Intelligent Data Pipeline

Alankrit Singh

June 2024

1 The Problem

This is a thorough analysis of the codebase of the Intelligent Data Pipeline. I'll cover the code that I have modified in folders and explain every file and how it connects and have the overarching goal of replacing GCP with Azure. I have edited about 33 files to fit Azure and some work is left which I pass on to those who will further work on this project.

2 Analysis

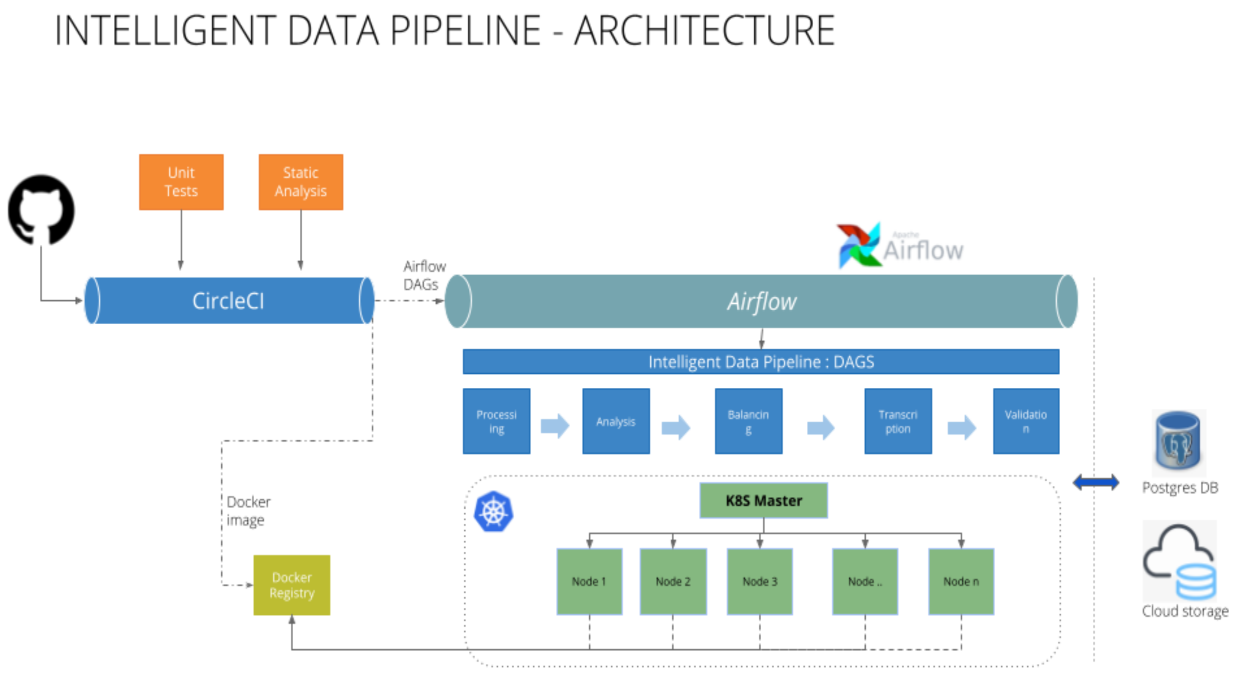

Figure 1: Architecture

2.1 .circleci

This folder contains the config.yml file for CircleCI. CircleCI is a cloud-based continuous integration and continuous delivery (CI/CD) platform that automates the process of software development, testing, and deployment. It enables developers to set up automated pipelines that build, test, and deploy code every time changes are made to a codebase. This automation helps to catch bugs early, ensure code quality, and streamline the software delivery process.

2.1.1 config.yml

The CI/CD pipeline is defined in a .circleci/config.yml file in the root of the repository. This file specifies the jobs, workflows, and steps to be executed. The jobs are described below.

build-dags This job builds and deploys pipeline workflow DAGs.

```
   echo ${GOOGLE_AUTH} > ${HOME}/gcp-key.json
   gcloud auth activate-service-account --key-file ${HOME}/gcp-key.json
```

`Job is authenticating with GCP using a service account key stored in the GOOGLE_AUTH environment variable.`

```
   gcloud --quiet config set project ${GCP_PROJECT}
```

This line sets the GCP project context for subsequent gcloud commands using the `GCP_PROJECT` environment variables.

build The build job prepares the environment necessary for testing a specific package within the project. It ensures that the correct package directory is confirmed and sets up dependencies required for testing, including Python libraries and system packages. The workspace persistence step `persist_to_workspace` makes the package directory available for use in subsequent jobs, such as testing or deployment.

`ekstep_build` The `ekstep_build` job automates the setup of the Ekstep pipelines testing environment, installs dependencies, runs linting checks, and preserves artifacts for further analysis. It ensures that the project environment is properly configured and ready for subsequent testing or deployment stages within the CI/CD pipeline.

deploy This deploy job automates the process of deploying Docker images to Google Container Registry as part of a continuous integration/continuous deployment (CI/CD) pipeline managed by CircleCI. It handles environment-specific tagging of Docker images and ensures that only authorized users with proper authentication can push images to GCR.

```
   echo ${GOOGLE_AUTH} > ${HOME}/gcp-key.json
   gcloud auth activate-service-account --key-file ${HOME}/gcp-key.json
   gcloud --quiet config set project ${GCP_PROJECT}
```

These 3 lines work the same as described for build-dags.

```
   name: Build and Deploy Image
             command: |
               echo ${GOOGLE_AUTH} > ${HOME}/gcp-key.json
               gcloud auth activate-service-account --key-file ${HOME}/gcp-key.json
               gcloud --quiet config set project ${GCP_PROJECT}
```

```
              if [[ << parameters.env_name >> == "test" ]]; then
                echo "Build ..."
                docker build --rm=false -t us.gcr.io/${GCP_PROJECT}/ekstep_data_pipelines:<< parameters.
    package_version >> -t us.gcr.io/${GCP_PROJECT}/ekstep_data_pipelines:<< parameters.env_name >>_<<
    parameters.package_version >> .
              else
               gcloud docker -- pull us.gcr.io/${GCP_PROJECT}/ekstep_data_pipelines:<< parameters.
    package_version >>
               echo "Tag ekstep_data_pipelines ... with << parameters.env_name >>"
               docker tag us.gcr.io/${GCP_PROJECT}/ekstep_data_pipelines:<< parameters.package_version >>
    us.gcr.io/${GCP_PROJECT}/ekstep_data_pipelines:<< parameters.env_name >>_<< parameters.package_version
    >>
              fi
```

Checks the value of env_name parameter.
For test environment:
docker build ...: Builds a Docker image tagged with ${GCP_PROJECT}/ekstep_data_pipelines:
<< parameters.package_version >> and ${GCP_PROJECT}/ekstep_data_pipelines:
<< parameters.env_name >>_<< parameters.package_version >>.
For other environments:
gcloud docker -- pull ...: Pulls the Docker image from Google Container Registry (GCR)
for the specified ${GCP_PROJECT}/ekstep_data_pipelines:<< parameters.package_version >>.
docker tag ...: Tags the pulled image with
${GCP_PROJECT}/ekstep_data_pipelines:<< parameters.env_name >>_<< parameters.package_version >>.

```
gcloud docker -- push us.gcr.io/${GCP_PROJECT}/ekstep_data_pipelines
```

Pushes the locally built or tagged Docker image to Google Container Registry under
${GCP_PROJECT}/ekstep_data_pipelines.

```
- run: name: Remove account details
       command: rm ${HOME}/gcp-key.json ; ls
```

Removes the gcp-key.json file containing the Google Cloud authentication token (GOOGLE_AUTH)
from the environment after completing the Docker image build and push operations.

E2E The purpose of this job is to execute end-to-end tests for a project. Specifies a Docker image (circleci/openjdk:8-jdk-stretch) to use as the environment for running the job. Sets environment variables for PostgreSQL. Checkout: Checks out the project repository to the specified path. E2E Tests: Runs a series of commands to set up and execute the end-to-end tests.

```
wget https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-sdk-265.0.0-linux-x86_64.
 tar.gz
tar -zxf google-cloud-sdk-*
cd google-cloud-sdk
pwd
```

Downloads and installs the Google Cloud SDK.

```
echo ${GOOGLE_AUTH} > ${HOME}/gcp-key.json
./bin/gcloud auth activate-service-account --key-file ${HOME}/gcp-key.json
./bin/gcloud --quiet config set project ${GCP_PROJECT}
export GOOGLE_APPLICATION_CREDENTIALS=${HOME}/gcp-key.json
echo $GOOGLE_APPLICATION_CREDENTIALS
```

Authenticates the service account with GCP and sets the project configuration.

```
wget https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64 -O cloud_sql_proxy
chmod +x cloud_sql_proxy
nohup ./cloud_sql_proxy -dir=./cloudsql -instances=ekstepspeechrecognition:us-central1:crowdsourcedb=tcp
 :5432 &
sleep 25s
cat nohup.out
```

Downloads, configures, and runs the Cloud SQL Proxy to connect to the Cloud SQL instance.

deploy-db This CircleCI job named deploy-db is responsible for setting up the environment, establishing a connection to a PostgreSQL database (hosted on Google Cloud SQL), and running database migrations using yoyo-migrations.

```
- run:
    name: Setup proxy for psql and environment
    command: |
      mkdir -p ~/.ssh
      echo ${SSH} > ~/.ssh/known_hosts
      cat ~/.ssh/known_hosts
      cd ${HOME}
      wget https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-sdk-265.0.0-linux-
 x86_64.tar.gz
      tar -zxf google-cloud-sdk-*
      cd google-cloud-sdk
      pwd
      ./install.sh --quiet
      echo ${GOOGLE_AUTH} > ${HOME}/gcp-key.json
      ./bin/gcloud auth activate-service-account --key-file ${HOME}/gcp-key.json
      ./bin/gcloud --quiet config set project ${GCP_PROJECT}
      export GOOGLE_APPLICATION_CREDENTIALS=${HOME}/gcp-key.json
      echo $GOOGLE_APPLICATION_CREDENTIALS
      wget https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64 -O cloud_sql_proxy
      chmod +x cloud_sql_proxy
      nohup ./cloud_sql_proxy -dir=./cloudsql -instances=${GCP_PROJECT}:us-central1:${DB_INSTANCE}=tcp
 :5432 &
      sleep 25s
      cat nohup.out
      pip install yoyo-migrations
      pip install psycopg2
```

Downloads and installs the Google Cloud SDK to interact with GCP.
Authenticates the service account and configures the project using the gcloud CLI.
Establishes a secure connection to the Cloud SQL instance, enabling local access to the PostgreSQL database hosted on GCP.

```
- run:
name: Update migrations
command: |
  ls ./migrations
  echo " The mode is << parameters.mode >>"
  export env_name=_<< parameters.env_name >>
  if [[ $env_name == "_prod" ]]
  then
    export env_name=''
  fi
  yoyo << parameters.mode >> --database postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@localhost/${
 POSTGRES_DB} --all -b ./migrations
```

Runs the database migrations using yoyo-migrations on the PostgreSQL database. Uses the Cloud SQL Proxy to connect to the PostgreSQL instance on GCP, but the migration process itself is not directly related to GCP.

workflows The following is a description of how the workflow is defined for this set of jobs. The workflow defines a clear sequence of steps for building, testing, and deploying the `ekstep_data_pipelines` package both in the test and production environments. It includes necessary dependencies and approvals to ensure the integrity of the deployment process.

Job	**Purpose**	**Dependencies**
ekstep_data_pipelines_build	Build the ekstep_data_pipelines package	None
ekstep_data_pipelines_deploy_test	Deploy the built package to the test environment	ekstep_data_pipelines_build
ekstep-database-migrations-apply-test	Apply database migrations in the test environment	ekstep_data_pipelines_deploy_test
pipeline-dag-build_test	Build DAGs for pipeline workflows in the test environment	ekstep-database-migrations-apply-test
ekstep_data_pipelines_e2e_build	Run end-to-end tests	pipeline-dag-build_test
approval_for_prod_deploy	Manual approval for production deployment	ekstep_data_pipelines_e2e_build
ekstep_data_pipelines_deploy_prod	Deploy the built package to production	approval_for_prod_deploy
ekstep-database-migrations-apply-prod	Apply database migrations in production	ekstep_data_pipelines_deploy_prod
approval-for-prod-dag-deploy	Manual approval for deploying DAGs in production	ekstep_data_pipelines_deploy_prod
pipeline-dag-build_prod	Build DAGs for pipeline workflows in production	ekstep-database-migrations-apply-prod, approval-for-prod-dag-deploy

Table 1: Summary of CI/CD Workflow