g"
  "spark.executor.instances": "2"
  "spark.hadoop.hive.metastore.warehouse.dir": "hdfs:///user/hive/warehouse/"
```

7. Explore files created by generator and describe them, including format, content, total size.

   There are 3 directories Batch1 Batch2 and Batch3 with each of them containing multiple csv and text files. The biggest ones are DailyMarket.txt and WatchHistory.txt with sizes of 296MB and 134MB. Directioes contain also multiple FINWIRE files varying in size from 900KB to 70KB.‘

8. Analyze tpcdi.py. What happened in the loading stage?

   During loading as no file_name is specified - all are read and saved to the bucket which has been set up as an environement variable. This includes: DATE, DAILY_MARKET, INDUSTRY, PROSPECT, CUSTOMER_MGMT, TAX_RATE, HR, WATCH_HISTORY, TRADE,

TRADE_HISTORY, STATUS_TYPE, TRADE_TYPE, HOLDING_HISTORY, CASH_TRANSACTION, CMP, SEC and FINwIRE files,

9. Using SparkSQL answer: how many table were created in each layer? from pyspark.sql import SparkSession def get_session(): session = SparkSession.builder
   .appName("TBD-TPC-DI-setup")
   .enableHiveSupport()
   .getOrCreate() for db in ['digen', 'bronze', 'silver', 'gold']: session.sql(f"CREATE DATABASE IF NOT EXISTS {db} LOCATION 'hdfs:///user/hive/warehouse/{db}.db' ") session.sql('USE digen') return session

query = """ SHOW TABLES """ session = get_session() result_df = session.sql(query) result_df.show()

– +———+—————-+———+ |namespace| tableName|isTemporary| +———+—————-+———+ | digen|cash_transaction| false| | digen| cmp| false| | digen| customer_mgmt| false| | digen| daily_market| false| | digen| date| false| | digen| fin| false| | digen| holding_history| false| | digen| hr| false| | digen| industry| false| | digen| prospect| false| | digen| sec| false| | digen| status_type| false| | digen| tax_rate| false| | digen| trade| false| | digen| trade_history| false| | digen| trade_type| false| | digen| watch_history| false| +———+—————-+———+

There has been 17 tables created.

10. Add some 3 more dbt tests and explain what you are testing. ***Add new tests to your repository.***

Testing if transacations are not for a negative value: "'sql select ct_ca_id from {{ source('brokerage', 'cash_transaction') }} WHERE ct_amt < 0

~~~
Testing if there are statuses with empty status_name:
```sql
select ct_ca_id
from {{ source('reference', 'status_type') }}
WHERE ST_NAME IS NULL
~~~

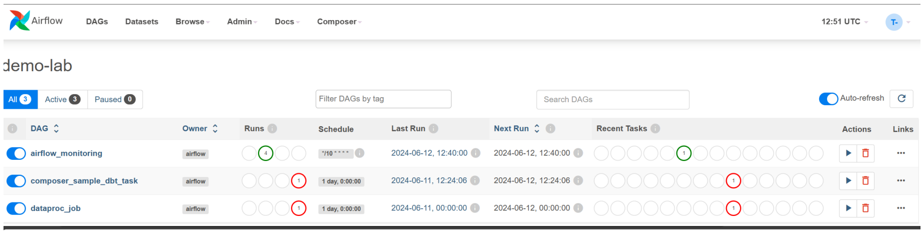
Testing if there are no employees who are their own managers: `sql select * from {{ source('hr', 'hr') }} WHERE employee_id = manager_id`

11. In main.tf update

```
dbt_git_repo            = "https://github.com/mwiewior/tbd-tpc-di.git"
dbt_git_repo_branch     = "main"
```

so dbt_git_repo points to your fork of tbd-tpc-di.

12. Redeploy infrastructure and check if the DAG finished with no errors:



demo-lab

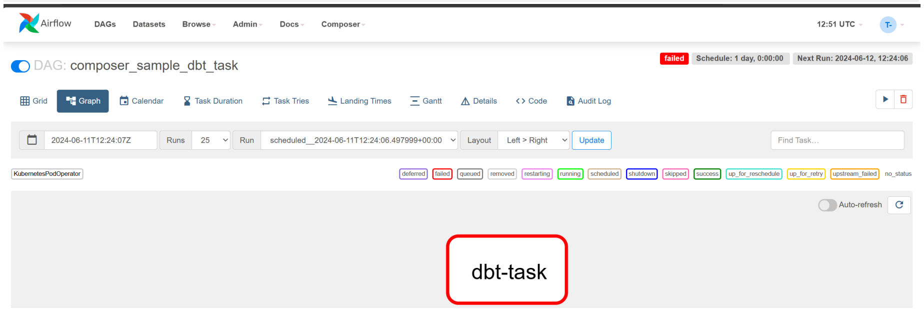
All 3 | Active 3 | Paused 0

Filter DAGs by tag

Search DAGs

Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
airflow_monitoring	airflow	4	*/10 * * * *	2024-06-12, 12:40:00	2024-06-12, 12:40:00	1	▶ 🗑	•••
composer_sample_dbt_task	airflow	1	1 day, 0:00:00	2024-06-11, 12:24:06	2024-06-12, 12:24:06	1	▶ 🗑	•••
dataproc_job	airflow	1	1 day, 0:00:00	2024-06-11, 00:00:00	2024-06-12, 00:00:00	1	▶ 🗑	•••



DAG: composer_sample_dbt_task

failed | Schedule: 1 day, 0:00:00 | Next Run: 2024-06-12, 12:24:06

2024-06-11T12:24:07Z | Runs 25 | Run scheduled__2024-06-11T12:24:06.497999+00:00 | Layout Left > Right

Analyzing the logs (including in Airflow), we concluded that there is a problem with memory limits. Despite many attempts, we were unable to resolve it. We tried changing the machine type and memory limits.

However, we were the first team to solve the problem that occurred during release at the beginning of task 2a. Thanks to this, other teams could solve the problem using our recommendations. After analyzing the logs and tracking CPU and SSD usage, we noticed that the problem was with qoutas. The problem was that the cluster tried to autoscale but couldn't due to upper limits, so after a few failed attempts it deleted itself and healthchecks had no way to reach the required pods.