

# Breweries Case

BEES Data Engineering - teste técnico

# SUMÁRIO

1. Introdução.....	4
1. Eventos.....	4
2. Gerenciamento de Repositório e Versionamento .....	5
4. Fluxo de Execução/Processamento .....	6
4.1 Fluxo Macro .....	6
4.2 Fluxo Detalhado .....	7
4.3. Instalação do Docker .....	8
5. O projeto: Python .....	11
6. Funções .....	11
6.1 Cria Diretorio .....	11
6.2 buscar_e_salvar_dados_brewery .....	11
6.3 transformar_dados_para_silver .....	11
6.4 transformar_dados_para_gold .....	12
7 Inicia o Processo .....	12

**Histórico de Revisões:**

<b>Data</b>	<b>Versã o</b>	<b>Descrição</b>	<b>Autor</b>
23/07/2024	1.0	Base da documentação	Rodrigo Henrique

## 1. Introdução

Este Projeto tem por finalidade, o desenvolvimento de uma rotina para análise, tratamento e simulação de um datalake seguindo a arquitetura medallion

## 1. Eventos

Os eventos narrados neste documento são;

**1.1. Docker:** Docker é uma plataforma de contêineres que permite aos desenvolvedores criar, implantar e executar aplicativos em contêineres. Um contêiner é uma unidade de software que empacota código e todas as suas dependências para que o aplicativo seja executado rapidamente e de maneira confiável de um ambiente computacional para outro. Contêineres são leves, portáteis e consistem em tudo que é necessário para executar um aplicativo, incluindo o sistema operacional, bibliotecas, dependências e o próprio código. Eles compartilham o kernel do sistema operacional do host, o que os torna mais eficientes em termos de recursos em comparação com máquinas virtuais (VMs).

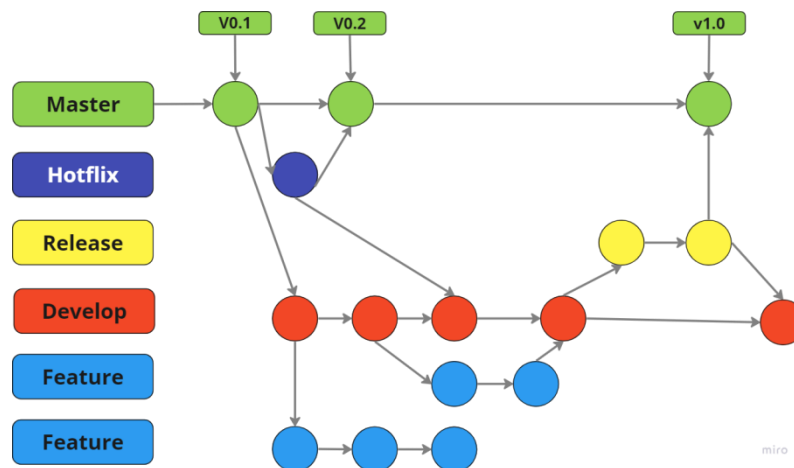
**1.2. Airflow:** Apache Airflow é uma plataforma de código aberto usada para criar, agendar e monitorar fluxos de trabalho programáticos. É amplamente utilizado para automação de pipelines de dados e ETL (Extração, Transformação e Carga de dados). Airflow fornece uma interface web para visualizar, gerenciar e monitorar os DAGs. A interface permite observar a execução das tarefas, ver logs, re-executar tarefas, e muito mais.

## 2. Gerenciamento de Repositório e Versionamento

O GitLab é uma plataforma de desenvolvimento de software que fornece um conjunto abrangente de ferramentas para gerenciar todo o ciclo de vida de desenvolvimento de software. Ele oferece recursos de controle de versão usando o Git, além de gerenciamento de projetos, rastreamento de problemas (issue tracking), integração contínua (CI/CD), gestão de requisitos, entre outros.

O GitLab permite que equipes de desenvolvimento colaborem em projetos de software, facilitando a colaboração, revisão de código, automação de processos de build e deploy, e oferecendo uma interface centralizada para diversos aspectos do desenvolvimento de software. Ele pode ser utilizado tanto como uma plataforma SaaS (Software as a Service) quanto instalado em servidores próprios (on-premise), oferecendo flexibilidade para equipes com diferentes necessidades e preferências de infraestrutura.

Figura 1: Modelo de fluxo de commit e versionamento

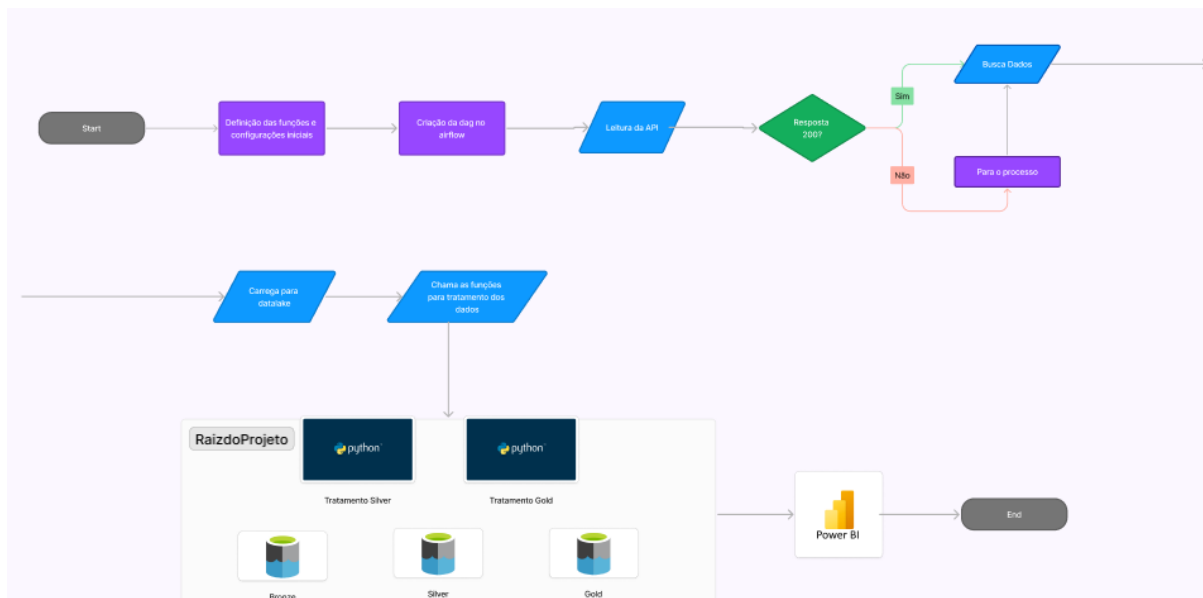


## 4. Fluxo de Execução/Processamento

### 4.1 Fluxo Macro

O processo de desenvolvimento do fluxo foi estruturado considerando as bases de dados que serão utilizadas, desde a elaboração até a finalização do ETL. Esse resultado é posteriormente empregado pelo Power bi para visualização dos dados.

Figura 2: Fluxo macro do projeto



## 4.2 Fluxo Detalhado

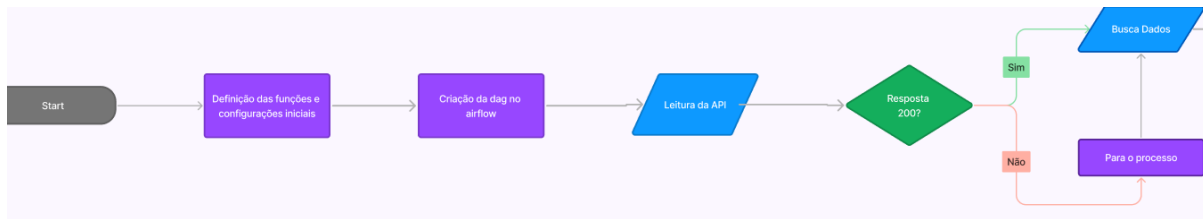


Figura 2.1: Fluxo macro do projeto



Figura 2.2: Fluxo macro do projeto

A base de dados, proveniente da API, que serviram de entradas para o python - pandas realizar a leitura dos dados e o respectivo tratamento. As tabelas da sigla "**bronze**", "**silver**" e "**gold**" são fundamentais nesse processo, contribuindo para a integralidade e precisão dos dados utilizados.

Durante a etapa de processamento, criamos três dataframes sendo eles: bronze, silver e gold ele contém todos os dados referentes a informações de vendas por tipo de cervejaria, estado;

O primeiro dataframe é o dataframe de referente a camada bronze que contém as colunas;

```
{ "id": "5128df48-79fc-4f0f-8b52-d06be54d0cec", "name": "(405) Brewing Co", "brewery_type": "micro", "address_1": "1716 Topeka St", "address_2": null, "address_3": null, "city": "Norman", "state_province": "Oklahoma", "postal_code": "73069-8224", "country": "United States", "longitude": "-97.46818222", "latitude": "35.25738891", "phone": "4058160490", "website_url": "http://www.405brewing.com", "state": "Oklahoma", "street": "1716 Topeka St" }
```

O segundo dataframe é o dataframe referente a camada **silver**, houve alguns tratamentos visando a melhor qualidade do dado

```

root
|-- id: string (nullable = true)
|-- nome: string (nullable = true)
|-- tipo_cervejaria: string (nullable = true)
|-- endereco_principal: string (nullable = true)
|-- endereco_secundario: string (nullable = true)
|-- endereco_alternativo: string (nullable = true)
|-- cidade: string (nullable = true)
|-- provincia: string (nullable = true)
|-- cep: string (nullable = true)
|-- pais: string (nullable = true)
|-- longitude: float (nullable = true)
|-- latitude: float (nullable = true)
|-- telefone: string (nullable = true)
|-- site: string (nullable = true)
|-- rua: string (nullable = true)
|-- website_url_validacao: boolean (nullable = true)
|-- indicador_seguranca_site: boolean (nullable = true)

```

E o terceiro dataframe é referente a camada gold com as agregações

```

root
|-- estado: string (nullable = true)
|-- tipo_cervejaria: string (nullable = true)
|-- count: long (nullable = true)

```

### 4.3. Instalação do Docker

1 – Verifique os requisitos mínimos do sistema;



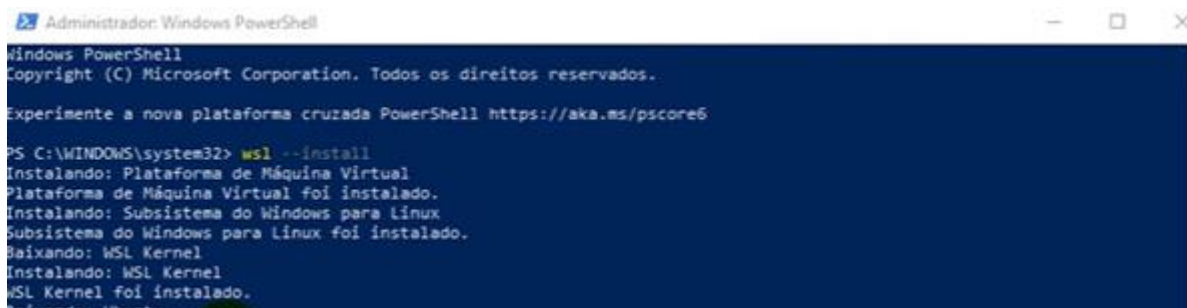
- WSL version 1.1.3.0 or later.
- Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
- Windows 10 64-bit:
  - We recommend Home or Pro 22H2 (build 19045) or higher, or Enterprise or Education 22H2 (build 19045) or higher.
  - Minimum required is Home or Pro 21H2 (build 19044) or higher, or Enterprise or Education 21H2 (build 19044) or higher

2 - Baixa o exe. do site oficial docker: <https://docs.docker.com/get-docker/>

3 – Instale o WSL:

3.1 - Abra seu PowerShell (Como admin)

3.2 - Cole o seguinte comando: **wsl --install**



```

Administrador: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

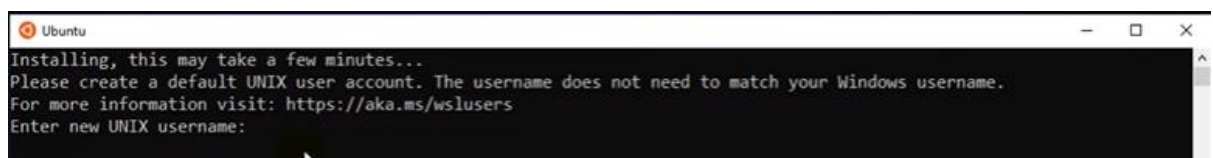
Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> wsl --install
Instalando: Plataforma de Máquina Virtual
Plataforma de Máquina Virtual foi instalado.
Instalando: Subsistema do Windows para Linux
Subsistema do Windows para Linux foi instalado.
Baixando: WSL Kernel
Instalando: WSL Kernel
WSL Kernel foi instalado.
Baixando: Ubuntu
  
```

Figura 3: PowerShell comando wsl

3.3 - Abra o Ubuntu: **W + S (digite na barra de pesquisa Ubuntu)**

3.4 - Digite usuário e senha de sua escolha:



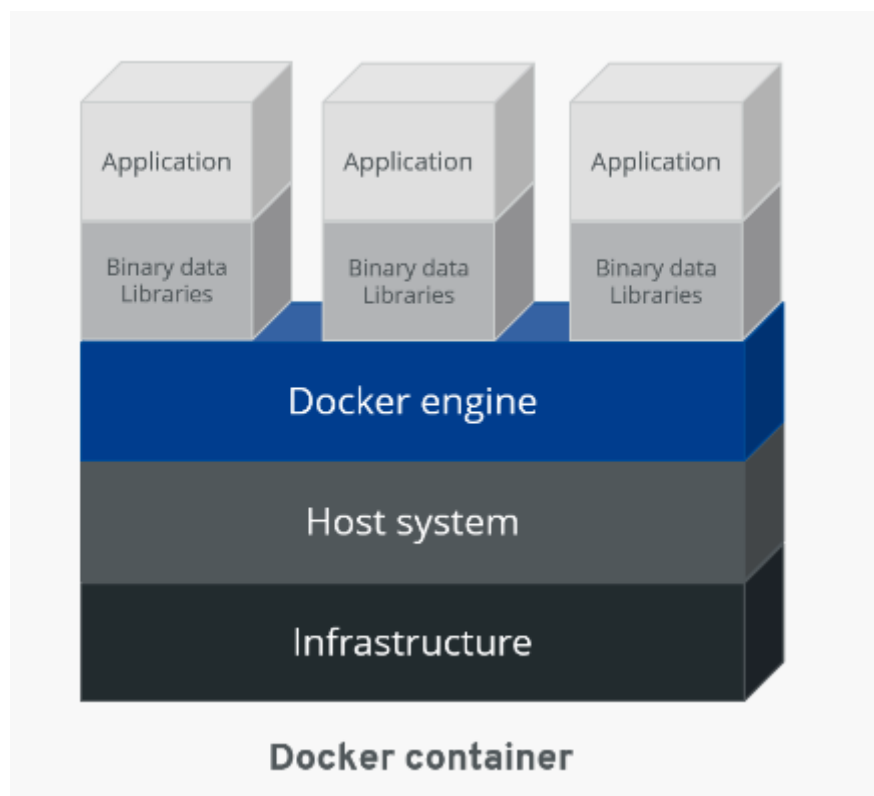
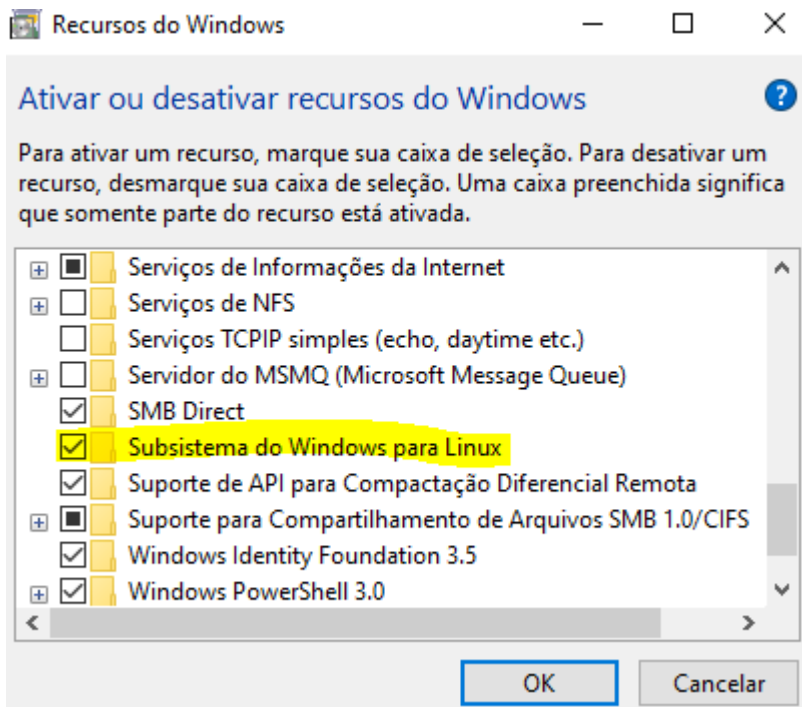
```

Ubuntu
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username:
  
```

4 – Clique duas vezes em Docker Desktop Installer.exe para executar o instalador. Por padrão, Docker Desktop é instalado em C:\Program Files\Docker\Docker.

5 – Deixe as duas opções durante a instalação preenchidas

6 – Verifique se o recurso está ativo no Windows:



## 5. O projeto: Python

O projeto foi desenvolvido utilizando Python e Pandas, adotando uma abordagem orientada a funções para a criação e utilização dos códigos, visando a eficiência e modularidade na implementação e análise de dados. Abaixo apresentamos a documentação das funções utilizadas.

## 6. Funções

### 6.1 Cria Diretorio

Cria um diretório caso ele não exista.

### 6.2 buscar\_e\_salvar\_dados\_brewery

Busca dados de cervejarias a partir de uma API, salva-os em formato JSON em uma zona de armazenamento local.

#### Operações:

- Define caminhos para o datalake e a landing zone.
- Cria os diretórios necessários se não existirem.
- Faz uma requisição inicial à API para determinar o número total de registros.
- Faz múltiplas requisições para buscar todos os dados, paginando conforme necessário.
- Salva os dados recuperados em um arquivo JSON na landing zone.

#### Exceções:

- Se houver erros nas requisições à API, a função imprime mensagens de erro e termina a execução.

### 6.3 transformar\_dados\_para\_silver

Transforma os dados da landing zone, aplica limpezas e validações, e salva os dados transformados na silver zone.

#### Operações:

- Define caminhos para o datalake, landing zone, e silver zone.
- Cria os diretórios necessários se não existirem.
- Lê os dados JSON da landing zone.
- Renomeia colunas para um formato padronizado.
- Aplica transformações, como limpeza de texto e preenchimento de valores nulos.
- Valida URLs dos sites das cervejarias.
- Agrupa as colunas de estado e cidade para reduzir o número de partições.
- Salva os dados transformados em arquivos Parquet na silver zone, particionados por estado e cidade.

## 6.4 transformar\_dados\_para\_gold

Agrega os dados da silver zone e salva os resultados na gold zone.

### Operações:

- Define caminhos para o datalake, silver zone, e gold zone.
- Cria os diretórios necessários se não existirem.
- Lê os dados Parquet da silver zone.
- Realiza uma agregação para contar o número de cervejarias por tipo e localização.
- Salva os dados agregados em um arquivo Parquet na gold zone.

## 7 Inicia o Processo

Neste momento vamos iniciar o processo:

### Configuração do Ambiente:

- Inicie seu docker desktop para que a engine seja ativada
- Navega até a pasta do projeto (airflow\_Case\_Beers) digite: **docker-compose up -d**
- Container será criado subindo o airflow**

### Preparação de Dados:

- Navegue ate <http://localhost:8080/>
- Entre no airflow web (user: airflow) - (senha: airflow)
- Dispare a trigger referente a dag criada

### Finalização:

- Verifique o log no airflow referente a cada tarefa

