

Documentação do Processo

Passo 1: Instalação do Docker

Instalar o Docker na máquina local para poder executar a aplicação Flask em um contêiner. O Docker permite empacotar e isolar a aplicação e suas dependências em uma imagem que pode ser executada em qualquer ambiente que tenha o Docker instalado. A instalação varia conforme o sistema operacional:

Windows/macOS: Usamos o Docker Desktop, que pode ser baixado e instalado a partir do site oficial do Docker.

Linux: Executar os comandos abaixo para instalar o Docker no Ubuntu/Debian:

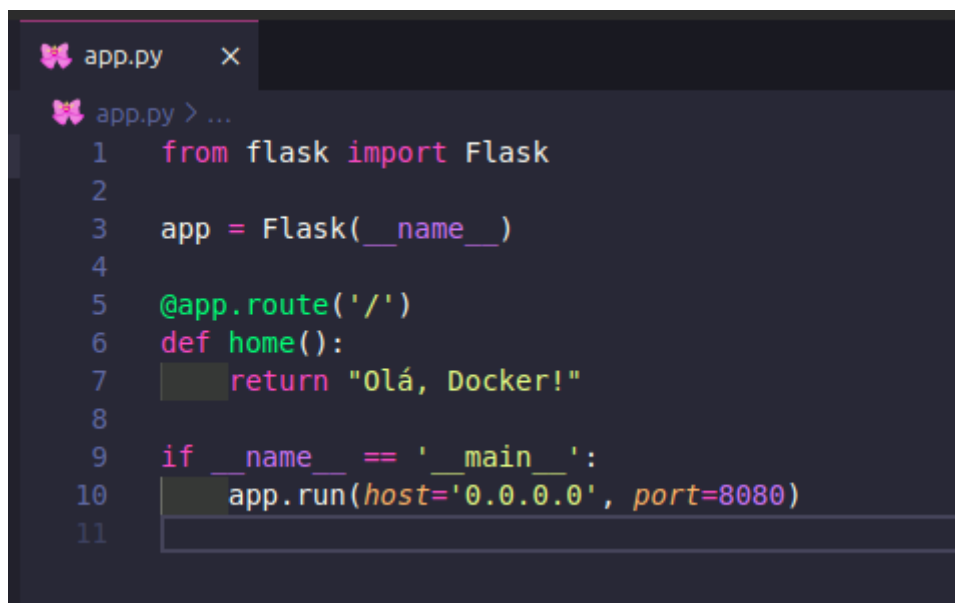
```
sudo apt-get update    sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Após a instalação, verificar se o Docker foi instalado corretamente com:

`docker --version` *Esse comando retorna a versão instalada do Docker, garantindo que está pronto para uso.*

Passo 2: Criação da Aplicação Flask

Criei um diretório onde os arquivos do projeto serão armazenados e *após* uma aplicação Flask simples que será containerizada com o Docker

A screenshot of a code editor with a dark theme. The editor has a tab at the top labeled 'app.py' with a butterfly icon and a close button. Below the tab, the code for 'app.py' is displayed. The code is a simple Flask application that imports Flask, creates an app, defines a home route that returns 'Olá, Docker!', and runs the app on host '0.0.0.0' and port 8080. Line numbers 1 through 11 are visible on the left side of the code block.

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def home():
7      return "Olá, Docker!"
8
9  if __name__ == '__main__':
10     app.run(host='0.0.0.0', port=8080)
11
```

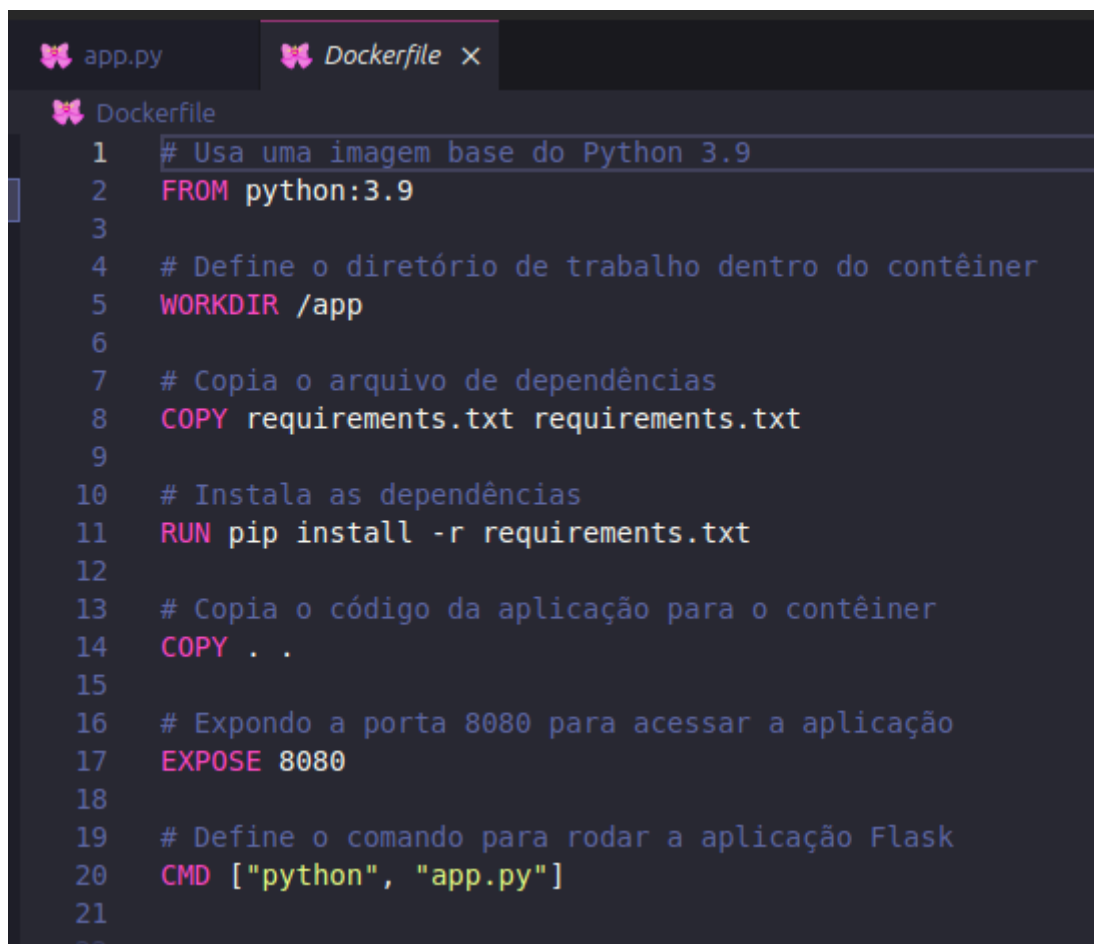
Explicação:

- **Flask**: É o framework web que estamos usando para criar uma aplicação simples.
- `@app.route(' / ')`: Define a rota principal da aplicação, onde a função `home` retorna a string “Olá, Docker!” quando a página principal for acessada.
- `app.run()`: Faz a aplicação rodar no endereço `0.0.0.0` (ou seja, acessível de qualquer IP) na porta `8080`.

Instalação do Flask *pip install flask* ou *sudo apt install python3-flask*

Passo 3: Criação do Dockerfile

O **Dockerfile** é um arquivo contendo as instruções necessárias para criar a imagem Docker da aplicação.



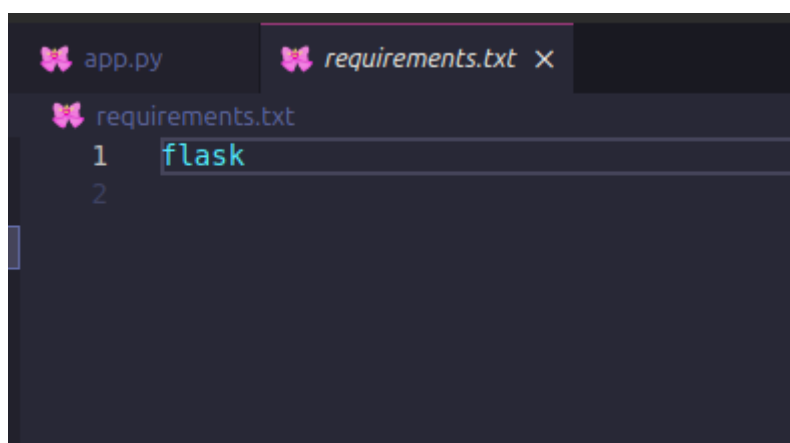
```
app.py  Dockerfile x
Dockerfile
1  # Usa uma imagem base do Python 3.9
2  FROM python:3.9
3
4  # Define o diretório de trabalho dentro do contêiner
5  WORKDIR /app
6
7  # Copia o arquivo de dependências
8  COPY requirements.txt requirements.txt
9
10 # Instala as dependências
11 RUN pip install -r requirements.txt
12
13 # Copia o código da aplicação para o contêiner
14 COPY . .
15
16 # Expondo a porta 8080 para acessar a aplicação
17 EXPOSE 8080
18
19 # Define o comando para rodar a aplicação Flask
20 CMD ["python", "app.py"]
21
22
```

Explicação:

- **FROM python:3.9**: Define a imagem base para o contêiner, que neste caso é o Python 3.9. Essa imagem já vem configurada com o Python, o que facilita a execução de aplicações Python.
- **WORKDIR /app**: Define **/app** como o diretório de trabalho dentro do contêiner, onde os arquivos da aplicação serão armazenados.
- **COPY requirements.txt**: Copia o arquivo **requirements.txt** para o contêiner, que contém as dependências da aplicação (nesse caso, o Flask).
- **RUN pip install -r requirements.txt**: Executa o comando para instalar as dependências listadas no **requirements.txt** usando **pip**.
- **COPY . .**: Copia todos os arquivos do diretório atual para o diretório de trabalho no contêiner.
- **EXPOSE 8080**: Informa ao Docker que a aplicação irá rodar na porta 8080.
- **CMD ["python", "app.py"]**: Define o comando que será executado quando o contêiner iniciar, que no caso é **python app.py**, para rodar a aplicação Flask.

Passo 4: Criação do Arquivo **requirements.txt**

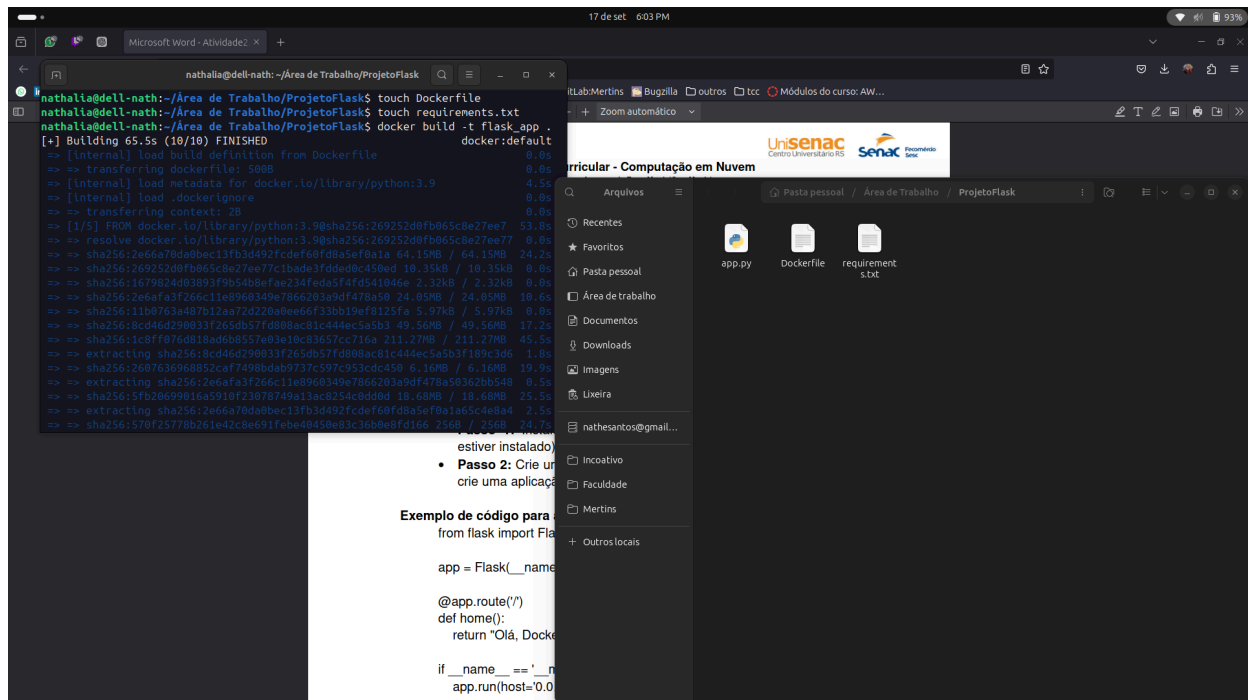
Para garantir que todas as dependências sejam instaladas dentro do contêiner, criamos o arquivo **requirements.txt** e adicionamos o Flask como a única dependência. Esse arquivo é usado pelo Docker para instalar o Flask no ambiente do contêiner.



```
app.py  requirements.txt X
requirements.txt
1 flask
2
```

Passo 5: Construção da Imagem Docker

Com o **Dockerfile** pronto, construí a imagem Docker que contém nossa aplicação Flask.



Passo 6: Executar a Aplicação no Contêiner

Com a imagem criada, podemos executar a aplicação dentro de um contêiner.

Comando de Execução do Contêiner: Usamos o seguinte comando para rodar o contêiner e mapear a porta 8080:

```
docker run -p 8080:8080 flask_app
```

1. Explicação:

- **docker run**: Executa um novo contêiner baseado na imagem especificada.
- **-p 8080:8080**: Mapeia a porta 8080 do host (máquina local) para a porta 8080 do contêiner, permitindo que a aplicação seja acessada localmente via navegador.
- **flask_app**: Nome da imagem que queremos executar.

Passo 7: Verificação da Aplicação

Para verificar se a aplicação está rodando corretamente, abrir um navegador e acessar <http://localhost:8080>. Se tudo foi configurado corretamente, a seguinte mensagem aparecerá:

Olá, Docker!

