



Building Modern Data Stacks

Foundation First !!!


Four Key Questions

- I. Where do we consolidate our data ? > [Storage](#)
- II. How will we get it there ? > [Ingestion](#)
- III. How will we clean it up? > [Transformation](#)
- IV. How will we analyze it? > [Reporting](#)



BRIAN GWAYI





The
Big Choice

Data Stack

Popular Options

Storage > [Snowflake](#), [BigQuery](#), [s3](#), Redshift

Ingestion > [Airbyte](#), [Airflow](#), Fivetran

Transformation > [dbt](#)

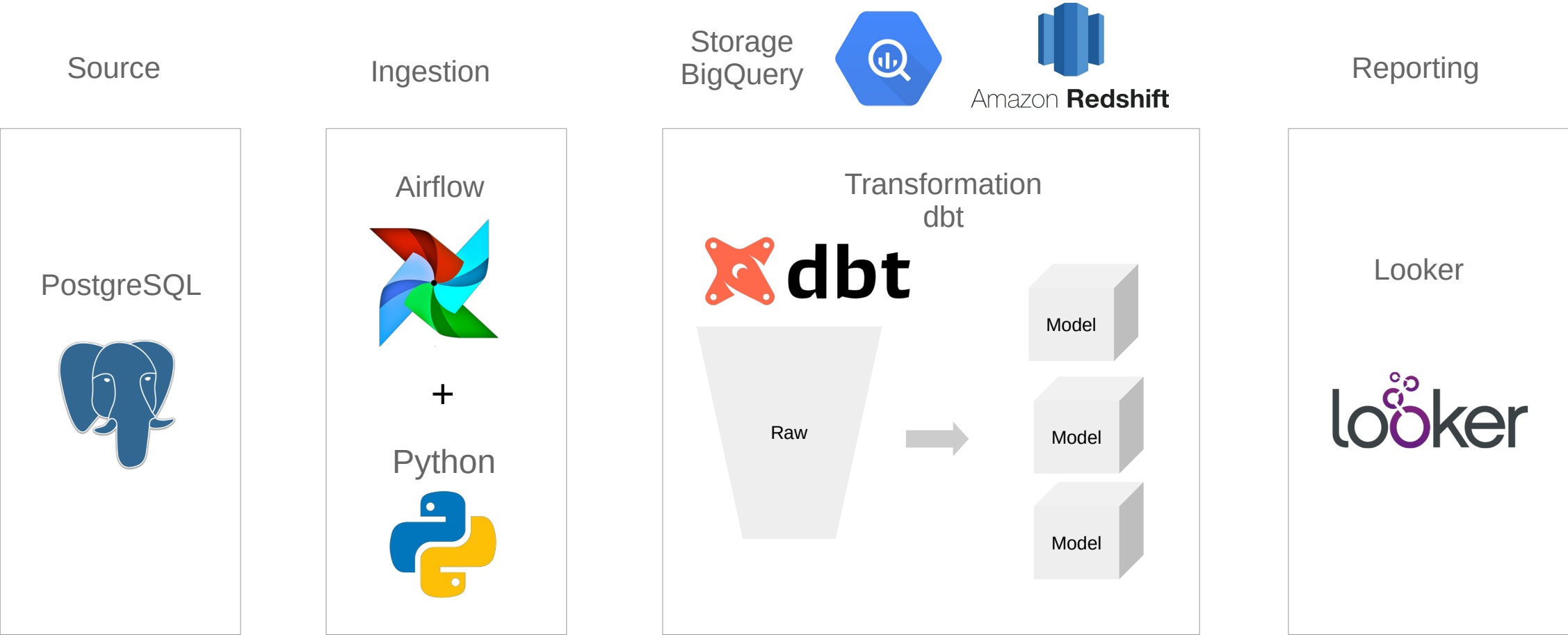
Reporting > [Tableau](#), Power BI, [Looker](#), Superset

N/B This is not an exhaustive list.

BRIAN GWAYI



Data Stack Architecture Design





End Goal

Put data to use



“Data is like garbage. You’d better know what
you are going to do with it before you collect it.”

~ Mark Twain

BRIAN GWAYI



Content

01

Storage/Database

Setting up Google [BigQuery](#)

Setting up [AWS Redshift](#)

02

Ingestion

Setting up [Apache Airflow](#)

Writing ELT Python script

Orchestrate data pipeline

03

Transformation

Setting up [dbt](#)

04

Reporting

Connecting [Looker](#)

02

Ingestion

Setting up Apache Airflow

- [Documentation](#)

- [Deployment](#)

Setting up ETL Directory

- [db.credentials.py](#)

Est connection wit databases

- [sql_queries.py](#)

Queries to extract & load data

- [elt/etl.p](#)

Run all operations

- [main.py](#)

Maintain Operation flow

02

Ingestion

Setting up Apache Airflow

[- Documentation](#)

[- Deployment](#)

Writing ELT Python Scripts

[- .py Code - Extract & Load](#)

Python

```
# importing libraries
```

```
from airflow.decorators import dag, task
from datetime import datetime, timedelta
import requests
from google.cloud import bigquery
import pandas as pd
import psycopg2
from io import StringIO
```

02

Ingestion

Setting up Apache Airflow

- [Airflow Documentation](#)
- [Production Deployment Documentation](#)

Writing ELT Python Script

- [.py Code - Extract & Load](#)

Python

instantiating DAG

```
args{  
    "owner": "gwayi",  
    "retries": 1,  
    "retry_delay": timedelta(minutes=5)  
}
```

```
@dag(  
    default_arguments = args  
    schedule=timedelta(minutes=30),  
    start_date=datetime(2024, 7, 29),  
    catchup=False,  
    tags=['Team B']  
)
```


02

Writing ELT Python Script - [.py Code - Extract & Load](#)

Python

```
@task()
def gt_tbls(conn):

    sql = """SELECT table_name
FROM information_schema.tables
WHERE table_type = 'BASE TABLE'
AND table_catalog = 'adventure_works'
AND table_schema NOT IN
('pg_catalog','information_schema');"""

    cursor = conn.cursor()
    cursor.execute(sql)
    tbls=cursor.fetchall()

    conn.commit()
    conn.close()

    tbls = [x[0] for x in tbls]
    Return tbls
```

02

Writing ELT Python Script - [.py Code - Extract & Load](#)

Python

```
@task()
def xt_tbls(tbls):

    dataframe = {}
    for tbl in tbls:
        sql = f"SELECT * FROM {tbl} WHERE
              createdAt <= (convert(datetime2, {last_rundate})) OR
              modifiedAt <=(convert(datetime2, {last_rundate}))"
        dataframe[tbl] = pd.read_sql(sql, conn)

    return dataframe
```

02

Writing ELT Python Script - [.py Code - Extract & Load](#)

Python

```
@task()

def upsert_tbls(df):
    client = bigquery.Client()
    table_id = "adventureworks-431609.adw_dwh.customer"
    job = client.load_table_from_dataframe(df, table_id)
    job.result()
    print(f"uploading data to Google BigQuery is {job.state}")
upsert_tbla()
```

Ingestion

Orchestrating & Running Workflow – Apache Airflow

Airflow

DAGsCluster ActivityDatasetsSecurityBrowseAdminDocs

09:12 EAT (+03:00)BG

DAG: myjobmag_etl_pipeline

Schedule: 1:00:00Next Run ID: 2024-08-01, 08:00:00 EAT▶◻

08 / 01 / 2024📅 08 : 31 : 58 AMAll Run Types▼All Run States▼Clear Filters

Auto-refresh🔄25▼

Press **shift** + **/** for Shortcuts

deferredfailedqueuedremovedrestartingrunningscheduledshutdownskippedsuccessup_for_rescheduleup_for_retryupstream_failedno_status

ClearMark state as...▼

<<» DAGRunmyjobmag_etl_pipeline / 🕒 2024-08-01, 08:00:00 EAT

⚠️ DetailsGraphGanttCodeAudit Log

Duration

00:01:01Aug 01, 08:44

00:00:30

00:00:00

gt_response_ke
tf_response_ke
gt_response_gh
tf_response_gh
gt_response_zk
tf_response_zk
gt_response_ng
tf_response_ng
merge_res
gt_delta
load_delta

+

-

↺

⛶

gt_response_zk
■ success
@task

tf_response_zk
■ success
@task

gt_response_gh
■ success
@task

tf_response_gh
■ success
@task

gt_response_ke
■ success
@task

tf_response_ke
■ success
@task

gt_response_ng
■ success
@task

tf_response_ng
■ success
@task

merge_res
■ success
@task

gt_delta
■ success
@task

load_delta
■ success
@task

Layout:
Left -> Right ▼

React Flow

02 Ingestion

Data loaded in Google BigQuery

Ingestion

Data loaded in Google BigQuery

Explorer

+ ADD

<

Search BigQuery resources

?

Viewing resources.

SHOW STARRED ONLY

adventureworks-431609

Queries

Notebooks

Data canvases

External connections

adw_dwh

customer

employees

orders

product

productcategory

Untitled query

RUN

SAVE

DOWNLOAD

SHARE

SCHEDULE

MORE

```

1 SELECT *
2 FROM
3 `adventureworks-431609.adw_dwh.customer`
4 LIMIT 1000

```

Query results

JOB INFORMATION

RESULTS

CHART

JSON

EXECUTION DETAILS

EXECUTION GRAPH

Row	customerid	firstname	lastname	fullname
1	1305	A.	Leonetti	A. Leonetti
2	829	Ed	Dudenhoefer	Ed Dudenhoefer
3	1953	H.	Valentine	H. Valentine
4	1917	Abe	Tramel	Abe Tramel
5	323	Amy	Alberts	Amy Alberts
6	735	Amy	Consentino	Amy Consentino
7	437	Ann	Beebe	Ann Beebe
8	1033	Ann	Hass	Ann Hass
9	801	Bev	Desalvo	Bev Desalvo
10	919	Bob	Gage	Bob Gage
11	1099	Bob	Hodges	Bob Hodges
12	509	Eli	Bowen	Eli Bowen

03

Transformation

Getting started with dbt

- Getting started documentation

Terminal

```
python -m venv adw_dbt # create virtual environment  
cd adw_dbt # change into directory  
source adw_dbt-env/bin/activate # activate environment  
pip install dbt-core dbt-bigquery # install dbt + adapter  
dbt -version # check version  
dbt init <project_name> # initiate dbt project  
dbt debug # debug setup  
dbt run # run dbt models  
dbt run --full-refresh
```

03

Transformation Building Data Models

Python

03

Transformation Testing Models

Python