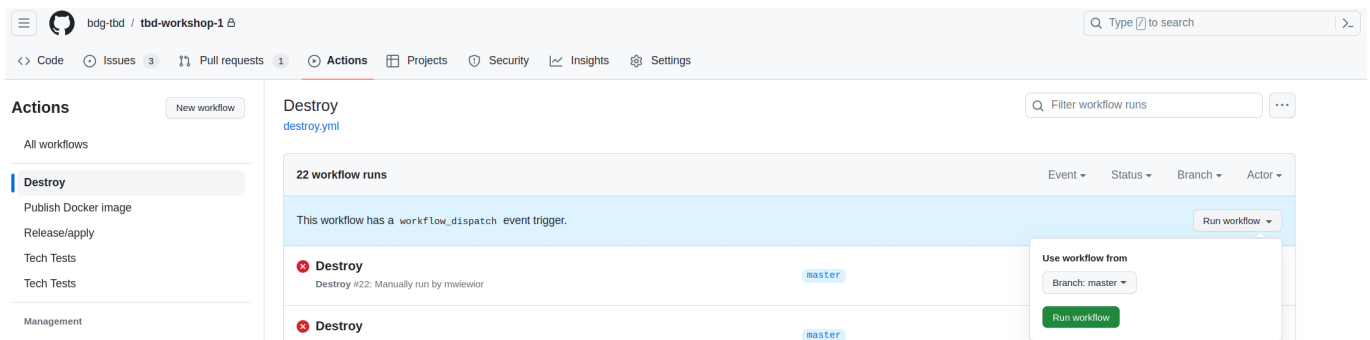


IMPORTANT 🚨 🚨 🚨 Please remember to destroy all the resources after each work session. You can recreate infrastructure by creating new PR and merging it to master.



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:

- Aleksandra Gryzik
- Gabriel Skowron-Rodriguez
- Jakub Rozkosz

Link to forked repo: <https://github.com/spacerunner00/tbd-workshop-1>

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] (<https://github.com/bdg-tbd/tbd-workshop-1/tree/v1.0.32#project-setup>)

b) upload [tpc-di-setup.ipynb](<https://github.com/bdg-tbd/tbd-workshop-1/blob/v1.0.36/notebooks/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 nodes -o json | jq -r '.items[0].status.addresses[0].address')
#"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 environment (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.

Wygenerowany zestaw danych to TPC-DI (Transaction Processing Performance Council - Data Integration). TPC-DI jest standardem do testowania wydajności baz danych przy użyciu generowanych danych.

Plik *Batch1_audit.csv* zawiera dwa wiersze informujące o przedziale czasowym, z którego pochodzą dane - dane z przeprowadzanego audytu finansowego. Pierwszy wiersz przedstawia datę początkową (rok 1950), a drugi – datę końcową (rok 2017). Dane te obejmują zatem rozległy okres historyczny. Analogiczne informacje znajdują się w plikach *Batch2_audit.csv* oraz *Batch3_audit.csv*, lecz dotyczą one bardziej aktualnych i krótszych przedziałów czasowych: *Batch2_audit.csv* zawiera dane z dnia 2017-07-08 (odpowiadające dniowi poprzedniemu), *Batch3_audit.csv* odnosi się do dnia 2017-07-09 (reprezentując dzień bieżący).

W folderach *Batch1*, *Batch2* oraz *Batch3* znajdują się dane dotyczą różnych aspektów działalności audytowanych przedsiębiorstw, obejmujące takie obszary jak transakcje gotówkowe (Cash transaction), zarządzanie klientami (Customer management) czy codzienny rynek (Daily market).

Do każdej dziedziny danych wygenerowane zostały dwa pliki - jeden w formacie csv, drugi w txt (przykładowo *TradeHistory_audit.csv* oraz *TradeHistory.txt*). Plik CSV zawiera podsumowanie i agregację danych, przedstawiając różne atrybuty związane z daną dziedziną. Znajdują się tam takie informacje jak liczba rekordów, operacji czy zdarzeń w określonym okresie czasu. Może zawierać również dane o stanie różnych elementów, takich jak liczba utworzonych kont, zamkniętych transakcji, zaktualizowanych informacji, itp. Plik TXT zawiera już szczegółowe dane o każdym rejestrowanym zdarzeniu, często w formie zapisów transakcji lub innych jednostkowych zdarzeń. Zawiera informacje takie jak daty, wartości operacji, identyfikatory, a także inne szczegóły związane z każdym zdarzeniem. W przeciwieństwie do pliku CSV, plik tekstowy może być używany do bardziej zaawansowanej analizy, np. w przypadku potrzeby prześledzenia poszczególnych operacji w danej dziedzinie.

Podsumowanie liczby rekordów:

```
jupyter@0115ef734e8f:/tmp/tpc-di$ cat digen_report.txt
TPC-DI Data Generation Report
=====

Start Time: 2025-01-06T15:42:51+0000
End Time: 2025-01-06T15:44:42+0000
DIGen Version: 1.1.0
Scale Factor: 10
AuditTotalRecordsSummaryWriter - TotalRecords for Batch1: 15980433
AuditTotalRecordsSummaryWriter - TotalRecords for Batch2: 67451
AuditTotalRecordsSummaryWriter - TotalRecords for Batch3: 67381
AuditTotalRecordsSummaryWriter - TotalRecords all Batches: 16115265 151524.77 records/second

Command options used: -sf 10 -o /tmp/tpc-di
PDGF Version: PDGF v2.5_#1343_b4177
Java version: Amazon.com Inc. 1.8.0_392
jupyter@0115ef734e8f:/tmp/tpc-di$
```

```
jupyter@0115ef734e8f:/tmp$ du -sh tpc-di/
962M    tpc-di/
```

Łączny rozmiar danych: `jupyter@0115ef734e8f:/tmp$`

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

W kodzie *tpcdi.py*, który uruchamiany był w loading stage, zdefiniowana jest funkcja *upload_files*, która przesyła pliki do Google Cloud Storage lub innej lokalizacji wskazanej przez *stagepath*. Funkcja ta przeszukuje katalog *output_directory* w poszukiwaniu plików pasujących do wzorca, a następnie w zależności od typu pliku

ustawia odpowiedni delimiter (np. przecinek dla CSV). Następnie pliki są przesyłane do chmury, a ich nazwa jest używana jako nazwa bloba w GCS. Jest ona wywoływana przez funkcję `_load_csv`.

Kolejnym krokiem jest funkcja `load_csv`, która odpowiedzialna jest za załadowanie plików CSV z GCS do DataFrame w Spark. Na początku ustalana jest ścieżka do wgranego pliku, a potem wywoływana jest funkcja `upload_files`, która załadowuje pliki do GCS. Następnie dane z GCS są wczytywane przy użyciu Spark, z uwzględnieniem podanego schematu, który opisuje strukturę danych w pliku. Po załadowaniu danych, wynikowy DataFrame jest przekazywany do funkcji `save_df`.

Funkcja `save_df` sprawdza, czy zmienna `show` jest ustawiona na `True`. Jeśli tak, dane w DataFrame są po prostu wyświetlane na ekranie za pomocą `df.show()`. Jeśli `show` jest ustawione na `False`, dane są zapisywane do systemu w formacie `parquet`, a dodatkowo tworzona jest tabela o nazwie określonej przez `table_name`. Tabela jest zapisywana w trybie nadpisywania, co oznacza, że jeżeli tabela o tej samej nazwie już istnieje, zostanie zastąpiona.

Pliki zostały wgrane do naszego bucketa w GCS:

TBD tbd-2024zz-305978 project

Search (/) for resources, docs, products, and more

Search

◆

🔍

🔔

📌

←

Bucket details

🔄 REFRESH

🔗 GO TO PATH

📁 tbd-2024zz-305978-data

Location

europa-west1 (Belgium)

Storage class

Standard

Public access

Not public

Protection

Soft Delete, Object versioning

OBJECTS

CONFIGURATIONPERMISSIONSPROTECTIONLIFECYCLEOBSERVABILITYINVENTORY REPORTSOPERATIONS

Folder browser

🔍

📁 tbd-2024zz-305978-data

📁 data/

📁 tpc-di/

Buckets > tbd-2024zz-305978-data > tpc-di

CREATE FOLDEROTHER SERVICESUPLOADTRANSFER DATA

Filter by name prefix onlyFilter objects and folders

Show Live object

<input type="checkbox"/>	Name	Size	Type	Created	Storage class
<input type="checkbox"/>	CashTransaction.txt	104.4 MB	text/plain	Jan 6, 2025, 5:04:15 PM	Standard
<input type="checkbox"/>	CustomerMgmt.xml	29.7 MB	application/xml	Jan 6, 2025, 5:02:23 PM	Standard
<input type="checkbox"/>	DailyMarket.txt	295.2 MB	text/plain	Jan 6, 2025, 5:01:33 PM	Standard
<input type="checkbox"/>	Date.txt	3.3 MB	text/plain	Jan 6, 2025, 5:01:19 PM	Standard
<input type="checkbox"/>	FINWIRE1967Q1	80.9 KB	application/octet-stream	Jan 6, 2025, 5:04:32 PM	Standard
<input type="checkbox"/>	FINWIRE1967Q2	76.7 KB	application/octet-stream	Jan 6, 2025, 5:04:39 PM	Standard
<input type="checkbox"/>	FINWIRE1967Q3	57.6 KB	application/octet-stream	Jan 6, 2025, 5:04:22 PM	Standard
<input type="checkbox"/>	FINWIRE1967Q4	62.4 KB	application/octet-stream	Jan 6, 2025, 5:04:23 PM	Standard
<input type="checkbox"/>	FINWIRE1968Q1	67 KB	application/octet-stream	Jan 6, 2025, 5:04:23 PM	Standard
<input type="checkbox"/>	FINWIRE1968Q2	71.2 KB	application/octet-stream	Jan 6, 2025, 5:04:40 PM	Standard
<input type="checkbox"/>	FINWIRE1968Q3	76.5 KB	application/octet-stream	Jan 6, 2025, 5:04:44 PM	Standard
<input type="checkbox"/>	FINWIRE1968Q4	81.1 KB	application/octet-stream	Jan 6, 2025, 5:04:23 PM	Standard
<input type="checkbox"/>	FINWIRE1969Q1	85.4 KB	application/octet-stream	Jan 6, 2025, 5:04:29 PM	Standard
<input type="checkbox"/>	FINWIRE1969Q2	89.6 KB	application/octet-stream	Jan 6, 2025, 5:04:26 PM	Standard
<input type="checkbox"/>	FINWIRE1969Q3	94.3 KB	application/octet-stream	Jan 6, 2025, 5:04:23 PM	Standard

9. Using SparkSQL answer: how many table were created in each layer?

The first screenshot shows the output of a Spark SQL query: `spark.sql("show databases").show()`. The output lists several databases: `namespace`, `bronze`, `default`, `demo_bronze`, `demo_gold`, `demo_silver`, `digen`, `gold`, and `silver`.

The second screenshot shows a more complex Spark SQL query: `spark.sql("SHOW DATABASES").collect()`. The output displays the total number of tables for each namespace and the total count across all namespaces. The results are as follows:

Namespace	Łączna liczba tabel w namespace
bronze	0
default	0
demo_bronze	17
demo_gold	12
demo_silver	14
digen	17
gold	0
silver	0
Całkowita liczba tabel we wszystkich namespace	60

10. Add some 3 more [dbt tests](#) and explain what you are testing. **Add new tests to your repository.**

Link do folderu z testami: [test](#)

Test 1. Sprawdza, czy wartości w kolumnie `executed_by` nie są NULL ani puste.

```
SELECT
    sk_trade_id,
    executed_by
FROM {{ ref('fact_trade') }}
WHERE executed_by IS NULL OR executed_by = ''
```

Test 2. Sprawdza, czy wartości w kolumnie `sk_trade_id` nie są NULL.

```
SELECT
    sk_trade_id
FROM {{ ref('fact_trade') }}
WHERE sk_trade_id IS NULL
```

Test 3. Sprawdza, czy suma wartości w kolumnie `amount` dla każdego `order_id` nie jest ujemna.

```
select
  order_id,
  sum(amount) as total_amount
from {{ ref('fct_payments') }}
group by 1
having total_amount < 0
```

Test 4. Sprawdza, czy wartości w kolumnie `trade_price` dla każdego `sk_trade_id` nie są ujemne.

```
SELECT
  sk_trade_id,
  trade_price
FROM {{ ref('fact_trade') }}
WHERE trade_price < 0
```

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`.

Zmieniliśmy wartość parametru żeby wskazywała na naszego forka:

```
dbt_git_repo      = "https://github.com/spacerunner00/tbd-tpc-di.git"
dbt_git_repo_branch = "main"
```

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

Composer

Environment details

OPEN AIRFLOW UI

OPEN DAGS FOLDER

SAVE SNAPSHOT

LOAD SNAPSHOT

REFRESH

DELETE

LEARN

demo-lab

This environment is running

MONITORING

LOGS

DAGS

ENVIRONMENT CONFIGURATION

AIRFLOW CONFIGURATION OVERRIDES

ENVIRONMENT VARIABLES

LABELS

PYPI PACKAGES

Filter

Filter DAGs

1 hour

6 hours

12 hours

1 day

2 days

4 days

7 days

14 days

30 days

DAG id	State	Description	Schedule interval	Last completed run	Active runs	Successful runs (1h)	Failed runs (1h)
airflow_monitoring	Active	liveness monitoring dag	* / 10 * * * *	5 minutes ago	0	6	0
composer_sample_dbt_task	Active		1 day	44 minutes ago	0	1	0
dataproc_job	Active		1 day	44 minutes ago	0	1	0

SEVERITY

TIMESTAMP

SUMMARY

>

2025-02-01 13:50:23.121 UTC

[base] WARNING: Illegal reflective access by org.apache.hadoop.shaded.org.xbill.DNS.ResolverConfig (file:/usr/local/lib/python3.10/site-packages/pyspark/jars/hadoop-client-run...

>

2025-02-01 13:50:23.121 UTC

[base] WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.shaded.org.xbill.DNS.ResolverConfig

>

2025-02-01 13:50:23.121 UTC

[base] WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations

>

2025-02-01 13:50:23.145 UTC

[base] WARNING: All illegal access operations will be denied in a future release

>

2025-02-01 13:50:27.121 UTC

[base] 25/02/01 13:50:23 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

>

2025-02-01 13:50:27.325 UTC

[base] 25/02/01 13:50:27 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.

>

2025-02-01 13:51:30.588 UTC

[base] 25/02/01 13:50:27 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.

>

2025-02-01 13:51:30.589 UTC

[base] [0m13:51:30 Connection test: [] [32mOK connection ok [0m] 2025-02-01T13:51:30.578718343Z

>

2025-02-01 13:51:32.596 UTC

[base] [0m13:51:30 [32mAll checks passed! [0m

>

2025-02-01 13:51:32.928 UTC

Pod dbt-task-go7hc1sl has phase Running

>

2025-02-01 13:51:35.167 UTC

Deleting pod: dbt-task-go7hc1sl

>

2025-02-01 13:51:35.595 UTC

Marking task as SUCCESS. dag_id=composer_sample_dbt_task, task_id=dbt-task, execution_date=20250131T134745, start_date=20250201T134923, end_date=20250201T135135

>

2025-02-01 13:51:36.304 UTC

Task exited with return code 0

>

2025-02-01 13:51:36.464 UTC

0 downstream tasks scheduled from follow-on schedule check

Composer

Environment details

OPEN AIRFLOW UI

OPEN DAGS FOLDER

SAVE SNAPSHOT

LOAD SNAPSHOT

REFRESH

DELETE

LEARN

Airflow

DAGs

Datasets

Browse

Admin

Docs

Composer

14:36 UTC

A-

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
<div>airflow_monitoring</div>	airflow	<div>6</div>	* / 10 * * * *	2025-02-01, 14:20:00	2025-02-01, 14:20:00	<div>1</div>	<div></div> <div></div>	...
<div>composer_sample_dbt_task</div>	airflow	<div>1</div>	1 day, 0:00:00	2025-01-31, 13:47:45	2025-02-01, 13:47:45	<div>1</div>	<div></div> <div></div>	...
<div>dataproc_job</div>	airflow	<div>1</div>	1 day, 0:00:00	2025-01-31, 00:00:00	2025-02-01, 00:00:00	<div>1</div>	<div></div> <div></div>	...

1

Showing 1-3 of 3 DAGs

