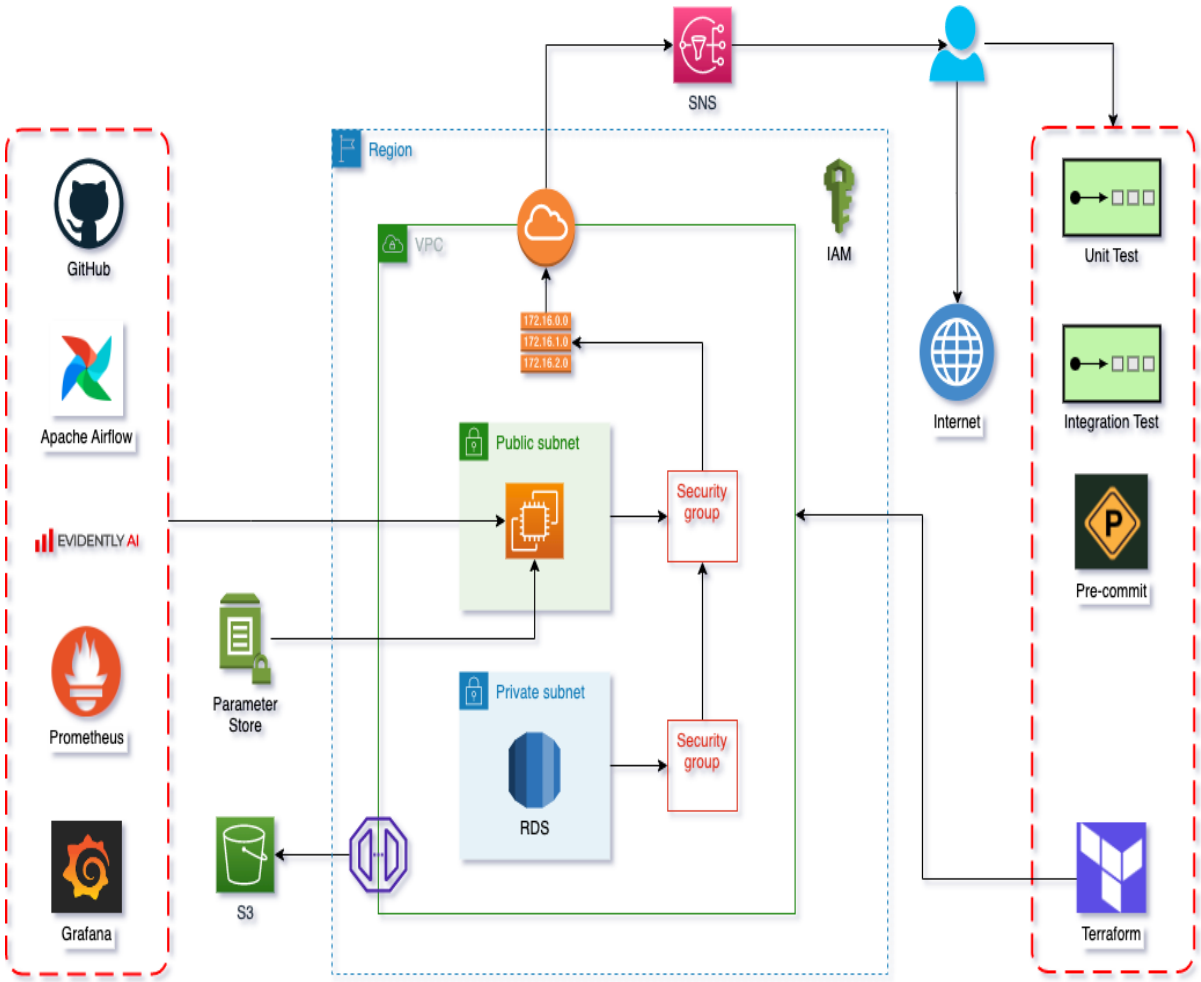
Waiter Tips Prediction Run Book

In this **Waiter Tips Prediction** MLOps project, we predict waiter tips with **XGBoost** based on features named ***total bill, gender, smoker, day, time*** *and* ***size***. We first deploy resources on *AWS* with **Terraform**. Once we have all the data, the model, and scripts on the local machine and have completed the tests, we commit the project directory onto *GitHub*, which, through *Actions*, installs the files on Ubuntu virtual machine.

Later, we initiate the **MLFlow** and **Airflow** servers to track experiments and manage the workflow. **Airflow** automatically runs every month, checking if any new data exists in the **S3** bucket and retrieving it from there to train a new model. The application saves the latest model in the **S3** bucket to use in the production environment. We can watch all metrics by **Evidently** on the localhost and by **Prometheus** on the **Grafana** dashboard. Every time we train the model and check for any data and concept drift, the user also receives an email notification.



Users can access the application through a web interface on *Flask*.

We run the tests on the local machine by implementing ***pytest, black, isort, localstack, pre-commit,*** and ***pylint*** before committing to ***GitHub***.

Installation

1. Create an AWS account and get a programmatic access.
2. Deploy AWS resources with *Terraform*. Run in the terraform folder:

   terraform init

   terraform plan

   terraform apply -auto-approve

3. Commit the project to *GitHub*:

   git add .

   git commit -m 'initial commit'

   git push origin main

If Actions are not in place, you can use **wget** command.

4. Install ***brew***, ***Prometheus***, start ***Prometheus*** and ***Grafana*** servers, and create ***PostgreSQL*** databases on ***RDS*** running the following command in the project folder:

   start.sh

Run

1. Start the **MLFlow** server:

   mlflow server -h 0.0.0.0 -p 5000 --backend-store-uri postgresql://DB_USER:DB_PASSWORD@DB_ENDPOINT:5432/DB_NAME --default-artifact-root s3://s3b-tip-predictor/mlflow/

2. Start the Evidently server:

   mlflow server -h 0.0.0.0 -p 5500 --backend-store-uri postgresql://DB_USER:DB_PASSWORD@DB_ENDPOINT:5432/DB_NAME --default-artifact-root s3://s3b-tip-predictor/evidently/

You need to choose the database user and password, and get the RDS database endpoint.

3. Open the **airflow.cfg** and set executor as **LocalExecutor**, and **sql_alchemy_conn** as

   sql_alchemy_conn = postgresql+psycopg2://<user>:<pass>@<host>:5432/<db>

4. Initialize ***Airflow*** database in the project folder (**/app/Waiter-Tips-Prediction**).

   airflow db init

5. Create a user:

   airflow users create --username <username> --password <password> --firstname <firstname> --lastname <lastname> --role Admin --email <email>

6. Start the ***Airflow*** server:

   airflow webserver -p 8080 -D

7. Start the ***Airflow*** scheduler:

   airflow scheduler -D

8. Do port forwarding on VSC for the following ports to access the Web UIs on the local machine:

   - **3000: *Grafana***
   - **3500: *Flask***
   - **3600: *Flask***
   - **5000: *MLflow***
   - **5500: *Evidently***
   - **8080: *Airflow* web server**
   - **8793: *Airflow* scheduler**
   - **9090: *Prometheus***
   - **9091: *Prometheus***

The *App* (**prediction**) runs on port **3500**, ***Evidently*** reports on **3600**, the app's ***Prometheus*** metrics on **9091**.

To start ***MLFlow*** and *Airflow* servers readily, you can use Makefile provided that MLflow server endpoints are updated:

make mlflow_5000

make mlflow_5500

make airflow_web

make airflow_scheduler