Como efetuar o deploy de uma aplicação usando o Docker e o Heroku.

Deploy.md

Heroku + Docker + Streamlit = ❤️

Neste tutorial iremos aprender a realizar a implantação de um serviço que disponibilizará a nossa aplicação para outras pessoas. Utilizaremos o Docker para criação do ambiente necessário e o Heroku para disponibilização do serviço via aplicação web, que será criada através do streamlit.

Antes de tudo, preciso que você entenda alguns conceitos, como containers e virtualização, deploy, heroku, etc. Para facilitar, confira o seguintes vídeos:

Pronto, agora podemos iniciar o nosso tutorial. Vamos começar com a instalação do Docker.

Como instalar o Docker no Ubuntu 20.04

Pré-requisitos

Para seguir este tutorial, você precisará apenas do Ubuntu 20.04 instalado, seja através da instalação *standalone* ou através do WSL, no windows.

Passo #1: Instalando o Docker

Iremos instalar o Docker através do repositório oficial. Para isso você precisa adicionar um novo pacote de dados, uma chave GPG para assegurar que o download é oficial e então instalar o pacote.

Primeiro, vamos atualizar os pacotes existentes em sua máquina:

```
sudo apt update && sudo apt upgrade
```

Depois disso iremos instalar algumas dependências através do comando:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-com
```

Então iremos adicionar a seu sistema a chave GPG do repositório oficial do Docker:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Agora adicionaremos o repositório do Docker as fontes APT:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubunt
```

Isso também atualizará nosso banco de dados de pacotes com os pacotes Docker do repositório recém-adicionado.

Agora, finalmente instalaremos o Docker:

```
sudo apt install docker-ce
```

O Docker agora deve estar instalado, Verifique se o processo está em está em execução:

```
sudo service docker status
```

Se o docker não estiver em funcionamento, digite:

```
sudo service docker start
```

e verifique novamente o seu status através do comando usado anteriormente:

```
sudo service docker status
```

A saída deve ser semelhante à seguinte, mostrando que o serviço está ativo e em execução:

```
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor
preset: enabled)
     Active: active (running) since Tue 2020-05-19 17:00:41 UTC; 17s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 24321 (dockerd)
      Tasks: 8
     Memory: 46.4M
     CGroup: /system.slice/docker.service
             └─24321 /usr/bin/dockerd -H fd:// --
containerd=/run/containerd/containerd.sock
```

> Para usuários windows, a saída deve ser algo como: `* Docker is runing`

Passo #2: Executando o Docker sem usar o comando sudo

Por padrão, o comando `docker` só pode ser executado pelo usuário `root` ou por um usuário do grupo `docker`, que é criado automaticamente durante o processo de instalação do Docker. Se você tentar executar o comando `docker` sem prefixá-lo com `sudo` ou sem estar no grupo `docker`, obterá uma saída como esta:

```
docker: Cannot connect to the Docker daemon. Is the docker daemon running on
this host?.
See 'docker run --help'.
```

Se você quiser evitar digitar sudo sempre que executar o comando docker, adicione seu nome de usuário ao grupo docker:

```
sudo usermod -aG docker ${USER}
```

Para aplicar a nova associação ao grupo, digite o seguinte comando:

```
su - ${USER}
```

Parabéns! Se você seguiu corretamente todos os passos, agora você deverá ter instalado em sua máquina o Docker. O próximo passo é preparar o Heroku.

Preparar o heroku

Nessa etapa do tutorial, você irá criar uma conta no heroku e instalar o heroku CLI.

Passo #1: Criar conta Heroku

Você precisa primeiro criar uma conta na plataforma Heroku. Para isso acesse a URL https://signup.heroku.com/ e siga as instruções.

Passo #2: Instalar o heroku CLI

Com a conta criada, iremos agora instalar o heroku CLI utilizando o seguinte comando:

```
curl https://cli-assets.heroku.com/install.sh | sh
```

Essa ferramenta é necessária para que possamos criar e manipular projetos heroku via linha de comando.

Para verificar se a instalação foi completada, utilize:

```
heroku --version
```

A saída desde comando deve ser algo como `heroku/7.0.0 (darwin-x64) node-v8.0.0`. Caso contrário, tente instalar novamente.

Com o heroku CLI instalado, precisamos efetuar o login através do comando:

```
heroku login
```

Depois de efetuado o login, podemos criar nossa aplicação.

Criando nossa aplicação

Finalmente iremos criar nossa aplicação de exemplo. Para isto, vá até o repositório https://github.com/omadson/project-template e clique em "Use this template" e crie um repositório a partir do template do projeto. Clone para sua máquina, vá até o repositório, instale as dependências através do comando

```
poetry install
```

ative o ambiente virtual usando

```
poetry shell
```

Neste repositório precisamos criar os arquivos `heroku.yml`, com as configurações de nossa aplicação heroku, `Dockerfile` com as configurações da imagem que será criada, `.dockerignore`, contendo os diretórios e arquivos que serão ignorados na criação da imagem e `app.py`, com a nossa aplicação. Vamos então dividir esta etapa em alguns passos.

Passo #1: "Olá, mundo" no Streamlit

De acordo com a documentação oficial,

> O Streamlit é uma biblioteca Python de código aberto que facilita a criação e o compartilhamento de aplicativos da Web personalizados e bonitos para aprendizado de máquina e ciência de dados. Em apenas alguns minutos, você pode criar e implantar aplicativos de dados poderosos. Então vamos começar!

Iremos criar uma aplicação simples utilizando o Streamlit. Ela conterá apenas um botão que ativará a exibição de um texto. Para isso precisamos primeiro instalar o seu pacote:

```
poetry add streamlit
```

Depois disso temos que criar o arquivo `app.py` com o seguinte conteúdo:

```
import streamlit as st


if st.button("Olá"):
    st.markdown("**mundo!**")
```

Podemos executar nossa aplicação através do comando (lembre-se de ativar o shell do poetry utilizando `poetry shell`):

```
streamlit run app.py
```

Note que o streamlit irá abrir seu navegador já com a aplicação em funcionamento.

Para uma descrição completa de uso do streamlit, acesse a documentação oficial.

A aplicação, neste caso, está funcionando diretamente no seu computador. No entanto precisamos que a aplicação funcione em um ambiente de produção, que consiga ser disponibilizado para outras pessoas. Para isso utilizaremos o heroku, com o docker como intermediário.

Passo #2: Criar a imagem Docker

Neste passo, iremos criar uma imagem Docker. Tal imagem será utilizada pelo heroku para criação do contêiner que hospedará nossa aplicação. O uso de contêineres é uma prática recomendada para uma fase de implantação, pois cada contêiner tem seu software, bibliotecas e arquivos de configuração. Além disso, a utilização de contêineres assegura que a aplicação irá funcionar em qualquer ambiente.

Começaremos com a criação de um arquivo chamado `Dockerfile` que deverá conter o seguinte conteúdo:

```
FROM python:3.8-slim

RUN mkdir /app
COPY . /app
WORKDIR /app

RUN pip install --upgrade pip && pip install -r requirements.txt

CMD streamlit run app.py --server.port $PORT
```

Esse arquivo informa como o Docker irá criar a imagem que servirá a sua aplicação. Além desse arquivo, você deverá criar o arquivo `.dockerignore`, que informará quais arquivos e pastas serão ignorados no momento de criação do container. No nosso caso, este arquivo deverá conter o seguinte conteúdo:

```
__pycache__
*.pyc
*.pyo
*.pyd
.Python
env
pip-log.txt
pip-delete-this-directory.txt
.tox
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
*.log
.git
.mypy_cache
.pytest_cache
.hypothesis
docs/
data/
notebooks/
references/
pyproject.toml
mkdocs.yml
poetry.lock
tasks.py
```

> Informações sobre a lista completa de comandos do Docker podem ser encontradas na documentação do Docker.

Com os arquivos criados, podemos efetuar o *build* da nossa imagem. Mas antes, precisamos criar o arquivo `requirements.txt` através do comando:

```
poetry export -o requirements.txt --without-hashes
```

Agora sim, criaremos nossa imagem através do seguinte comando:

```
docker build -t my-image .
```

Temos então uma imagem local que rodará a nossa aplicação utilizando o seguinte comando:

```
docker run -d --rm -e PORT=8000 -p 8000:8000 --name my-container my-image
```

A saída deste comando é somente o código do container criado.

Acesse o endereço `http://localhost:8000` pelo seu navegador preferido e veja a aplicação em funcionamento, agora em um container docker.

O seu contêiner `my-container` estará rodando em segundo plano. Para finalizá-lo, utilize o comando

```
docker stop my-container
```

Passo #3: Criar aplicação heroku

Agora só precisamos criar a nossa aplicação heroku. Com o heroku CLI essa tarefa se torna muito simples, precisando apenas criar o arquivo de configuração `heroku.yml`, com o seguinte conteúdo:

```
build:
  docker:
    web: Dockerfile
```

Ele contém as informações necessárias para o heroku identificar que queremos efetuar o deploy usando o Dockerfile criado.

Depois disso precisamos criar nossa aplicação usando:

```
heroku create
```

Definir o uso de containers no heroku (que por padrão utiliza Dynos):

```
heroku stack:set container
```

Guardar as mudanças em nossa aplicação através do git:

```
git add -A && git commit -m "Configura aplicação."
```

Enviar mudanças para o repositório do heroku:

```
git push heroku master
```

E finalmente utilizar o seguinte comando para abrir sua aplicação:

```
heroku open
```

Note que o endereço foi criado automaticamente. Esse endereço poderá ser compartilhado com qualquer pessoa para acesso a sua aplicação.

> Para mais informações sobre o deploy de aplicações heroku através de containers Docker, acesse a documentação oficial.

fiuzatayna commented on 20 Jun

Versão do Heroku instalada aqui hoje (20 jun 2022)

```
$ heroku --version
heroku/7.60.2 linux-x64 node-v14.19.0
```