Relatório de Desenvolvimento de API Flask com Docker

Lucas

11 de fevereiro de 2025

Resumo

Este relatório descreve o desenvolvimento de uma API utilizando o framework Flask em Python, configurada e executada em um ambiente Docker. A API foi projetada para oferecer uma resposta JSON simples, e o processo de integração com o Docker foi detalhado, com o intuito de garantir um ambiente isolado e eficiente para a execução da aplicação.

1 Introdução

O presente trabalho tem como objetivo apresentar o desenvolvimento de uma API simples utilizando o framework Flask, seguido da configuração de um ambiente Docker para a execução da aplicação. A API possui uma única rota GET, `"/"`, que retorna uma resposta em formato JSON. Este relatório aborda o processo de desenvolvimento da API, as configurações realizadas no Flask e a integração com o Docker para garantir a execução em um ambiente controlado.

2 Desenvolvimento da API

A aplicação foi desenvolvida a partir de um tutorial básico que ensina a criação de uma API simples com Flask. A estrutura da API é composta por uma única rota, `"/"`, que responde a requisições GET com um JSON contendo informações sobre o status da API. O código da aplicação é apresentado a seguir:

```
from flask import Flask, jsonify
import os
```

```
app = Flask(__name__)

@app.route('/', methods=['GET'])
def home():
    return jsonify({
        "success": "true",
        "message": "Seja Bem-Vindo(a) ao mundo Docker!",
        "version": "1.0.0"
    })

if __name__ == '__main__':
    port = int(os.getenv('PORT', 5000))
    print(f"Aplica  o executando na porta: {port}")
    app.run(host='0.0.0.0', port=port)
```

A resposta gerada pela API é um JSON contendo três campos principais:

- `success`: Indica o sucesso da requisição.
- `message`: Mensagem de boas-vindas.
- `version`: Versão atual da API.

3 Configuração do Docker

A configuração do Docker foi realizada para garantir que a aplicação fosse executada em um ambiente isolado e de fácil replicação. O processo de configuração consistiu nas etapas seguintes:

3.1 Instalação das dependências

As dependências do projeto foram registradas no arquivo `requirements.txt` com o comando:

```
pip freeze > requirements.txt
```

3.2 Criação do Dockerfile

O `Dockerfile` foi configurado para criar a imagem da aplicação. Abaixo está o conteúdo do arquivo `Dockerfile` utilizado:

```
FROM python:3.10-slim

WORKDIR /app

COPY . .

RUN pip install --upgrade pip

RUN pip install --no-cache-dir -r requirements.txt

EXPOSE 5000

CMD ["gunicorn", "app:app", "--bind", "0.0.0.0:5000",
   "--workers", "3"]
```

O `Dockerfile` realiza as seguintes ações:

- Utiliza a imagem `python:3.10-slim` como base.
- Define o diretório de trabalho como `/app`.
- Copia os arquivos da aplicação para o container.
- Atualiza o `pip` e instala as dependências listadas no `requirements.txt`.
- Expondo a porta 5000 para a aplicação Flask.
- Configura o uso do `Gunicorn` como servidor, com 3 workers.

3.3 Configuração do Docker Compose

Foi utilizado o `docker-compose` para orquestrar o ambiente, facilitando o gerenciamento de múltiplos containers caso seja eventualmente necessário. O arquivo `docker-compose.yml` foi configurado da seguinte forma:

```
services:
  flask_app:
    build: .
    ports:
      - "5000:5000"
    environment:
      - PORT=5000
```

A aplicação foi iniciada com o comando:

```
docker-compose up --build
```

Este comando constrói e inicia os containers definidos no `docker-compose.yml`.

4 Resultados Obtidos

Após a execução do `docker-compose`, a aplicação Flask foi iniciada corretamente. Acessando o endereço `http://localhost:5000/`, foi possível obter a seguinte resposta JSON:

```
{
    "message": "Seja Bem-Vindo(a) ao mundo Docker!",
  "success": true,
  "version": "1.0.0"
}
```

Esse retorno confirma que a API está funcionando conforme esperado.

5 Conclusão

O desenvolvimento da API utilizando Flask foi concluído com sucesso, e a integração com o Docker foi realizada de maneira eficaz, proporcionando um ambiente isolado para a execução da aplicação. A API responde corretamente à requisição `"/"` com a mensagem e versão da aplicação, como esperado.