

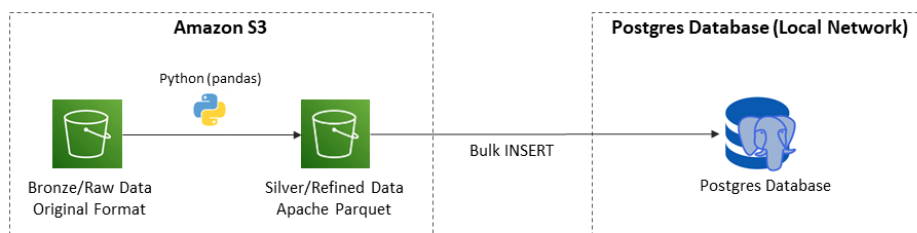
Desafio Dev Python – Descritivo da Solução

A solução desenvolvida para o desafio proposto engloba as seguintes tecnologias: Airflow, Python e Postgres.

Os datasets fornecidos no repositório do GitHub foram armazenados em um bucket do Amazon S3, o intuito foi o de simular o máximo possível o ambiente/plataforma de dados da empresa.

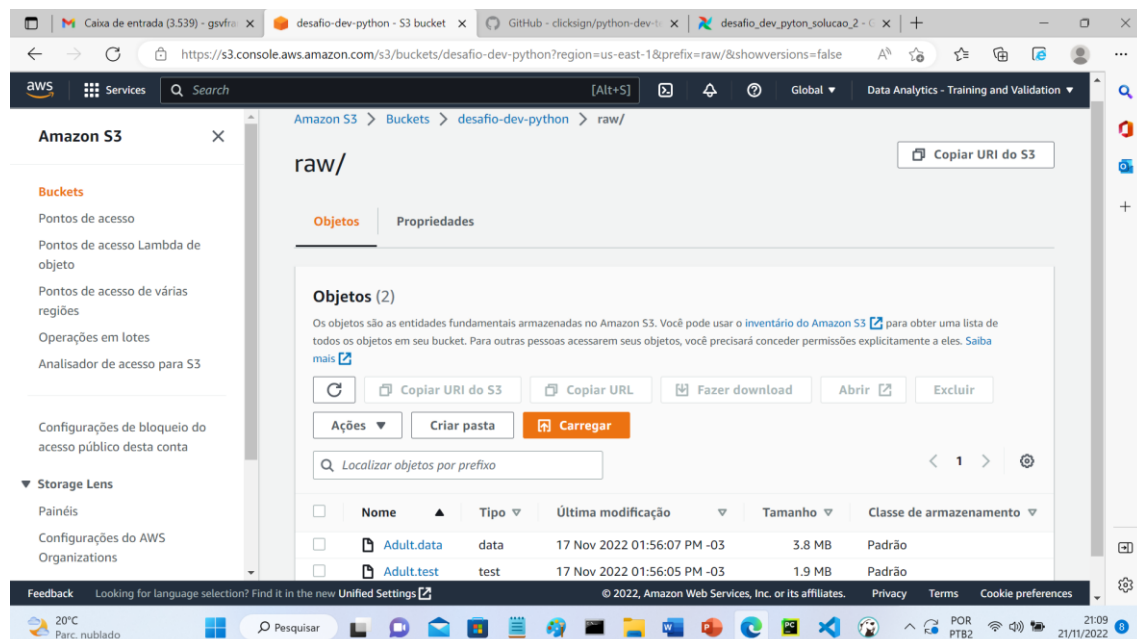
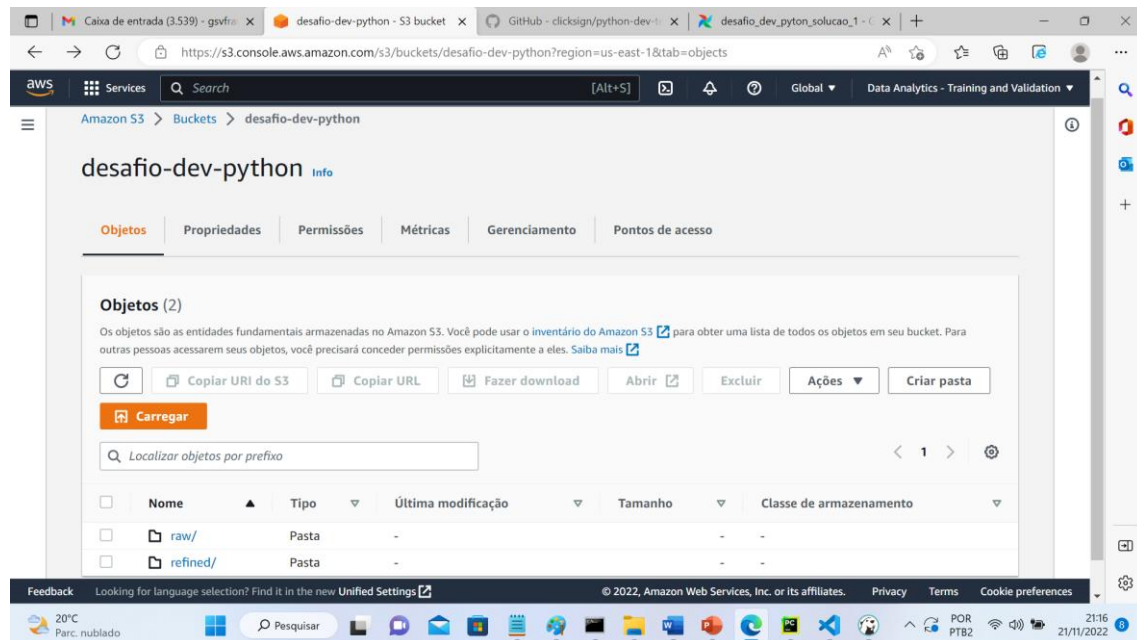
Para execução automática e programada do processo de ETL desenvolvido, a ferramenta escolhida foi o Airflow, foi feito um deploy local da ferramenta por meio do Docker compose. O banco de dados Postgres também foi instalado localmente.

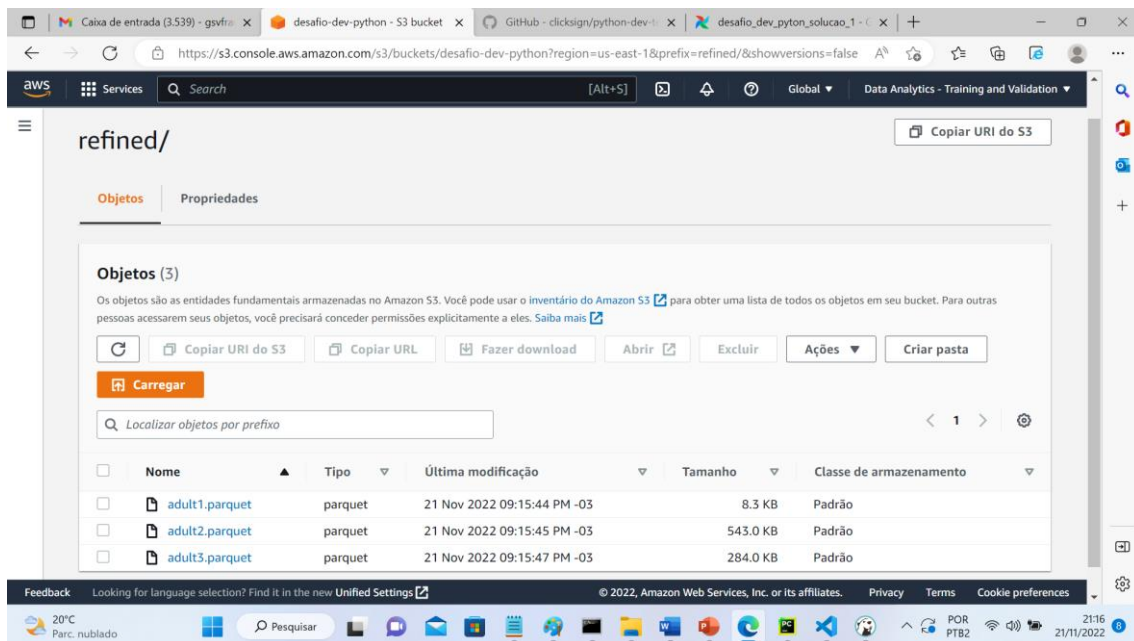
Segue abaixo o desenho da arquitetura da solução:



Para o correto funcionamento da solução é necessário ter o Airflow instalado, localmente ou em algum servidor / máquina virtual, e uma conta na AWS, para usar o serviço S3 de storage / armazenamento de dados.

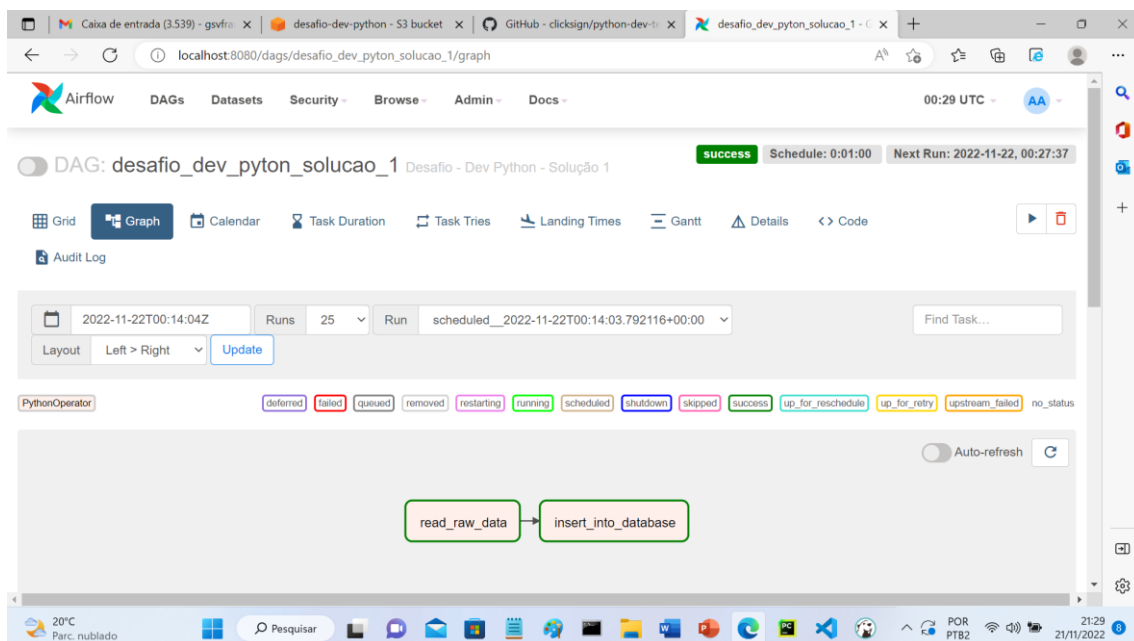
Falando agora sobre o funcionamento do processo de ETL, os datasets de origem (Adult.data e Adult.test), que foram armazenados em um bucket do S3, são lidos por um script Python, que aplica as transformações/ajustes necessários, e escreve o dataset resultante, de saída, no mesmo bucket do S3, em uma pasta/zona refinada (refined).



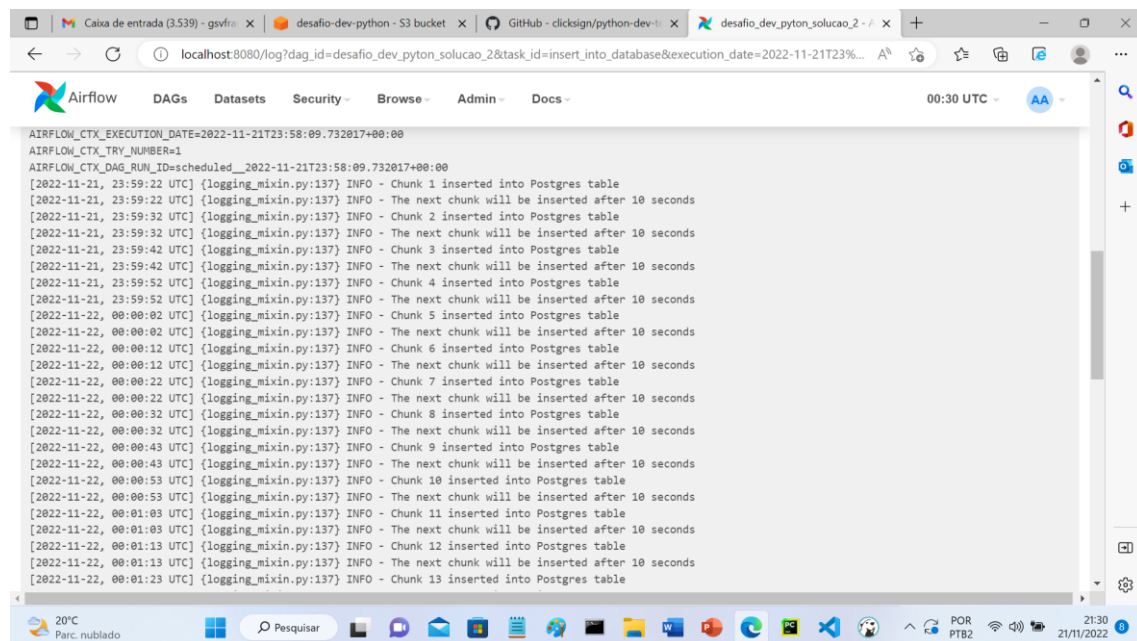
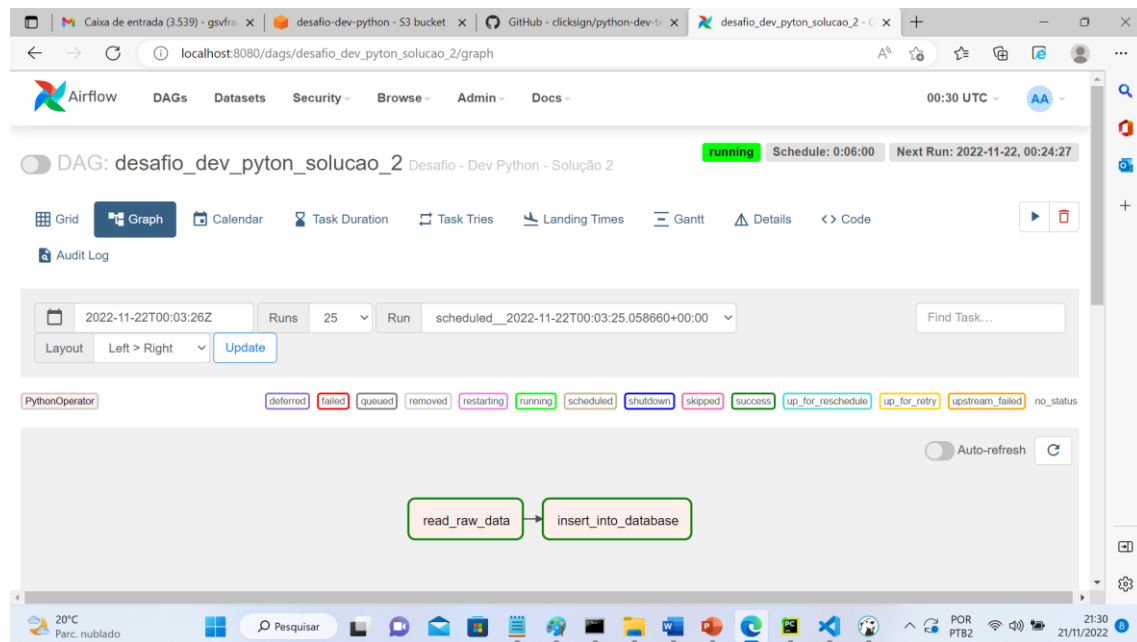


Sendo sincero, não sei se entendi direito o que o desafio esperava em relação ao terceiro requisito: "Seu script deve ser agendado para rodar a cada 10 segundos processando 1.630 registros", e por isso criei duas DAGs no Airflow para fazer o processo de ETL.

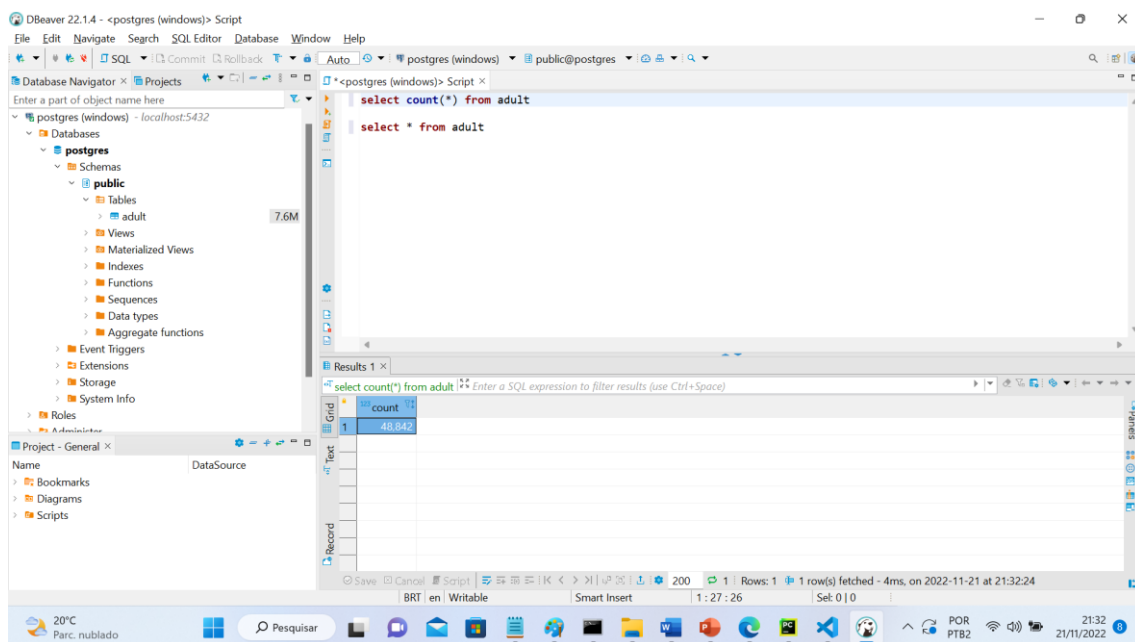
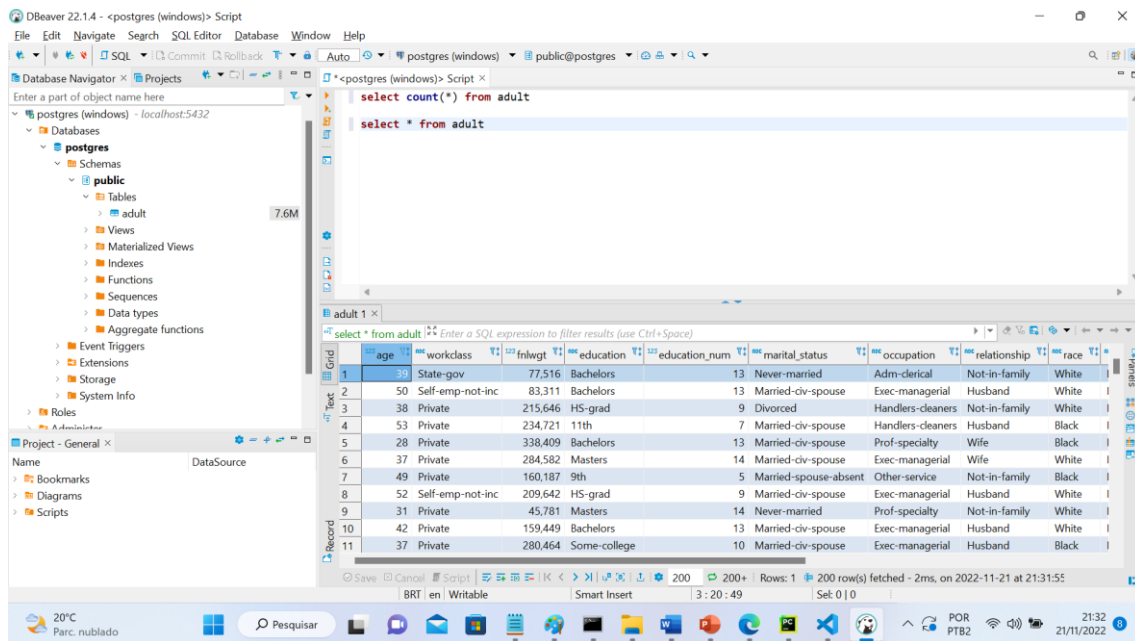
A primeira DAG (desafio_dev_pyton_soluciao_1) lê os datasets de origem do S3, aplica as transformações necessárias, e faz um bulk insert dos dados no Postgres, essa DAG foi agendada para rodar de 30 em 30 segundos, o tempo de execução médio dela é de 14 segundos, não consegui reduzir esse tempo.



A segunda DAG (desafio_dev_pyton_solucao_2) lê os datasets de origem do S3, aplica as transformações necessárias, gera um dataframe do pandas, e dividi o dataframe em chunks de 1630 registros, inserindo os chunks no banco de dados de 10 em 10 segundos.



Para carregar os dados processados, criei uma tabela em um banco de dados Postgres.



Construí também dois scripts Python, que funcionam de acordo com a mesma lógica das DAGs. Já é sabido, mas vale lembrar que os códigos dos scripts tem uma estrutura diferente dos códigos das DAGs.

As DAGs do Airflow foram disponibilizadas na pasta "dags", e os scripts Python foram disponibilizados na pasta "scripts" do GitHub.