Build Robust Automated Data Modeling

A Beginner's Guide to Collaborative Data Modeling with DBT and Airflow

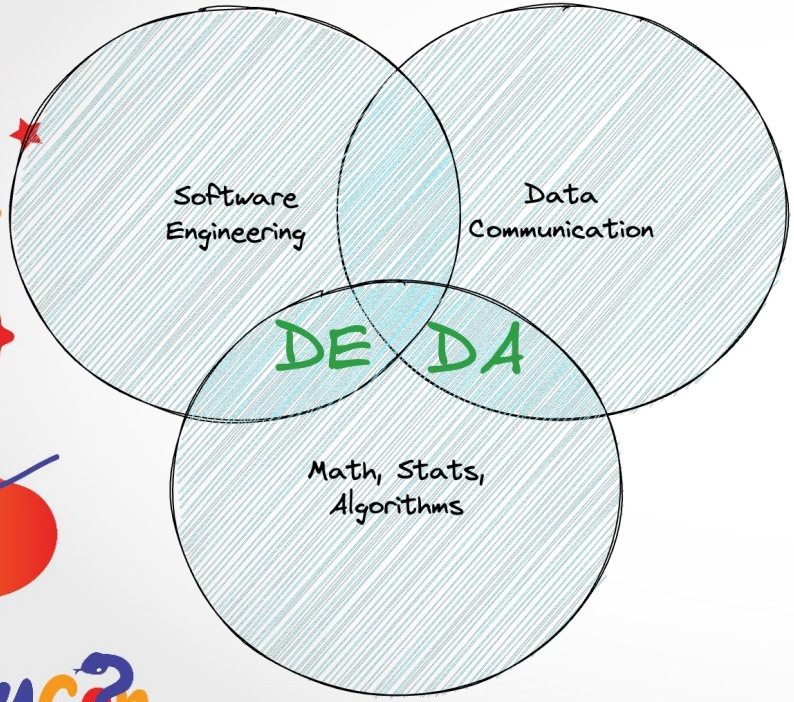
Ahmad Shohibus Sulthoni

Sr. Data Analyst @Pintu
Founder & Mod @Pelajar Data

https://linktr.ee/as_sulthoni

PyCon ID 2023
Bandung, Nov 18th - 19th

Analyst and Engineer Role



Data Analysts are strong in :

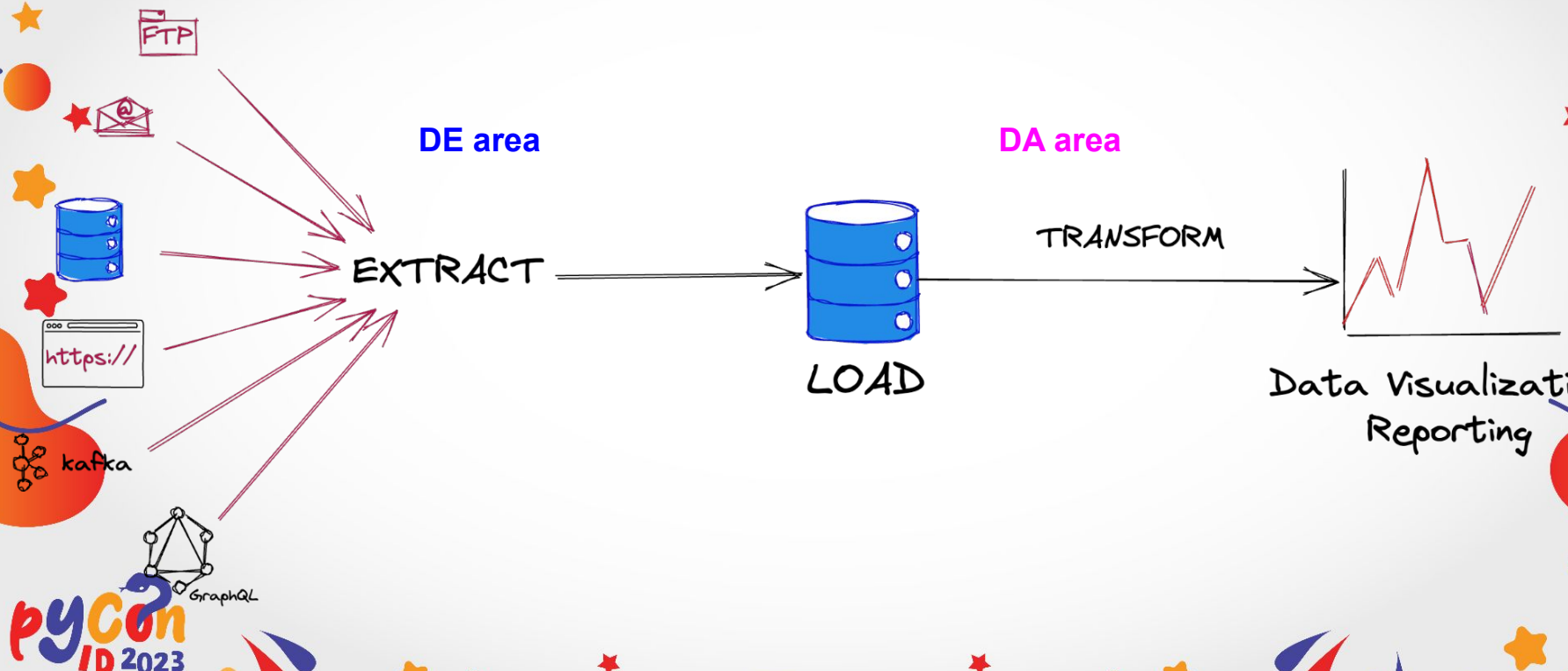
- Business Subject
- Data Visualization and Presentation
- Logic/formula on creating metrics and KPI

Data Engineers are strong in :

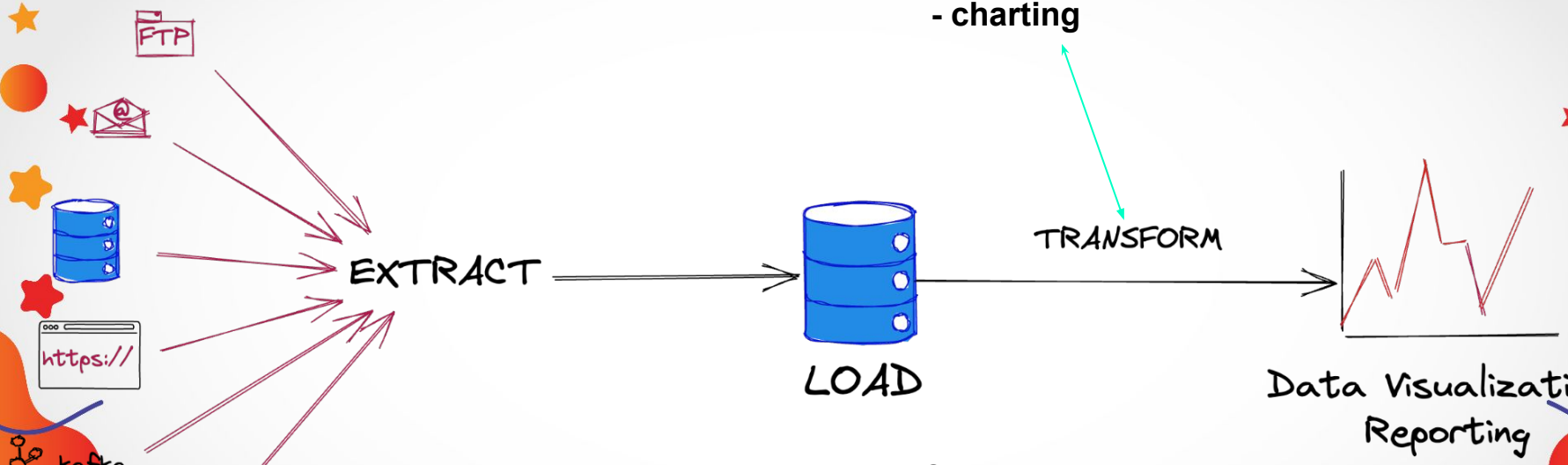
- Software Engineer best practices
- Automation
- Computation optimization

ELT (Extract, Transform, Load) Design

Before


FTP
https://
kafka
GraphQL
DE area
EXTRACT
LOAD
DA area
TRANSFORM
Data Visualization/
Reporting

Before

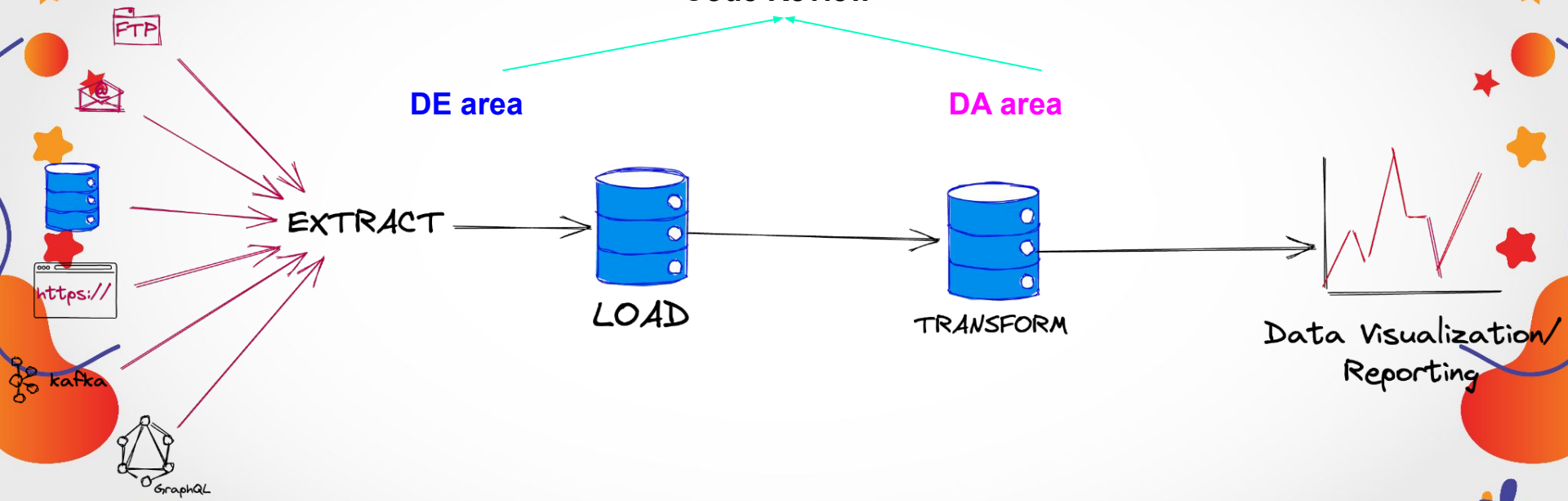


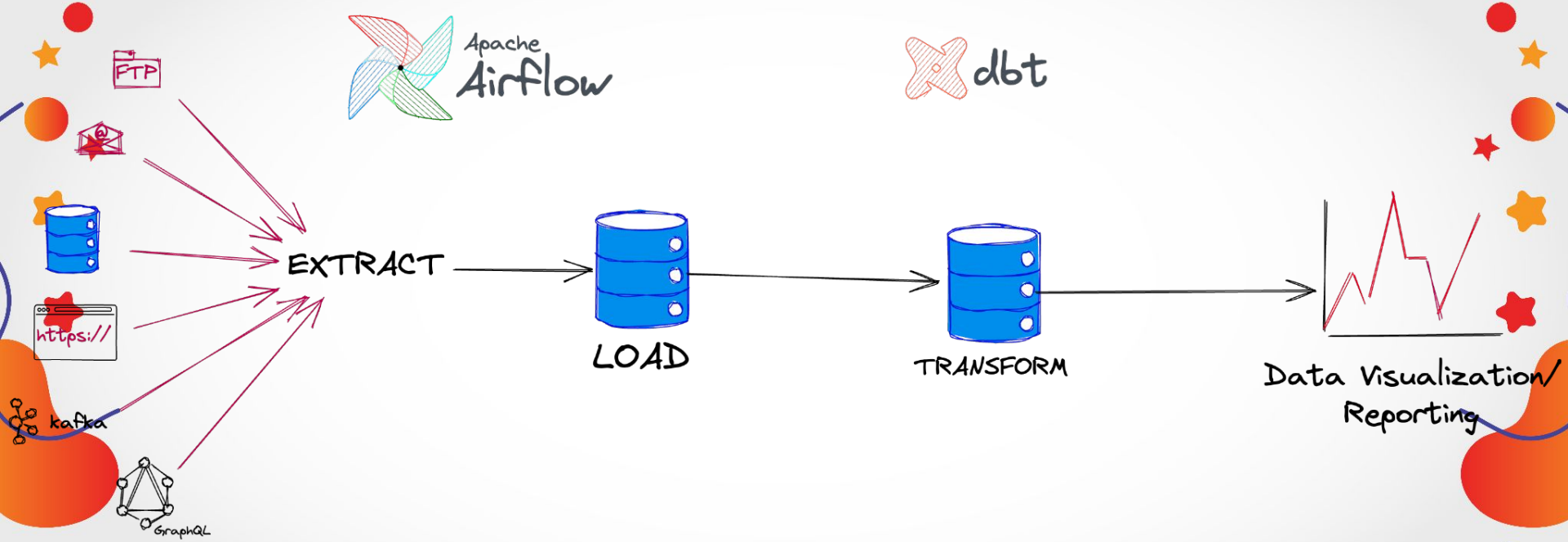
Contains:
- business logic
- charting

Common Problems:
- Metrics Silos
- Unmanageable query and dashboard
- Data Quality is poor

After

- Centralized logic
- QA on Data
- Code Review




Apache
Airflow
dbt
FTP
https://
kafka
GraphQL
EXTRACT
LOAD
TRANSFORM
Data Visualization/
Reporting

Airflow

Apache Airflow™ is an open-source platform for **developing, scheduling, and monitoring batch-oriented** workflows. Airflow's extensible Python framework enables you to build workflows connecting with virtually any technology. A web interface helps manage the state of your workflows. Airflow is deployable in many ways, varying from a single process on your laptop to a distributed setup to support even the biggest workflows.

DBT

dbt is a transformation workflow that helps you get more work done while producing higher quality results. You can use dbt to **modularize and centralize your analytics code**, while also providing your data team with guardrails typically found in software engineering workflows. Collaborate on data models, version them, and test and document your queries before safely deploying them to production, with monitoring and visibility.

Repo Structure

https://github.com/assulthoni/airflow-dbt-integration

```
.
├── Dockerfile
├── LICENSE
├── README.md
├── config
├── dags
│   ├── dbt
│   │   └── simple_data_modeling
│   │       ├── README.md
│   │       ├── analyses
│   │       ├── dbt_packages
│   │       ├── dbt_project.yml
│   │       ├── macros
│   │       ├── models
│   │       │   ├── continent_metrics_model.sql
│   │       │   ├── example
│   │       │   ├── gov_continent_metrics_model.sql
│   │       │   ├── gov_metrics.sql
│   │       │   ├── schema.yml
│   │       │   └── source.yml
│   │       ├── profiles.yml
│   │       ├── seeds
│   │       ├── snapshots
│   │       └── tests
│   ├── dbt_dag_advance.py
│   └── dbt_dag_basic.py
├── db-init-scripts
│   └── world.sql
├── docker-compose.yaml
├── plugins
└── requirements.txt
```

DBT Model Example

```sql
{{ config(materialized='table') }}

WITH country_data AS (
    SELECT
        governmentform,
        continent,
        code,
        surfacearea,
        population
    FROM {{ source('public', 'country') }}
),

final AS (
    SELECT
        governmentform,
        continent,
        COUNT(code) AS country_count,
        SUM(surfacearea) AS total_area,
        SUM(population) AS total_population
    FROM country_data
    GROUP BY governmentform, continent
)

SELECT *
FROM final
```

```sql
{{ config(materialized='table') }}

WITH country_data AS (
    SELECT
        governmentform,
        continent,
        country_count,
        total_area,
        total_population
    FROM {{ ref('gov_continent_metrics_model') }}
),

final AS (
    SELECT
        governmentform,
        COUNT(DISTINCT continent) AS ctd_continent,
        SUM(total_population) AS total_population
    FROM country_data
    GROUP BY governmentform
)

SELECT *
FROM final
```

Orchestrating DBT

```python
import os
from datetime import datetime
from airflow import DAG
from airflow.operators.bash_operator import BashOperator

CURRENT_DIR = os.getcwd()
DBT_DIR = CURRENT_DIR + '/dags/dbt/simple_data_modeling'

dag = DAG(
    "dbt_dag_basic",
    start_date=datetime(2020, 12, 23),
    description="A sample Airflow DAG to invoke dbt runs using a BashOperator",
    schedule_interval='@daily',
    catchup=False,
)

task_run = BashOperator(
    dag=dag,
    task_id="dbt_run",
    bash_command=f"cd {DBT_DIR} && dbt run --profiles-dir ."
)

task_test = BashOperator(
    dag=dag,
    task_id="dbt_test",
    bash_command=f"cd {DBT_DIR} && dbt run --profiles-dir ."
)

task_run >> task_test
```

Orchestrating DBT (Advance)

```python
import os
import json
from datetime import datetime
from airflow import DAG
from airflow.operators.bash_operator import BashOperator

CURRENT_DIR = os.getcwd()
DBT_DIR = CURRENT_DIR + '/dags/dbt/simple_data_modeling'

dag = DAG(
    "dbt_dag_advance",
    start_date=datetime(2020, 12, 23),
    description="A sample Airflow DAG to invoke dbt runs using a BashOperator",
    schedule_interval='@daily',
    catchup=False,
)


def load_manifest():
    local_filepath = f"{DBT_DIR}/target/manifest.json"
    with open(local_filepath) as f:
        data = json.load(f)
        return data
```

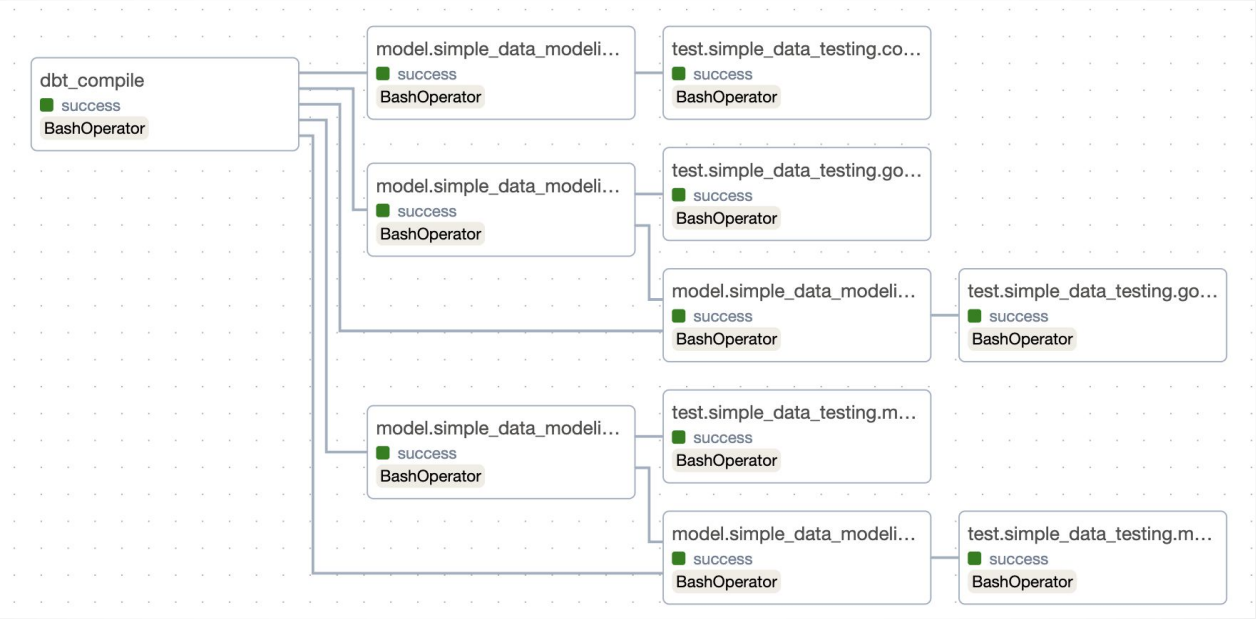
Orchestrating DBT (Advance)

```python
def make_dbt_task(node, dbt_verb):
    model = node.split(".")[-1]
    if dbt_verb == "run":
        dbt_task = BashOperator(
            dag=dag,
            task_id=node,
            bash_command=(
                f"cd {DBT_DIR} && "
                f"dbt {dbt_verb} --models {model} "
                f"--profiles-dir ."
            ),
        )
    elif dbt_verb == "test":
        node_test = node.replace("model", "test")
        dbt_task = BashOperator(
            dag=dag,
            task_id=node_test,
            bash_command=(
                f"cd {DBT_DIR} && "
                f"dbt {dbt_verb} --models {model} "
                f"--profiles-dir ."
            ),
        )
    return dbt_task
```

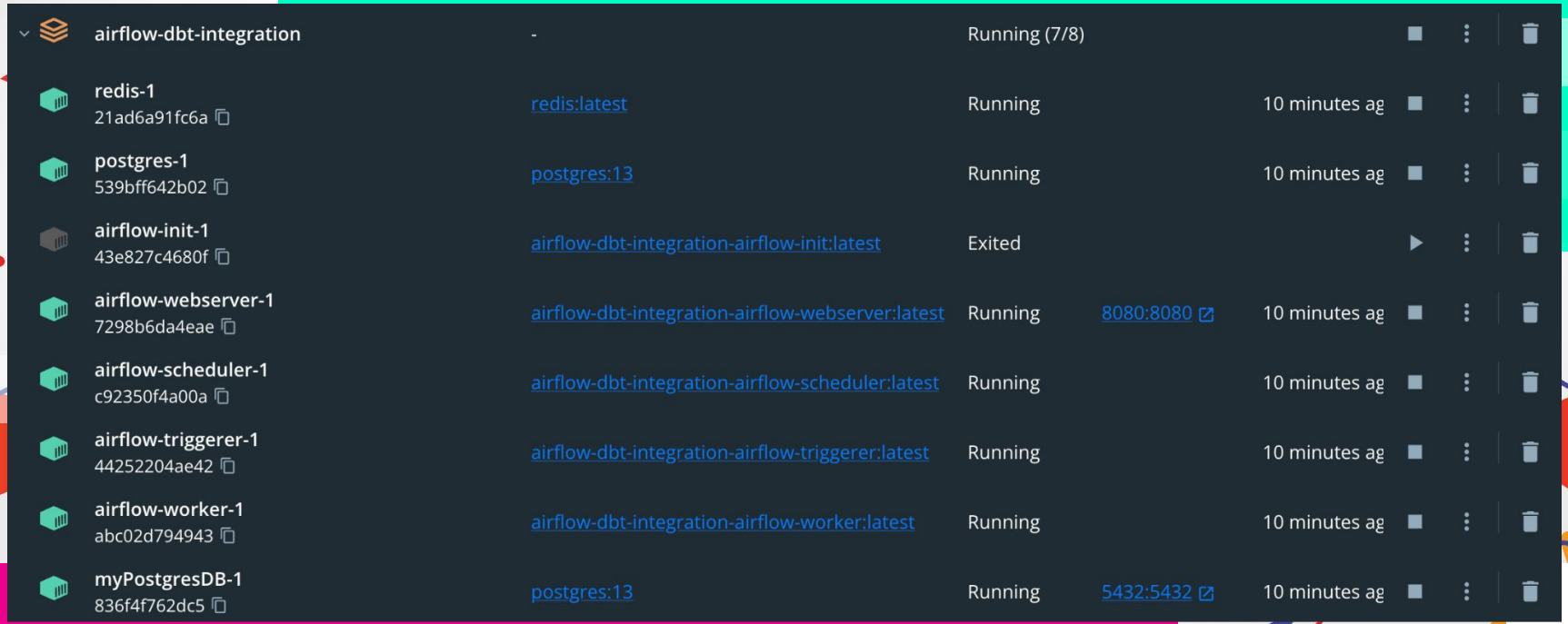
Orchestrating DBT (Advance)

```
dbt_compile = BashOperator(
    dag=dag,
    task_id='dbt_compile',
    bash_command=(
        f"cd {DBT_DIR} && "
        f"dbt compile "
        f"--profiles-dir ."
    ),
)

data = load_manifest()
dbt_tasks = {}

for node in data["nodes"].keys():
    if node.split(".")[0] == "model":
        node_test = node.replace("model", "test")
        dbt_tasks[node] = make_dbt_task(node, "run")
        dbt_tasks[node_test] = make_dbt_task(node, "test")

for node in data["nodes"].keys():
    if node.split(".")[0] == "model":
        node_test = node.replace("model", "test")
        dbt_compile >> dbt_tasks[node] >> dbt_tasks[node_test]
        for upstream_node in data["nodes"][node]["depends_on"]["nodes"]:
            upstream_node_type = upstream_node.split(".")[0]
            if upstream_node_type == "model":
                dbt_compile >> dbt_tasks[upstream_node] >> dbt_tasks[node]
```

DAG Graph on DBT



Run Example


airflow-dbt-integration - Running (7/8)
redis-1 21ad6a91fc6a redis:latest Running 10 minutes ag
postgres-1 539bff642b02 postgres:13 Running 10 minutes ag
airflow-init-1 43e827c4680f airflow-dbt-integration-airflow-init:latest Exited
airflow-webserver-1 7298b6da4eae airflow-dbt-integration-airflow-webserver:latest Running 8080:8080 10 minutes ag
airflow-scheduler-1 c92350f4a00a airflow-dbt-integration-airflow-scheduler:latest Running 10 minutes ag
airflow-triggerer-1 44252204ae42 airflow-dbt-integration-airflow-triggerer:latest Running 10 minutes ag
airflow-worker-1 abc02d794943 airflow-dbt-integration-airflow-worker:latest Running 10 minutes ag
myPostgresDB-1 836f4f762dc5 postgres:13 Running 5432:5432 10 minutes ag

Reference

- https://aws.amazon.com/what-is/data-modeling/
- https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html
- https://docs.astronomer.io/learn/airflow-dbt

QnA

Feel free to ask/give any question or feedback