



UNIVERSIDADE FEDERAL DO CEARÁ - UFC
CAMPUS CRATEÚS
CURSO DE SISTEMAS DE INFORMAÇÃO

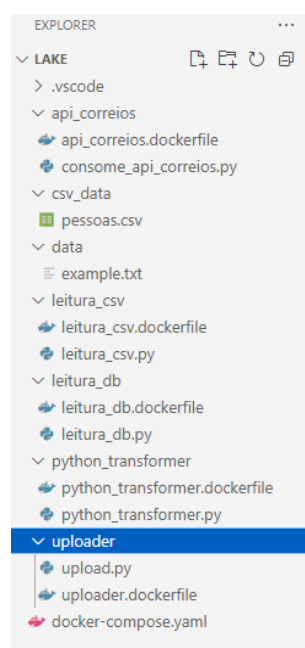
Disciplina: Inteligência do Negócio e Visualização de dados
Professor: Bruno Riccelli dos Santos Silva
Semestre: 2025.1

Prática 3 - Data lake com múltiplas ingestões

1. Introdução


Este documento apresenta um guia passo a passo para containerizar um Data lake compatível com S3 e observar algumas funcionalidades relacionadas a múltiplas ingestões. A partir da prática anterior, será possível implementar múltiplas fontes de dados e salvar em data lakes diferentes.

2. Composição de pastas e arquivos do projeto



3. Crie as pastas de acordo com o print acima e crie os respectivos arquivos

3.1 Pasta api_correios

api_correios >  api_correios.dockerfile

```
1 FROM python:3.11-slim
2 WORKDIR /app
3 COPY consome_api_correios.py .
4 RUN pip install minio requests
5 CMD ["python", "consome_api_correios.py"]
```

api_correios >  consome_api_correios.py

```
1 import os
2 import time
3 import json
4 import requests
5 from minio import Minio
6
7 endpoint = os.getenv("MINIO_ENDPOINT", "localhost:9000")
8 access_key = os.getenv("MINIO_ACCESS_KEY", "minioadmin")
9 secret_key = os.getenv("MINIO_SECRET_KEY", "minioadmin")
10
11 client = Minio(
12     endpoint,
13     access_key=access_key,
14     secret_key=secret_key,
15     secure=False
16 )
17
18 bucket = "raw"
19
20 time.sleep(5)
21
22 if not client.bucket_exists(bucket):
23     client.make_bucket(bucket)
24
25 response = requests.get("https://viacep.com.br/ws/63708825/json/")
26 cep_data = response.json()
27
28 file_path = "/tmp/cep.json"
29 with open(file_path, "w") as f:
30     json.dump(cep_data, f)
31
32 client.fput_object(bucket, "api_correios/cep.json", file_path)
33
```

3.2 pasta csv_data

```
csv_data > 📄 pessoas.csv
```

```
1 nome,idade
2 Alice,30
3 Bruno,33
4 Carla,28
5
```

3.3 pasta leitura_csv

```
leitura_csv > 📄 leitura_csv.dockerfile
```

```
1 FROM python:3.11-slim
2 WORKDIR /app
3 COPY leitura_csv.py .
4 RUN pip install minio
5 CMD ["python", "leitura_csv.py"]
6
```

```
leitura_csv > 📄 leitura_csv.py
```

```
1 import os
2 import time
3 from minio import Minio
4
5 endpoint = os.getenv("MINIO_ENDPOINT", "localhost:9000")
6 access_key = os.getenv("MINIO_ACCESS_KEY", "minioadmin")
7 secret_key = os.getenv("MINIO_SECRET_KEY", "minioadmin")
8
9 client = Minio(
10     endpoint,
11     access_key=access_key,
12     secret_key=secret_key,
13     secure=False
14 )
15
16 bucket = "raw"
17
18 time.sleep(5)
19
20 if not client.bucket_exists(bucket):
21     client.make_bucket(bucket)
22
23 file_path = "/data/pessoas.csv"
24 client.fput_object(bucket, "csv/pessoas.csv", file_path)
25 print("Arquivo enviado para raw/csv/pessoas.csv")
26
```

3.5 pasta leitura_db

leitura_db >  leitura_db.dockerfile

```
1 FROM python:3.11-slim
2 WORKDIR /app
3 COPY leitura_db.py .
4 RUN pip install minio psycopg2-binary
5 CMD ["python", "leitura_db.py"]
6
```

leitura_db >  leitura_db.py

```
1 import os
2 import time
3 import csv
4 import psycopg2
5 from minio import Minio
6
7 endpoint = os.getenv("MINIO_ENDPOINT", "localhost:9000")
8 access_key = os.getenv("MINIO_ACCESS_KEY", "minioadmin")
9 secret_key = os.getenv("MINIO_SECRET_KEY", "minioadmin")
10
11 client = Minio(
12     endpoint,
13     access_key=access_key,
14     secret_key=secret_key,
15     secure=False
16 )
17
18 bucket = "raw"
19
20 time.sleep(10) # espera postgres e minio subirem
21
22 if not client.bucket_exists(bucket):
23     client.make_bucket(bucket)
24
25 # Conecta ao banco
26 conn = psycopg2.connect(
27     host=os.getenv("DB_HOST"),
28     database=os.getenv("DB_NAME"),
29     user=os.getenv("DB_USER"),
30     password=os.getenv("DB_PASS"),
31 )
32 cur = conn.cursor()
```

```

leitura_db > leitura_db.py
33
34 cur.execute("CREATE TABLE IF NOT EXISTS clientes (id SERIAL PRIMARY KEY, nome TEXT, email TEXT);")
35 cur.execute("INSERT INTO clientes (nome, email) VALUES ('João', 'joao@exemplo.com') ON CONFLICT DO NOTHING;")
36 conn.commit()
37
38 cur.execute("SELECT * FROM clientes")
39 rows = cur.fetchall()
40
41 file_path = "/tmp/clientes.csv"
42 with open(file_path, "w", newline="") as f:
43     writer = csv.writer(f)
44     writer.writerow(["id", "nome", "email"])
45     writer.writerows(rows)
46
47 client.fput_object(bucket, "db/clientes.csv", file_path)
48 print("Arquivo enviado para raw/db/clientes.csv")
49

```

3.6 pasta python_transformer

```

python_transformer > python_transformer.dockerfile
1 FROM python:3.11-slim
2 WORKDIR /app
3 COPY python_transformer.py .
4 RUN pip install minio pandas
5 CMD ["python", "python_transformer.py"]
6

```

python_transformer > python_transformer.py

```
1  import os
2  import time
3  import io
4  import pandas as pd
5  from minio import Minio
6
7  endpoint = os.getenv("MINIO_ENDPOINT", "localhost:9000")
8  access_key = os.getenv("MINIO_ACCESS_KEY", "minioadmin")
9  secret_key = os.getenv("MINIO_SECRET_KEY", "minioadmin")
10
11  client = Minio(
12      endpoint,
13      access_key=access_key,
14      secret_key=secret_key,
15      secure=False
16  )
17  bucket_raw = "raw"
18  bucket_custom = "custom"
19
20  time.sleep(10)
21
22  if not client.bucket_exists(bucket_custom):
23      client.make_bucket(bucket_custom)
24
25  objects = client.list_objects(bucket_raw, recursive=True)
26  for obj in objects:
27      if obj.object_name.endswith(".csv"):
28          response = client.get_object(bucket_raw, obj.object_name)
29          df = pd.read_csv(response)
30          df.columns = [col.upper() for col in df.columns]
31          out_csv = df.to_csv(index=False).encode("utf-8")
32          out_name = obj.object_name.replace("raw/", "")
33          client.put_object(bucket_custom, out_name, io.BytesIO(out_csv), len(out_csv))
34          print(f"Transformado e salvo: {out_name}")
35
```

4. Docker compose

 docker-compose.yml

```
2
3  services:
4      minio:
5          image: minio/minio:latest
6          container_name: minio
7          environment:
8              MINIO_ROOT_USER: minioadmin
9              MINIO_ROOT_PASSWORD: minioadmin
10         volumes:
11             - minio_data:/data
12         ports:
13             - "9000:9000"
14             - "9001:9001"          # console UI
15         command: server /data --console-address ":9001"
16
17     postgres:
18         image: postgres:13
19         container_name: postgres
20         environment:
21             POSTGRES_USER: useradmin
22             POSTGRES_PASSWORD: useradmin
23             POSTGRES_DB: meudb
24         volumes:
25             - pg_data:/var/lib/postgresql/data
26
27     api_correios:
28         build:
29             context: ./api_correios/
30             dockerfile: api_correios.dockerfile
31         container_name: api_correios
32         depends_on: [minio]
33         environment:
34             MINIO_ENDPOINT: minio:9000
35             MINIO_ACCESS_KEY: minioadmin
36             MINIO_SECRET_KEY: minioadmin
```

```

38  leitura_db:
39  build:
40      context: ./leitura_db/
41      dockerfile: leitura_db.dockerfile
42  container_name: leitura_db
43  depends_on: [postgres, minio]
44  environment:
45      MINIO_ENDPOINT: minio:9000
46      DB_HOST: postgres
47      DB_USER: useradmin
48      DB_PASS: useradmin
49      DB_NAME: meudb
50
51  leitura_csv:
52  build:
53      context: ./leitura_csv/
54      dockerfile: leitura_csv.dockerfile
55  container_name: leitura_csv
56  depends_on: [minio]
57  volumes:
58      - ./csv_data:/data
59  environment:
60      MINIO_ENDPOINT: minio:9000
61      MINIO_ACCESS_KEY: minioadmin
62      MINIO_SECRET_KEY: minioadmin
63
64  python_transformer:
65  build:
66      context: ./python_transformer/
67      dockerfile: python_transformer.dockerfile
68  depends_on: [minio, postgres, leitura_csv, leitura_db, api_correios ]
69  environment:
70      MINIO_ENDPOINT: minio:9000
71      MINIO_ACCESS_KEY: minioadmin
72      MINIO_SECRET_KEY: minioadmin
73
74  volumes:
75      minio_data:
76      pg_data:
77

```

5. execute o comando: `docker compose down -v`

6. Em seguida, `docker compose up --build`

7. Por fim, teremos dois buckets (um com dados brutos e outro com dados transformados) criados a partir de diferentes fontes de ingestão: csv, API e banco de dados relacional.