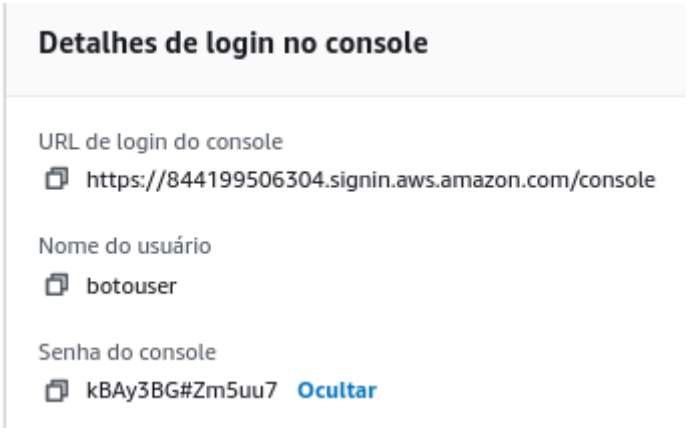
Desafio Parte 1

Ingestão Batch: a ingestão dos arquivos CSV em Bucket Amazon S3 **RAW Zone**. Nesta etapa do desafio deve ser construído um código Python que será executado dentro de um container Docker para carregar os dados locais dos arquivos para a nuvem. Nesse caso utilizaremos, principalmente, as lib boto3 como parte do processo de ingestão via batch para geração de arquivo (CSV).

Atenção: Foi necessário a criação de um usuario para realizar a conexão.



Perguntas dessa tarefa:

- Implementar código Python:

- Ler os 2 arquivos (filmes e series) no formato CSV inteiros, ou seja, sem filtrar os dados

```
filmes_df = pd.read_csv('data/movies.csv', sep='|', low_memory=False)
series_df = pd.read_csv('data/series.csv', sep='|', low_memory=False)
```

- Utilizar a lib boto3 para carregar os dados para a AWS

- Acessar a AWS e grava no S3, no bucket definido com RAW Zone

- no momento da gravação dos dados deve-se considerar o padrão: <nome do bucket>\<camada de armazenamento>\<origem do dado>\<formato do dado>\<especificação do dado>\<data de processamento separada por ano\mes\dia>\<arquivo>

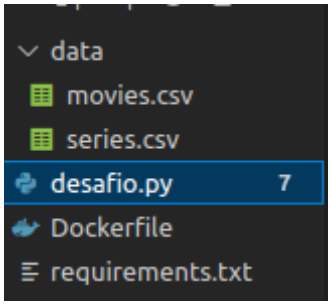
Por exemplo:

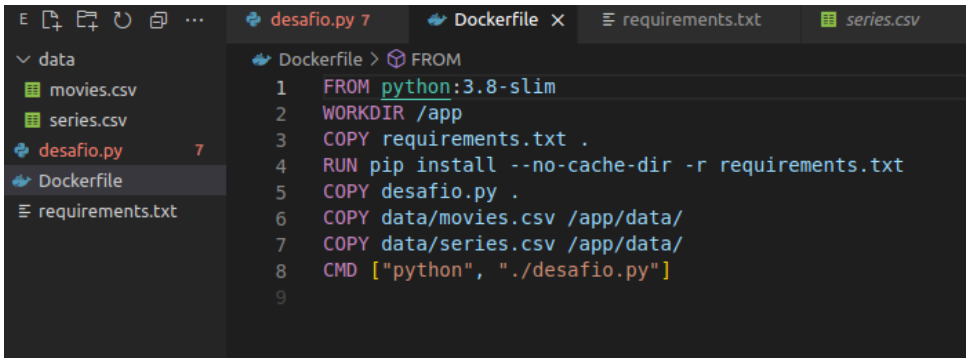
S3:\\data-lake-do-fulano\Raw\Local\CSV\Movies\2022\05\02\movies.csv

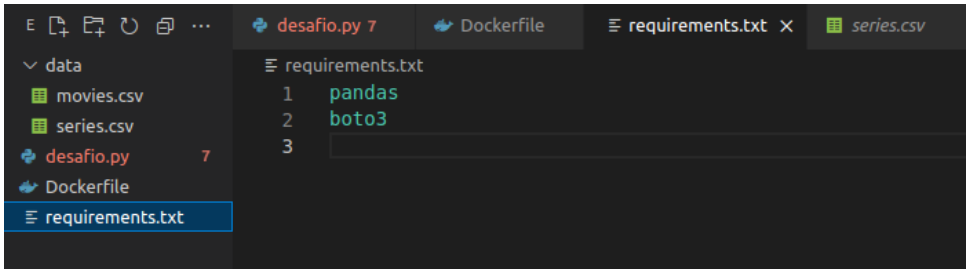
S3:\\data-lake-do-fulano\Raw\Local\CSV\Series\2022\05\02\series.csv

```python
s3 = boto3.client('s3', aws_access_key_id=aws_access_key_id, aws_secret_access_key=aws_secret_access_key)

bucket_name = 'armazemdedados'
data_processamento = datetime.now().strftime('%Y/%m/%d')

try:
    filmes_csv_bytesio = BytesIO(filmes_df.to_csv(sep='|', index=False).encode())
    series_csv_bytesio = BytesIO(series_df.to_csv(sep='|', index=False).encode())

    s3.upload_fileobj(filmes_csv_bytesio, bucket_name, 'Raw/Local/CSV/Movies/{data_processamento}/movies.csv'.format(data_processame
    s3.upload_fileobj(series_csv_bytesio, bucket_name, 'Raw/Local/CSV/series/{data_processamento}/series.csv'.format(data_processame

    print("DataFrames carregados com sucesso para o Amazon S3.")
except Exception as e:
    print(f"Erro durante o upload para o S3: {e}")
```

- Criar container Docker com um volume para armazenar os arquivos CSV e executar processo Python implementado:







- Executar localmente o container docker para realizar a carga dos dados ao S3:

```
lins@lins-Lenovo-G460:~/Downloads/Filmes+e+Series$ sudo docker build -t pulls3 -f Dockerfile .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  253.4MB
Step 1/8 : FROM python:3.8-slim
 ---> 475f7b7896d0
Step 2/8 : WORKDIR /app
 ---> Using cache
 ---> 9febaf7c5280
Step 3/8 : COPY requirements.txt .
 ---> Using cache
 ---> 18976414ed62
Step 4/8 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Using cache
 ---> db8383e450e8
Step 5/8 : COPY desafio.py .
 ---> Using cache
 ---> c1b2c833df92
Step 6/8 : COPY data/movies.csv /app/data/
 ---> bded2a7530ba
Step 7/8 : COPY data/series.csv /app/data/
 ---> 27b22edd8c9a
Step 8/8 : CMD ["python", "./desafio.py"]
 ---> Running in a5b89ce9006d
Removing intermediate container a5b89ce9006d
 ---> 0a7be66f9796
Successfully built 0a7be66f9796
Successfully tagged pulls3:latest
```

```
lins@lins-Lenovo-G460:~/Downloads/Filmes+e+Series$ sudo docker run -v $(pwd)/data:/app/data pulls3
Bucket Name: armazemdedados, Created Time: 2023-11-10 05:01:14+00:00
DataFrames carregados com sucesso para o Amazon S3.
```