



FINAL PROJECT DATA ENGINEER

END TO END PIPELINE – DE BATCH 5

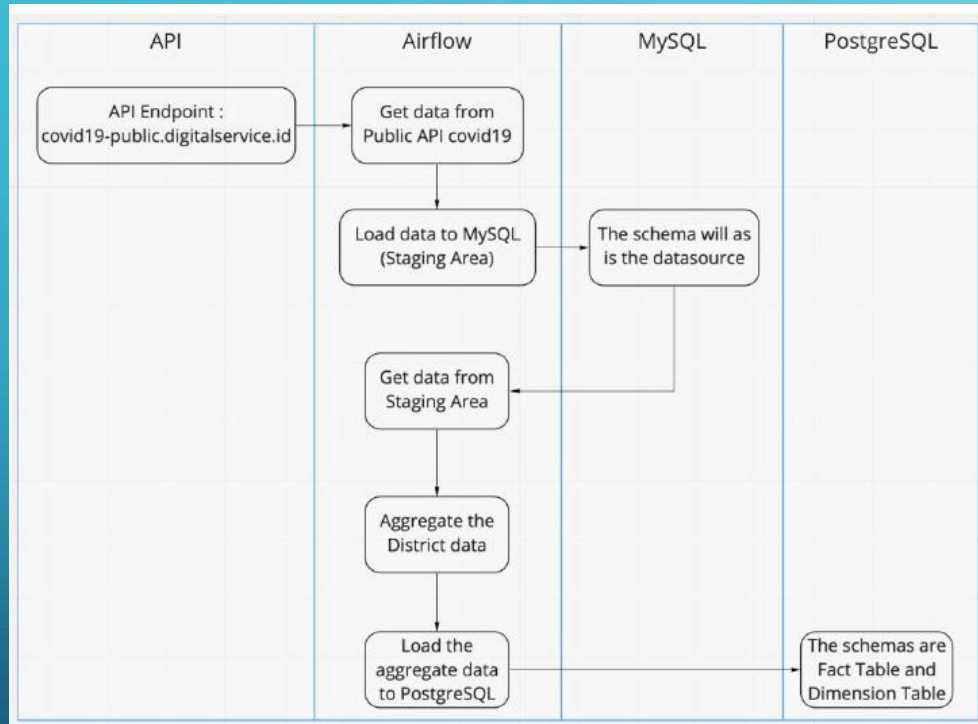
ARDHANI RAHMADIANTO

INDEX

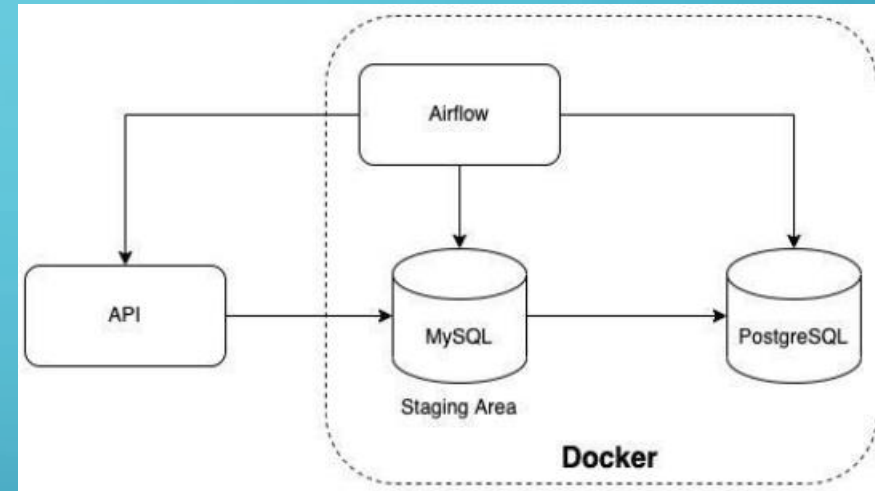
- Project Briefing
- Setup Environment (Airflow, Postgres, MySQL using Docker Compose)
- Setting Airflow Configuration
- DAG File Explanation (Python, SQL, etc.)
- Running Airflow DAG
- ERD & Table Result

PROJECT BRIEFING

Workflow



ETL Architecture



PROJECT BRIEFING

API

```
{
  "status_code": 200,
  "data": {
    "metadata": {
      "last_update": null
    },
    "content": [
      {
        "tanggal": "2020-08-05",
        "kode_prov": "32",
        "nama_prov": "Jawa Barat",
        "kode_kab": "3204",
        "nama_kab": "Kabupaten Bandung",
        "SUSPECT": 2210,
        "CLOSECONTACT": 274,
        "PROBABLE": 26,
        "suspect_diisolasi": 31,
        "suspect_discarded": 2179,
        "closecontact_dikarantina": 0,
        "closecontact_discarded": 274,
        "probable_diisolasi": 0,
        "probable_discarded": 0,
        "CONFIRMATION": 0,
        "confirmation_sembuh": 0,
        "confirmation_meninggal": 0,
        "suspect_meninggal": 0,
        "closecontact_meninggal": 0,
        "probable_meninggal": 26
      }
    ]
  }
}
```

Dimension table

1. Province table
 - a. province_id
 - b. province_name

2. District table

- a. district_id
- b. province_id
- c. district_name

3. Case table

- a. Id
- b. Status name (suspect, closecontact, probable, confirmation)
- c. Status detail

Fact table

1. Province Daily Table
 - a. Id (auto generate)
 - b. province_id
 - c. case_id
 - d. date
 - e. total

2. Province Monthly Table

- a. Id (auto generate)
- b. province_id
- c. case_id
- d. month
- e. total

3. Province Yearly Table

- a. Id (auto generate)
- b. province_id
- c. case_id
- d. year
- e. total

4. District Monthly Table

- a. Id (auto generate)
- b. district_id
- c. case_id
- d. month
- e. total

5. District Yearly Table

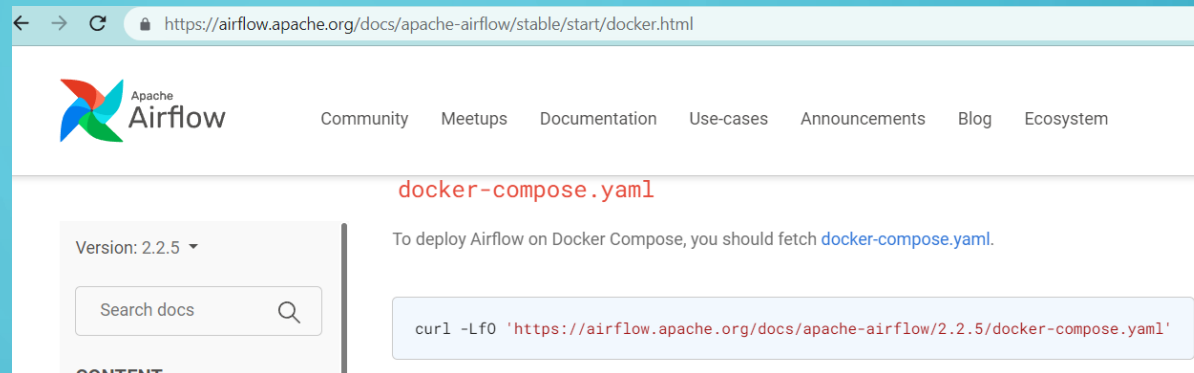
- a. Id (auto generate)
- b. district_id
- c. case_id
- d. year
- e. total

Table Result Expectation

SETUP DOCKER ENVIRONMENT

Step 1

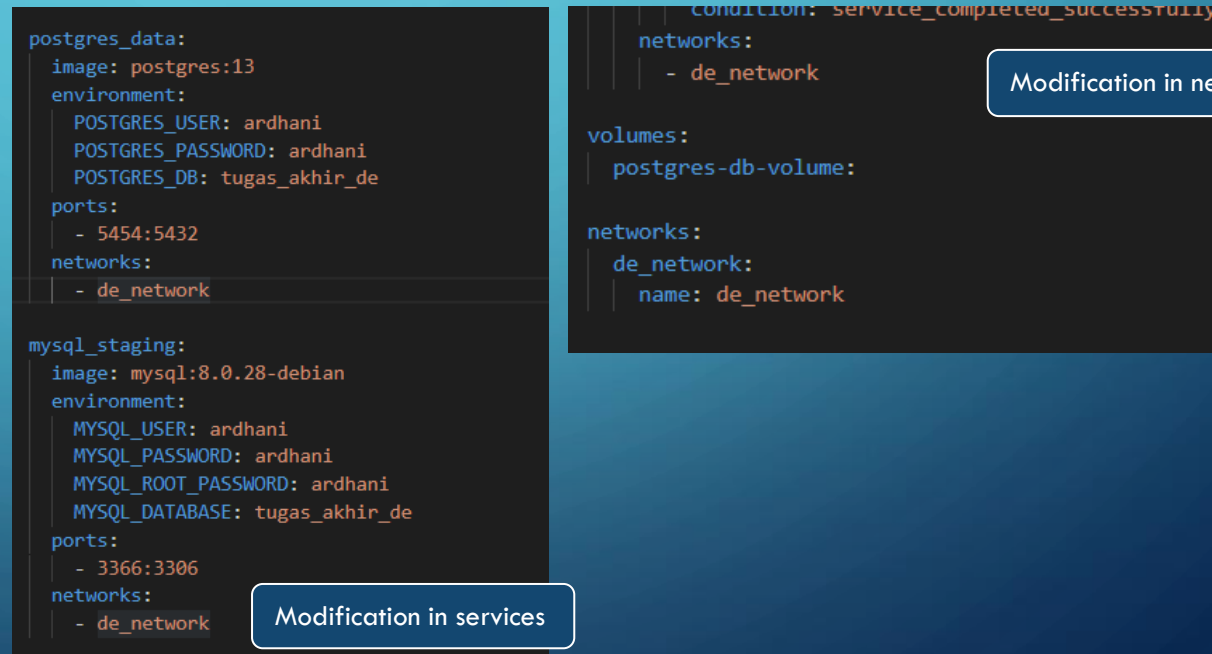
Get docker-compose.yml from
<https://airflow.apache.org/docs/apache-airflow/stable/start/docker.html>



Step 2

Modify the .yml file, add the postgres
& mysql services and network

Makesure all the services within same
networks



SETUP DOCKER ENVIRONMENT

Step 3

Setup Airflow Environment by execute command below

```
mkdir -p ./dags ./logs ./plugins  
echo -e "AIRFLOW_UID=$(id -u)" > .env
```

Step 4

Setup Airflow Environment by execute command below

```
docker-compose up airflow-init
```

After initialization is complete, you should see a message like below.

```
airflow-init_1      | Upgrades done  
airflow-init_1      | Admin user airflow created  
airflow-init_1      | 2.2.5  
start_airflow-init_1 exited with code 0
```

The account created has the login `airflow` and the password `airflow`.

Step 5

Running Aiflow

```
docker-compose up
```

SETTING AIRFLOW CONFIGURATION

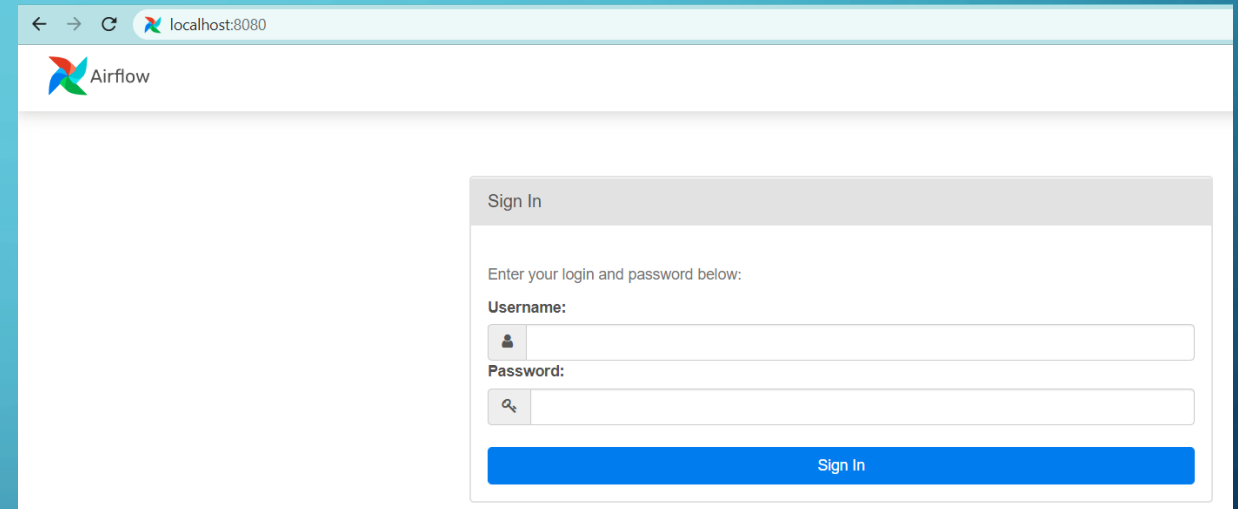
Step 1

Make sure the files and directory is like below

```
airflow_final_project_de
├── dags
│   ├── __pycache__
│   ├── config
│   │   └── data_login_db.json
│   ├── script
│   │   ├── pyscript
│   │   │   ├── __pycache__
│   │   │   └── func_tugas_akhir.py
│   │   └── sqlscript
│   │       ├── agg_insert_data_dim_fact.sql
│   │       ├── create_table_postgres.sql
│   │       └── dag_tugas_akhir_de.py
│   ├── logs
│   ├── plugins
│   ├── .env
│   └── docker-compose.yml
```

Step 2

Access Airflow UI using browser



The screenshot shows a web browser window with the address bar set to localhost:8080. The page displays the Airflow logo and a 'Sign In' form. The form includes a prompt to enter login and password, fields for 'Username' and 'Password', and a 'Sign In' button.

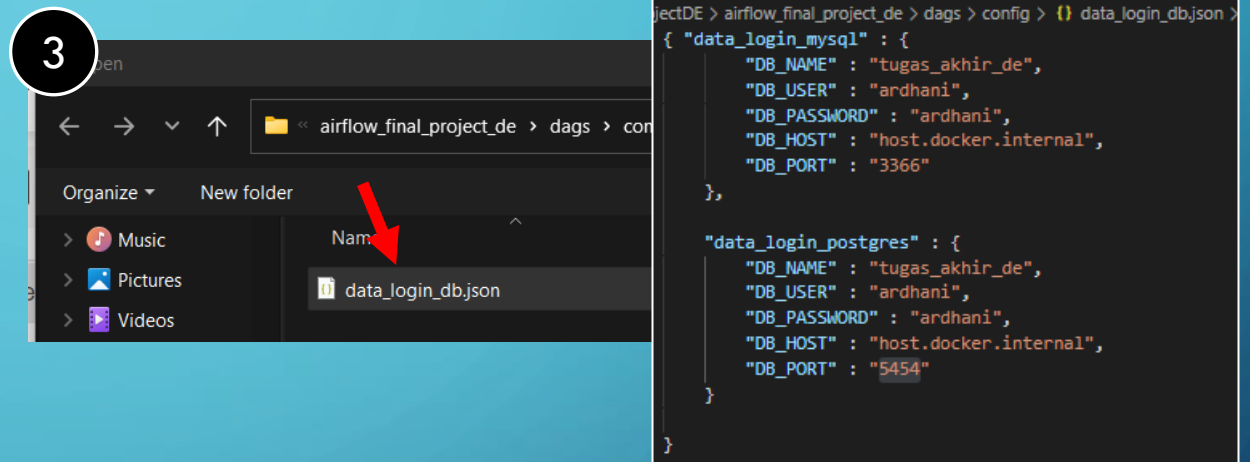
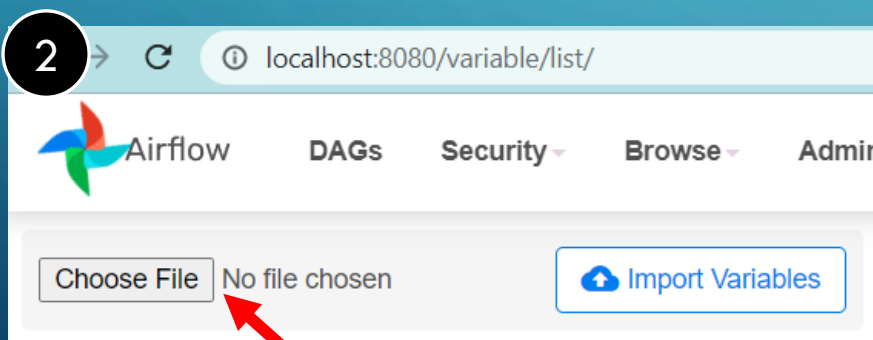
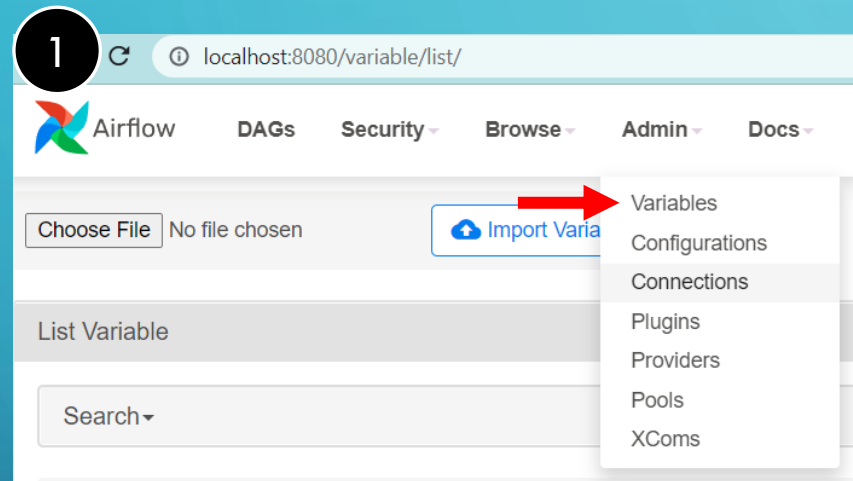
Username : airflow

Password : airflow

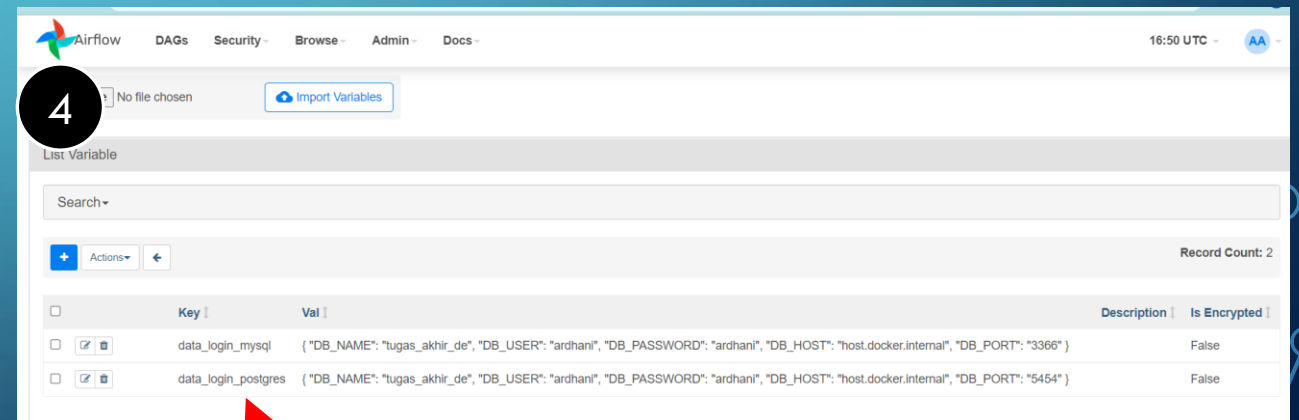
SETTING AIRFLOW CONFIGURATION

Step 3

Upload “data_login_db.json” to setup Aiflow Variables



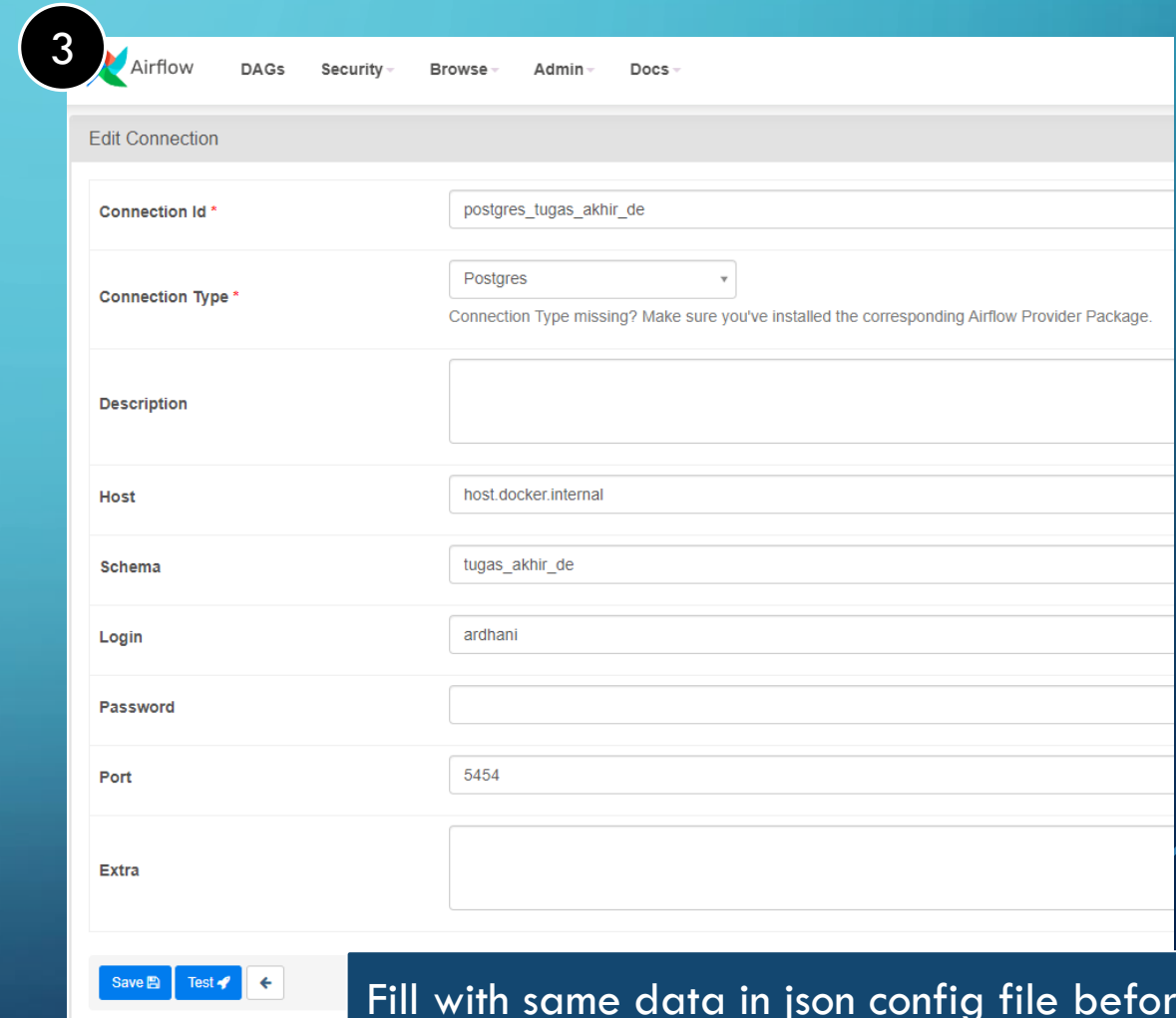
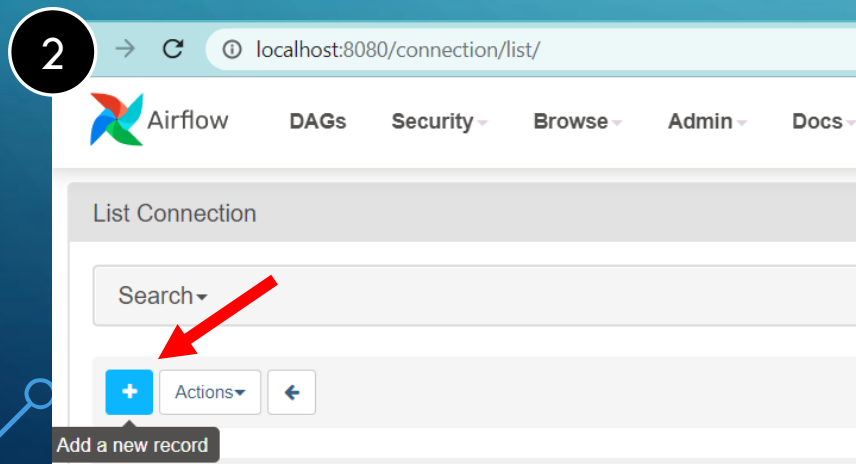
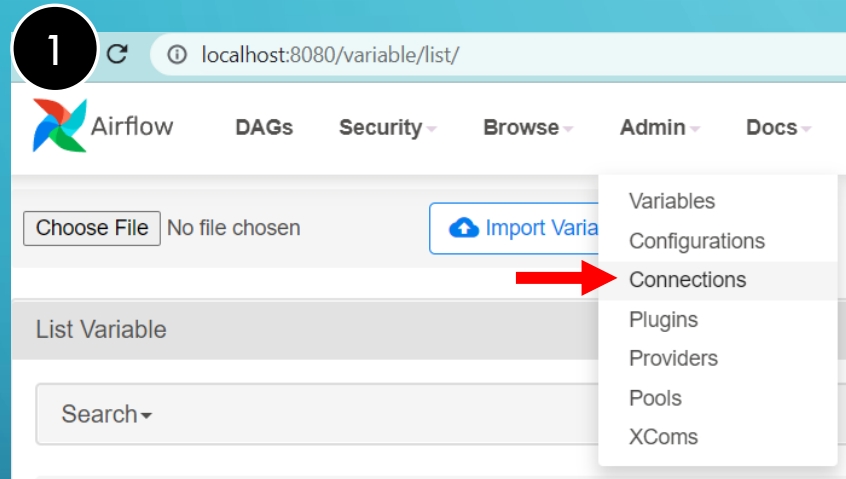
```
projectDE > airflow_final_project_de > dags > config > data_login_db.json >
{
  "data_login_mysql" : {
    "DB_NAME" : "tugas_akhir_de",
    "DB_USER" : "ardhani",
    "DB_PASSWORD" : "ardhani",
    "DB_HOST" : "host.docker.internal",
    "DB_PORT" : "3366"
  },
  "data_login_postgres" : {
    "DB_NAME" : "tugas_akhir_de",
    "DB_USER" : "ardhani",
    "DB_PASSWORD" : "ardhani",
    "DB_HOST" : "host.docker.internal",
    "DB_PORT" : "5454"
  }
}
```



SETTING AIRFLOW CONFIGURATION

Step 3

Setup Airflow Connection to Postgres DB connection



Fill with same data in json config file before

DAG FILE EXPLANATION

Workflow

The screenshot displays the Apache Airflow web interface for a specific DAG. At the top, the navigation bar includes the Airflow logo and links for DAGs, Security, Browse, Admin, and Docs. The current time is 16:58 UTC. The main header shows the DAG name 'dag_tugas_akhir_de' with a 'success' status, a '@daily' schedule, and a next run time of 2022-04-28, 00:00:00. Below this, a toolbar offers various views: Tree, Graph (selected), Calendar, Task Duration, Task Tries, Landing Times, Gantt, Details, and Code. A filter section allows selecting a date (2022-04-27T00:00:01Z), the number of runs (25), and a specific run (scheduled__2022-04-27T00:00:00+00:00). An 'Update' button and a 'Find Task...' search bar are also present. A legend identifies task operators (PostgresOperator, PythonOperator) and their states (queued, running, success, failed, up_for_retry, up_for_reschedule, upstream_failed, skipped, scheduled, deferred, no_status). The bottom section shows the DAG's task flow as a linear sequence of six tasks: get_data_from_api, send_data_to_MySQL, get_from_MySQL_to_Postgres, create_table_postgres, create_table_dim_case, and insert_aggregate_dim_fact. An 'Auto-refresh' toggle is located in the top right of the task graph area.

Airflow DAGs Security Browse Admin Docs 16:58 UTC AA

DAG: dag_tugas_akhir_de success Schedule: @daily Next Run: 2022-04-28, 00:00:00

Tree Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code

2022-04-27T00:00:01Z Runs 25 Run scheduled__2022-04-27T00:00:00+00:00 Layout Left > Right Find Task... Update

PostgresOperator PythonOperator queued running success failed up_for_retry up_for_reschedule upstream_failed skipped scheduled deferred no_status

Auto-refresh

get_data_from_api → send_data_to_MySQL → get_from_MySQL_to_Postgres → create_table_postgres → create_table_dim_case → insert_aggregate_dim_fact

DAG FILE EXPLANATION

Operators in DAG

```
with DAG(
    dag_id='dag_tugas_akhir_de',
    schedule_interval='@daily',
    start_date=datetime(2022,4,25),
    catchup= False
) as dag:
    get_data_from_api = PythonOperator(
        task_id = 'get_data_from_api',
        python_callable = func_tugas_akhir.get_data_from_api,
        op_kwargs = {"con_conf" : config_MySQL , "url_api" : 'https://covid19-public.digitalservice.id/api/v1/rekapitulasi_v2/jabar/harian?level=kab'}
    )

    send_data_to_MySQL = PythonOperator(
        task_id = 'send_data_to_MySQL',
        python_callable = func_tugas_akhir.send_data_MySQL,
        op_kwargs = {"con_conf" : config_MySQL , "url_api" : 'https://covid19-public.digitalservice.id/api/v1/rekapitulasi_v2/jabar/harian?level=kab'}
    )

    get_from_MySQL_to_Postgres = PythonOperator(
        task_id = 'get_from_MySQL_to_Postgres',
        python_callable = func_tugas_akhir.get_from_MySQL_to_Postgres,
        op_kwargs = {"con_conf_mysql" : config_MySQL,"con_conf_postgres" : config_Postgres}
    )

    create_table_postgres = PostgresOperator(
        task_id="create_table_postgres",
        postgres_conn_id="postgres_tugas_akhir_de",
        sql="script/sqlscript/create_table_postgres.sql"
    )

    create_table_dim_case = PythonOperator(
        task_id = 'create_table_dim_case',
        python_callable = func_tugas_akhir.create_table_dim_case,
        op_kwargs = {"con_conf_postgres" : config_Postgres}
    )

    insert_aggregate_dim_fact = PostgresOperator(
        task_id="insert_aggregate_dim_fact",
        postgres_conn_id="postgres_tugas_akhir_de",
        sql="script/sqlscript/agg_insert_data_dim_fact.sql"
    )
```

```
from airflow.models import Variable
```

```
from script.pyscript import func_tugas_akhir
```

```
config_MySQL = Variable.get("data_login_mysql",deserialize_json = True)
```

```
config_Postgres = Variable.get("data_login_postgres",deserialize_json = True)
```

DAG FILE EXPLANATION

Python Function Used

```
def get_data_from_api(url_api,ti):

    df_json_raw = pd.read_json(url_api)
    df_json = pd.DataFrame(df_json_raw.loc['content']['data'])
    df_for_xcom = df_json.to_json(orient = 'records')
    print("df_for_xcom type : ", type(df_for_xcom))
    ti.xcom_push(key='data_covid_jabar',value=df_for_xcom)

def send_data_MySQL(con_conf,ti):
    data_xcom = ti.xcom_pull(key='data_covid_jabar')#, task_ids = 'get_data_from_api')
    # print("data_xcom type : ", type(data_xcom))
    data_covid_jabar = pd.read_json(data_xcom, orient = 'records')
    # print("data_covid_jabar type : ", type(data_covid_jabar))

    CONNECTION_STRING = f"mysql+mysqldb://{con_conf['DB_USER']}:{con_conf['DB_PASSWORD']}@{con_conf['DB_HOST']}:{con_conf['DB_PORT']}/{con_conf['DB_NAME']}"

    engine = create_engine(CONNECTION_STRING)
    if not database_exists(engine.url):
        create_database(engine.url)

    print("GET DATA API : Connection to MySQL : ",database_exists(engine.url))
    data_covid_jabar.to_sql(name = "staging_area_covid_data", con=engine, if_exists="replace", index=False)
```

Airflow DAGs Security Browse Admin Docs

List XComs

Search

Actions

Key	Value
	[{"tanggal": "2020-08-05", "kode_prov": "32", "nama_prov": "Jawa Barat", "kode_kab": "3204", "nama_kab": "Kabupaten Bandung", "SUSPECT": 2210, "CLOSECONTACT": 274, "PROBABLE": 26, "suspect_diisolasi": 31, "suspect_ditolak": 3}
	[{"tanggal": "2020-08-05", "kode_prov": "32", "nama_prov": "Jawa Barat", "kode_kab": "3217", "nama_kab": "Kabupaten Bandung Barat", "SUSPECT": 776, "CLOSECONTACT": 534, "PROBABLE": 7, "suspect_diisolasi": 4, "suspect_ditolak": 3}
	[{"tanggal": "2020-08-05", "kode_prov": "32", "nama_prov": "Jawa Barat", "kode_kab": "3216", "nama_kab": "Kabupaten Bekasi", "SUSPECT": 5536, "CLOSECONTACT": 2127, "PROBABLE": 33, "suspect_diisolasi": 4001, "suspect_ditolak": 0}
	[{"tanggal": "2020-08-05", "kode_prov": "32", "nama_prov": "Jawa Barat", "kode_kab": "3201", "nama_kab": "Kabupaten Bogor", "SUSPECT": 0, "CLOSECONTACT": 0, "PROBABLE": 163, "suspect_diisolasi": 0, "suspect_ditolak": 0}
	[{"tanggal": "2020-08-05", "kode_prov": "32", "nama_prov": "Jawa Barat", "kode_kab": "3207", "nama_kab": "Kabupaten Ciamis", "SUSPECT": 2075, "CLOSECONTACT": 1295, "PROBABLE": 3, "suspect_diisolasi": 2075, "suspect_ditolak": 0}
	[{"tanggal": "2020-08-05", "kode_prov": "32", "nama_prov": "Jawa Barat", "kode_kab": "3203", "nama_kab": "Kabupaten Cianjur", "SUSPECT": 0, "CLOSECONTACT": 0, "PROBABLE": 0, "suspect_diisolasi": 0, "suspect_ditolak": 0}
	[{"tanggal": "2020-08-05", "kode_prov": "32", "nama_prov": "Jawa Barat", "kode_kab": "3209", "nama_kab": "Kabupaten Cirebon", "SUSPECT": 300, "CLOSECONTACT": 442, "PROBABLE": 1, "suspect_diisolasi": 5, "suspect_ditolak": 0}

	tanggal	kode_prov	nama_prov	kode_kab	nama_kab	SUSPECT	CLOSECONTACT	PROBABLE
0	2020-08-05	32	Jawa Barat	3204	Kabupaten Bandung	2210	274	26
1	2020-08-05	32	Jawa Barat	3217	Kabupaten Bandung Barat	776	534	7
2	2020-08-05	32	Jawa Barat	3216	Kabupaten Bekasi	5536	2127	33
3	2020-08-05	32	Jawa Barat	3201	Kabupaten Bogor	0	0	163

*tugas_akhi... data_warehouse dim_case public temp_dim_case temp_fact fact_provinc... data_warehouse dim_case staging_area... × 14

Properties Data ER Diagram tugas_akhir_de 2 Databases tugas_akhir_de Tables staging_area_covid_data

staging_area_covid_data Enter a SQL expression to filter results (use Ctrl+Space)

id	tanggal	kode_prov	nama_prov	kode_kab	nama_kab	SUSPECT	CLOSECONTACT	PROBABLE	suspect_diisolasi	suspect_ditolak
0	2020-08-05	32	Jawa Barat	3204	Kabupaten Bandung	2,210	274	26	31	3
1	2020-08-05	32	Jawa Barat	3217	Kabupaten Bandung Barat	776	534	7	4	3
2	2020-08-05	32	Jawa Barat	3216	Kabupaten Bekasi	5,536	2,127	33	4,001	0
3	2020-08-05	32	Jawa Barat	3201	Kabupaten Bogor	0	0	163	0	0
4	2020-08-05	32	Jawa Barat	3207	Kabupaten Ciamis	2,075	1,295	3	2,075	0
5	2020-08-05	32	Jawa Barat	3203	Kabupaten Cianjur	0	0	0	0	0
6	2020-08-05	32	Jawa Barat	3209	Kabupaten Cirebon	300	442	1	5	0
7	2020-08-05	32	Jawa Barat	3205	Kabupaten Garut	3,000	3,000	0	3,000	0

DAG FILE EXPLANATION

SQL Used

```
create_table_postgres.sql x
FinalProjectDE > airflow_final_project_de > dags > script > sqlscript > create_table_postgres.sql
1 --Table Creation DDL
2 --DIM Table
3 -- 1.Province Table
4 create table if not exists dim_province(
5     province_id text
6     , province_name text
7     , primary key(province_id)
8 );
9 -- 2.District Table
10 create table if not exists dim_district(
11     district_id text
12     , province_id text
13     , district_name text
14     , primary key(district_id)
15     , foreign key(province_id) references dim_province(province_id)
16 );
17 -- 3.Case Table
18 create table if not exists dim_case(
19     case_id serial
20     , status_name text -- SUSPECT, CLOSECONTACT, PROBABLE, CONFIRMATION
21     , status_detail text -- suspect_diisolasi, suspect_discarded, closecontact_dikarantina, closecontact_dikawatani
22     , primary key(case_id)
23 );
24 --Fact Table
25 --1. Province Daily table
26 create table if not exists fact_province_daily(
27     id serial
28     , province_id text
29     , case_id int
30 );

agg_insert_data_dim_fact.sql x
FinalProjectDE > airflow_final_project_de > dags > script > sqlscript > agg_insert_data_dim_fact.sql
1 -- clear the table data before inserting whole updated data
2 truncate dim_province restart identity cascade;
3 truncate dim_district restart identity cascade;
4 truncate dim_case restart identity cascade;
5 truncate fact_province_daily restart identity cascade;
6 truncate fact_province_monthly restart identity cascade;
7 truncate fact_province_yearly restart identity cascade;
8 truncate fact_district_monthly restart identity cascade;
9 truncate fact_district_yearly restart identity cascade;
10
11 -- Dim Table data insert
12 -- insert data for dim case
13 INSERT INTO dim_case (status_name,status_detail)
14     SELECT * FROM temp_dim_case;
15
16 -- insert data for dim province
17 insert into dim_province
18     select distinct kode_prov, nama_prov
19     from data_warehouse;
20
21 -- insert data for dim district
22 insert into dim_district
23     select distinct kode_kab,kode_prov,nama_kab
24     from data_warehouse;
25
26 -- insert data to temp_fact for pre processed data before insert to fact table
27 insert into temp_fact
28     select kode_prov,kode_kab, tanggal::date,
29         unnest(array ['suspect_diisolasi','suspect_discarded','suspect_meninggal','closecontact_dil
30         'closecontact_dikarantina','closecontact_dikawatani','probable','confirmation']) as status_name,
```

DAG FILE EXPLANATION

Dim Table Creation

```
def create_table_dim_case(con_conf_postgres,ti):
    data_xcom = ti.xcom_pull(key='data_covid_jabar')#, task_ids = 'get_data'
    # print("data_xcom type : ", type(data_xcom))
    data_covid_jabar = pd.read_json(data_xcom, orient = 'records')
    # algorithm for splitting the status name & details

    temp = data_covid_jabar.columns
    status_name = []
    status_detail = []

    #check for every colum name in dataframe
    for nama_kolom in temp:
        # column name with uppercase letter is the name of status
        if nama_kolom.isupper():
            status_name.append(nama_kolom)
        # detail of the status
        else:
            status_detail.append(nama_kolom)

    # splitting the 2 words of status name and detail the save into list
    merge = []
    for word in status_name:
        for sentence in status_detail:
            split = sentence.split("_")
            if word.lower() in split:
                merge.append([split[0].lower(), split[1]])
    dim_case = pd.DataFrame(merge, columns=['status_name','status_detail'])
```

```
dim_case.to_sql(name = "temp_dim_case", con=engine_postgres, if_exists="replace", index=False)
```

123 SUSPECT	123 CLOSECONTACT	123 PROBABLE	123 suspect_diisolasi	123 suspect_discarded	123 closeco
2,210	274	26	31	2,179	
776	534	7	3	773	
5 536	2 127	33	4 001	1 535	

	status_name	status_detail
0	suspect	diisolasi
1	suspect	discarded
2	suspect	meninggal
3	closecontact	dikarantina
4	closecontact	discarded
5	closecontact	meninggal
6	probable	diisolasi
7	probable	discarded
8	probable	meninggal
9	confirmation	sembuh
10	confirmation	meninggal

temp_dim_case		
	status_name	status_detail
1	suspect	diisolasi
2	suspect	discarded
3	suspect	meninggal
4	closecontact	dikarantina
5	closecontact	discarded
6	closecontact	meninggal
7	probable	diisolasi
8	probable	discarded
9	probable	meninggal
10	confirmation	sembuh
11	confirmation	meninggal

DAG FILE EXPLANATION

Aggregate Function

```
-- insert data to temp_fact for pre processed data before insert to fact table
insert into temp_fact
select kode_prov, kode_kab, tanggal::date,
       unnest(array ['suspect_diisolasi', 'suspect_discarded', 'suspect_meninggal',
                    'closecontact_dikarantina', 'closecontact_discarded', 'closecontact_meninggal',
                    'probable_diisolasi', 'probable_discarded']) as case_id
from data_warehouse;
```

Properties Data ER Diagram

temp_fact Enter a SQL expression to filter results (use Ctrl+Space)

	abc province_id	abc district_id	date	abc case	123 total
1	32	3204	2020-08-05	suspect_diisolasi	31
2	32	3204	2020-08-05	suspect_discarded	2,179
3	32	3204	2020-08-05	suspect_meninggal	0
4	32	3204	2020-08-05	closecontact_dikarantina	0
5	32	3204	2020-08-05	closecontact_discarded	274
6	32	3204	2020-08-05	closecontact_meninggal	0
7	32	3204	2020-08-05	probable_diisolasi	0
8	32	3204	2020-08-05	probable_discarded	0

```
-- insert data to fact_province_monthly
insert into fact_province_monthly(province_id, case_id, "month", total)
select province_id, dc.case_id, to_char(date, 'YYYY-MM') as "month", sum(total) as total
from temp_fact tf inner join dim_case dc on concat(dc.status_name, '_', dc.status_detail) = tf."case"
group by province_id, case_id, "month"
order by province_id, case_id, "month" asc;
```

Grid	123 case_id	ABC status_name	ABC status_detail
1	1	suspect	diisolasi
2	2	suspect	discarded
3	3	suspect	meninggal
4	4	closecontact	dikarantina
5	5	closecontact	discarded
6	6	closecontact	meninggal
7	7	probable	diisolasi
8	8	probable	discarded
9	9	probable	meninggal
10	10	confirmation	sembuh
11	11	confirmation	meninggal



Properties

Data

ER Diagram

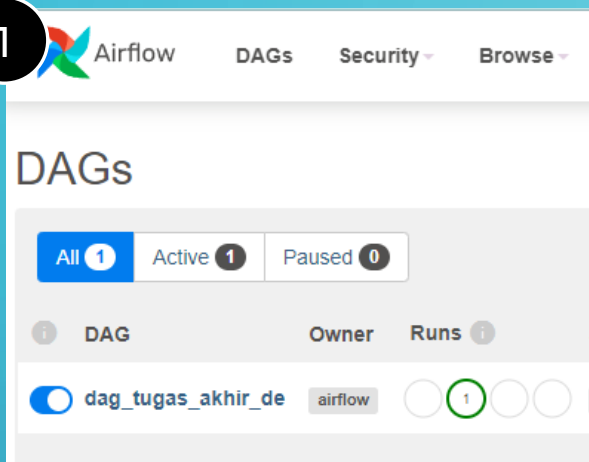
fact_province_monthly

Enter a SQL expression to filter results (use Ctrl+Space)

	123 id	ABC province_id	123 case_id	ABC month	123 total
1	1	32	1	2020-08	22,951
2	2	32	1	2020-09	2,043
3	3	32	1	2020-10	2,311
4	4	32	1	2020-11	3,481
5	5	32	1	2020-12	4,677
6	6	32	1	2021-01	5,337
7	7	32	1	2021-02	2,656
8	8	32	1	2021-03	3,294
9	9	32	1	2021-04	1,813
10	10	32	1	2021-05	1,613
11	11	32	1	2021-06	5,168

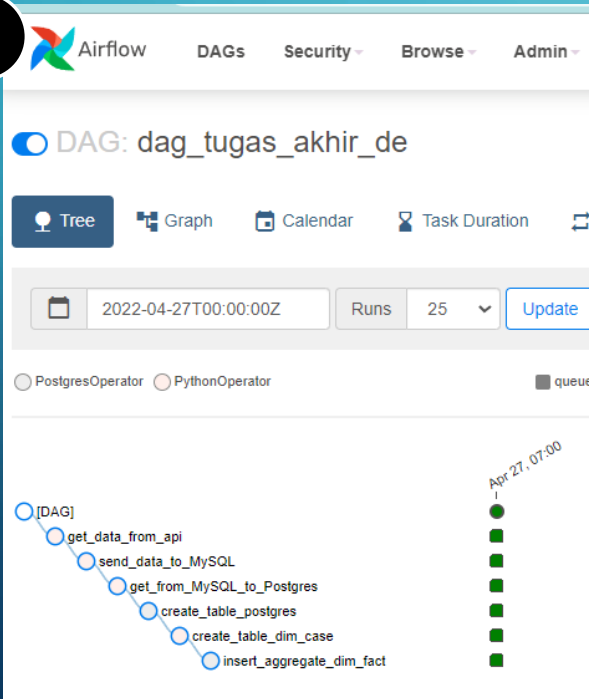
RUNNING AIRFLOW DAG

1



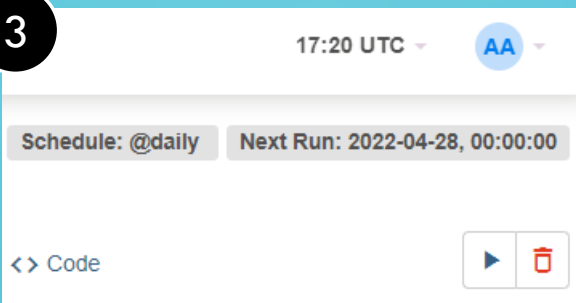
The screenshot shows the Airflow web interface with the 'DAGs' tab selected. At the top, there are tabs for 'All' (1), 'Active' (1), and 'Paused' (0). Below this is a table with columns 'DAG', 'Owner', and 'Runs'. The first entry is 'dag_tugas_akhir_de' owned by 'airflow', with a status indicator showing 1 successful run.

2



The screenshot shows the details for the DAG 'dag_tugas_akhir_de'. It includes a 'Tree' view of the DAG structure, a 'Graph' view, and a 'Calendar' view. The 'Runs' section shows a list of runs, with the most recent run at '2022-04-27T00:00:00Z' having 25 runs. The DAG structure is shown as a tree with tasks: [DAG], get_data_from_api, send_data_to_MySQL, get_from_MySQL_to_Postgres, create_table_postgres, create_table_dim_case, and insert_aggregate_dim_fact.

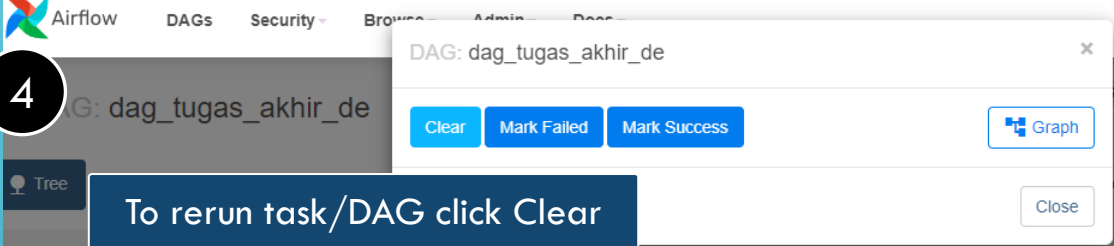
3



The screenshot shows the 'Run' button (a blue play icon) and the 'Code' button (a red trash icon) for the DAG. The 'Schedule' is set to '@daily' and the 'Next Run' is '2022-04-28, 00:00:00'.

Play button to run DAG
in new timeline

4

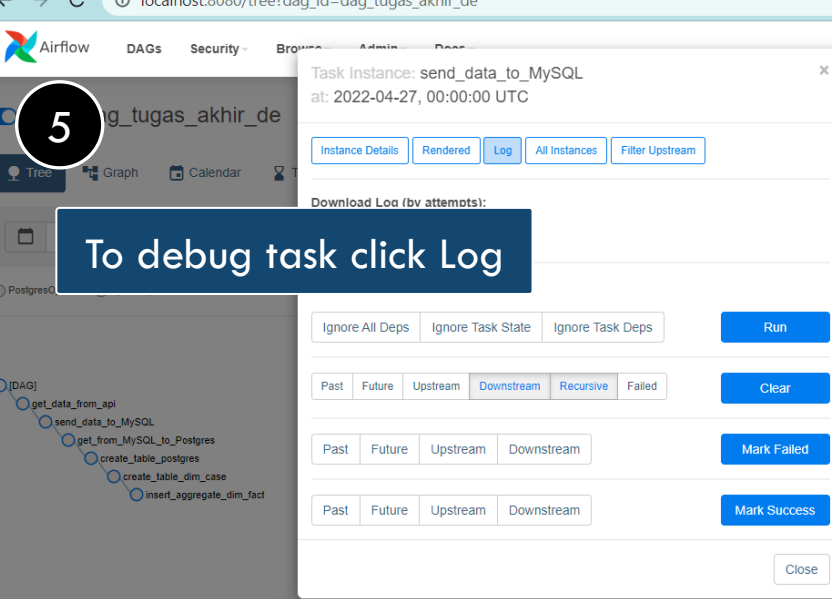


The screenshot shows the 'Clear' button (a blue button with a trash icon) and the 'Mark Failed' button (a blue button with a red X icon). The 'DAG: dag_tugas_akhir_de' is selected.

To rerun task/DAG click Clear

Select DAG or Task

5

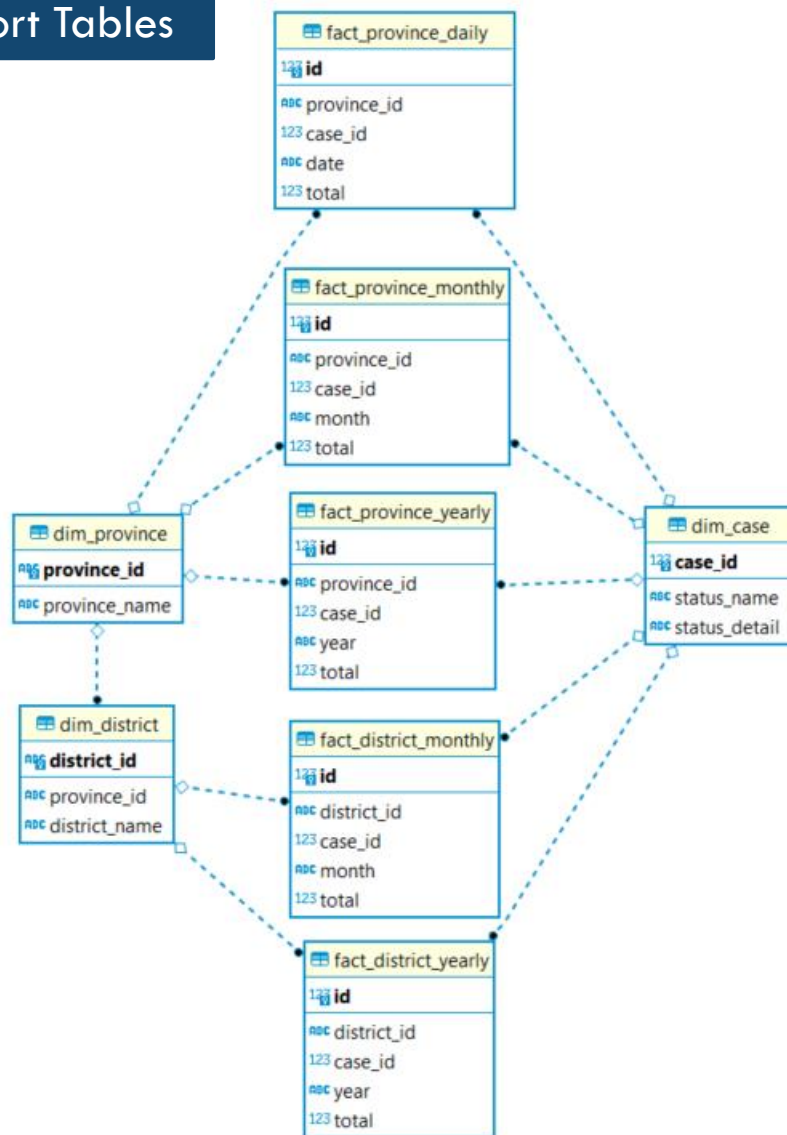


The screenshot shows the 'Task Instance: send_data_to_MySQL' details. It includes a 'Log' button (a blue button with a magnifying glass icon) and the 'Run' button (a blue button with a play icon). The 'Task Instance' is 'send_data_to_MySQL' at '2022-04-27, 00:00:00 UTC'.

To debug task click Log

ERD RESULT

Report Tables



Other Tables

temp_fact	data_warehouse
ABC province_id	ABC tanggal
ABC district_id	123 kode_prov
ABC date	ABC nama_prov
ABC case	123 kode_kab
123 total	ABC nama_kab
temp_dim_case	123 SUSPECT
ABC status_name	123 CLOSECONTACT
ABC status_detail	123 PROBABLE
	123 suspect_diisolasi
	123 suspect_discarded
	123 closecontact_dikarantina
	123 closecontact_discarded
	123 probable_diisolasi
	123 probable_discarded
	123 CONFIRMATION
	123 confirmation_sembuh
	123 confirmation_meninggal
	123 suspect_meninggal
	123 closecontact_meninggal
	123 probable_meninggal

TABLES RESULT

Dimension Tables

dim_province | Enter a SQL expression to filter results

Grid	province_id	province_name
1	32	Jawa Barat

dim_case | Enter a SQL expression to filter results (use Ctrl+Space)

Grid	case_id	status_name	status_detail
1	1	suspect	diisolasi
2	2	suspect	discarded
3	3	suspect	meninggal
4	4	closecontact	dikarantina
5	5	closecontact	discarded
6	6	closecontact	meninggal
7	7	probable	diisolasi
8	8	probable	discarded
9	9	probable	meninggal
10	10	confirmation	sembuh
11	11	confirmation	meninggal

dim_district | Enter a SQL expression to filter results (use Ctrl+Space)

Grid	district_id	province_id	district_name
1	3205	32	Kabupaten Garut
2	3279	32	Kota Banjar
3	3277	32	Kota Cimahi
4	3271	32	Kota Bogor
5	3203	32	Kabupaten Cianjur
6	3215	32	Kabupaten Karawang
7	3212	32	Kabupaten Indramayu
8	3210	32	Kabupaten Majalengka
9	3213	32	Kabupaten Subang
10	3273	32	Kota Bandung
11	3206	32	Kabupaten Tasikmalaya
12	3214	32	Kabupaten Purwakarta
13	3208	32	Kabupaten Kuripan

TABLES RESULT

Fact Tables

fact_province_daily Enter a SQL expression to filter results (use Ctrl+Space)						
	123 id	abc province_id	123 case_id	abc date	123 total	
1	1	32	1	2020-08-05	42,540	
2	2	32	1	2020-08-06	436	
3	3	32	1	2020-08-07	42	
4	4	32	1	2020-08-08	120	
5	5	32	1	2020-08-09	498	
6	6	32	1	2020-08-10	264	
7	7	32	1	2020-08-11	40	
8	8	32	1	2020-08-12	136	
9	9	32	1	2020-08-13	170	
10	10	32	1	2020-08-14	204	
11	11	32	1	2020-08-15	102	
12	12	32	1	2020-08-16	100	

fact_province_monthly Enter a SQL expression to filter results (use Ctrl+Space)						
	123 id	abc province_id	123 case_id	abc month	123 total	
1	1	32	1	2020-08	45,902	
2	2	32	1	2020-09	4,086	
3	3	32	1	2020-10	4,622	
4	4	32	1	2020-11	6,962	
5	5	32	1	2020-12	9,354	
6	6	32	1	2021-01	10,674	
7	7	32	1	2021-02	5,312	
8	8	32	1	2021-03	6,588	
9	9	32	1	2021-04	3,626	
10	10	32	1	2021-05	3,226	
11	11	32	1	2021-06	10,336	
12	12	32	1	2021-07	5,312	

fact_province_yearly Enter a SQL expression to filter results (use Ctrl+Space)						
	123 id	abc province_id	123 case_id	abc year	123 total	
1	1	32	1	2020	35,463	
2	2	32	1	2021	23,972	
3	3	32	1	2022	0	
4	4	32	2	2020	100,562	
5	5	32	2	2021	107,756	
6	6	32	2	2022	0	
7	7	32	3	2020	3,125	
8	8	32	3	2021	0	
9	9	32	3	2022	0	
10	10	32	4	2020	55,133	
11	11	32	4	2021	72,689	
12	12	32	4	2022	0	

fact_district_monthly Enter a SQL expression to filter results (use Ctrl+Space)						
	123 id	abc district_id	123 case_id	abc month	123 total	
1	1	3201	1	2020-08	1,004	
2	2	3201	1	2020-09	588	
3	3	3201	1	2020-10	1,034	
4	4	3201	1	2020-11	948	
5	5	3201	1	2020-12	964	
6	6	3201	1	2021-01	658	
7	7	3201	1	2021-02	796	
8	8	3201	1	2021-03	448	
9	9	3201	1	2021-04	148	
10	10	3201	1	2021-05	244	
11	11	3201	1	2021-06	142	
12	12	3201	1	2021-07	6	

fact_district_yearly Enter a SQL expression to filter results (use Ctrl+Space)						
	123 id	abc district_id	123 case_id	abc year	123 total	
1	1	3201	1	2020	2,269	
2	2	3201	1	2021	1,221	
3	3	3201	1	2022	0	
4	4	3201	2	2020	6,125	
5	5	3201	2	2021	1,347	
6	6	3201	2	2022	0	
7	7	3201	3	2020	702	
8	8	3201	3	2021	0	
9	9	3201	3	2022	0	
10	10	3201	4	2020	0	
11	11	3201	4	2021	0	
12	12	3201	4	2022	0	