Desafio técnico: Desenvolvedor (a) Backend - Produtos de dados

Tarefa 2: Arquitetura em nuvem

Nessa tarefa, você irá descrever uma arquitetura de solução em nuvem (Google Cloud, AWS ou Azure) para implantação da API desenvolvida. Não há necessidade de desenvolvimento de código ou criação de um projeto real em uma nuvem. O objetivo é criar uma apresentação que especifique e justifique a escolha das ferramentas e tecnologias em cada componente de sua arquitetura.

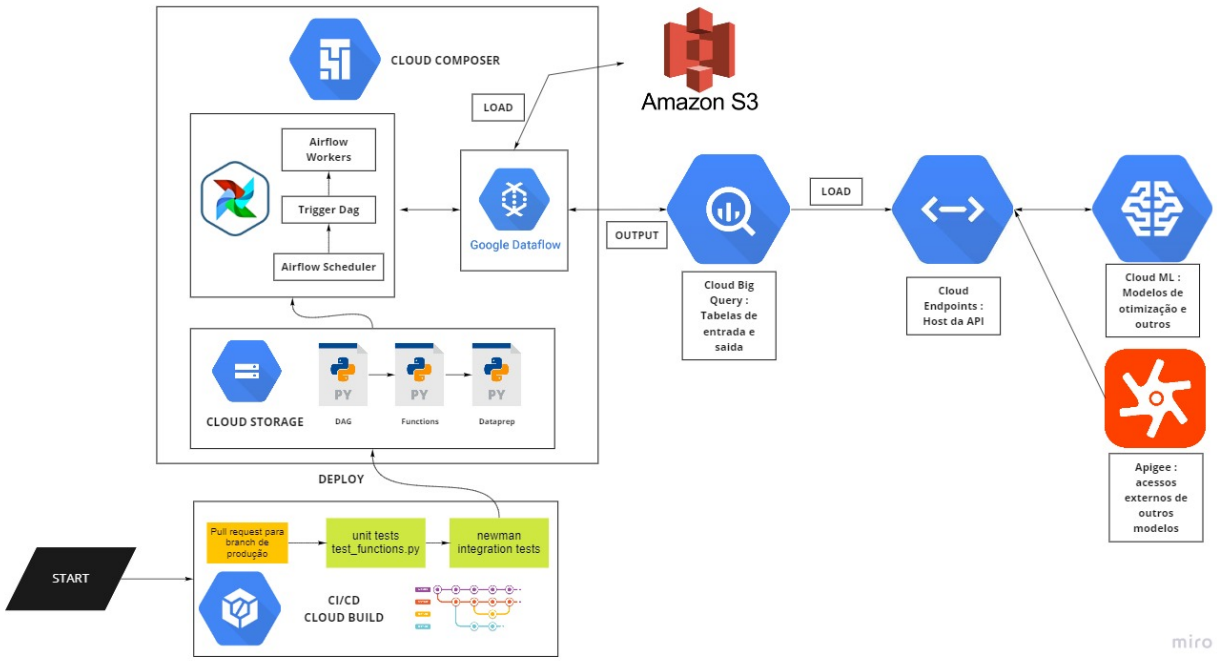
A arquitetura deve contemplar os seguintes requisitos e funcionalidades:

• O sistema será responsável por rodar um job diário para processamento de dados dos arquivos de audiência e disponibilidade consumidos de um serviço de armazenamento (S3, Cloud Storage ou Blob Storage). O conteúdo dos dois arquivos e resultado do script de pré-processamento deverá ser salvo em uma base de dados que servirá para consultas pela API desenvolvida;

• O job roda uma vez ao dia. O sistema não precisa ficar ligado o restante do tempo. Com o sistema desligado é necessário guardar seu estado de alguma forma;

• O sistema pode falhar no meio do processo e precisa voltar de onde parou; • O sistema precisa saber se a gravação no banco de dados ocorreu com sucesso;

• O ideal é ser serverless, se não for possível no momento, a solução deve evoluir para esse objetivo



CI/CD e GOVERNAÇA DE DESENVOLVIMENTO:

O início do desenvolvimento do projeto se dá por meio de um controle de CI/CD com regras de gitflow integradas ao Cloud Build, no bloco start.

Os desenvolvedores trabalham em branches de desenvolvimento e após homologado o código com os devidos testes é realizado um pull request do código para a branch de produção.

A ferramenta cloud build, é acionada por meio do pull-request de produção e realiza testes unitários de preprocessamento com stubs dos dataframes, e testes de integração com a API de ingestão e fornecimento dos dados.

Após concluído os testes, os códigos python são salvos no storage de produção para o trabalho do cloud-composer

Se não concluído os testes o Cloud Build realiza um roll-back do pull request de código e dispara os alertas à equipe de desenvolvimento.

AUTOMAÇÃO E AGENDAMENTO COM O CLOUD-COMPOSER:

Na ferramenta cloud composer é orquestrado uma DAG do airflow que realiza o fluxo de trabalho por meio de uma task, com agendamento diário.

Task Functions: Esta task o acionamento das funções de carregamento dos dados similar ao arquivo Functions/functions.py deste projeto.

As funções de carregamento utilizarão a ferramenta Google Dataflow para realizar o stream dos dados armazenados externamente e armazená-los no DW Cloud Bigquery.

Após o armazenamento, a task dataprep executa as funções de carregamento de dados para os dataframes, o pacote pandas-gbq pode ser utilizado para tornar o desenvolvimento do processamento similar ao que é realizado com a biblioteca pandas neste projeto local.

Após o carregamento a task aciona a função de calculo da mediana, mesclagem com o conjunto de dados de intervalo de tempo disponível, salvando o dataframe resultante em uma tabela no big-query através do dataflow.

Em qualquer momento de falha de ingestão com o S3 ou falha de pre-processamento o airflow interrompe o processo e dispara os alertas devidos para a equipe de desenvolvimento.

HOSTING DA API DE CONSULTA DOS DADOS PROCESSADOS :

A api de consulta dos dados processados, similar ao código api.py na raiz deste projeto, pode ser desenvolvida e hospedada na ferramenta Cloud Endpoints, onde as devidas rotas de consulta por período ou por programa, acionam uma consulta integrada com o big-query.

Um proxy do apigee também pode ser desenvolvido e em suas rotas é possível implementar diretamente as consultadas ao big-query, sobrepondo a tarefa do cloud endpoint, porém a atuação ideal do apigee neste cenário seria para expor as consultas à modelos de otimização externos.

Os modelos de otimização se hospedados dentro do GCP como por exemplo na ferramenta Cloud ML, podem realizar as consultas de dados na ferramenta Cloud Endpoints.