Docker image

Tiago Heinrich

UniSociesc Joinville

07/05/2020

Objetivo

- Criar um conjunto de imagens básicas
- Explorar algumas opções oferecidas pelo Docker
- Atividade D2

Image

- Site do DockerHub
- Objetivo é utilizar imagens oficiais para criar uma nova imagem com aplicação
- Usando o `docker build`, os usuários podem criar um build automatizado que executa várias instruções da linha de comando em sucessão

```
docker search tomcat
docker pull tomcat
docker inspect tomcat
```

Docker exec

- Solicite um pseudo-tty e um comando interativo
- Você também pode executar processos adicionais em segundo plano via `docker exec` (você perde a repetibilidade da implantação da imagem se você depende deste mecanismo)

```
docker run -d -t ubuntu /bin/bash
docker ps
docker inspect ID
docker exec -t -i ID /bin/bash
```

Docker

- O `log` captura qualquer coisa enviada para `stdout` ou `stderr` no contêiner (capturada pelo daemon do Docker)
- A ferramenta pode transmitir a partir deste ponto final e, a cada poucos segundos, o relatório de volta em um ou mais contêineres listados, fornecendo informações estatísticas básicas sobre o que acontecendo

```
docker logs ID
docker stats
```

Docker

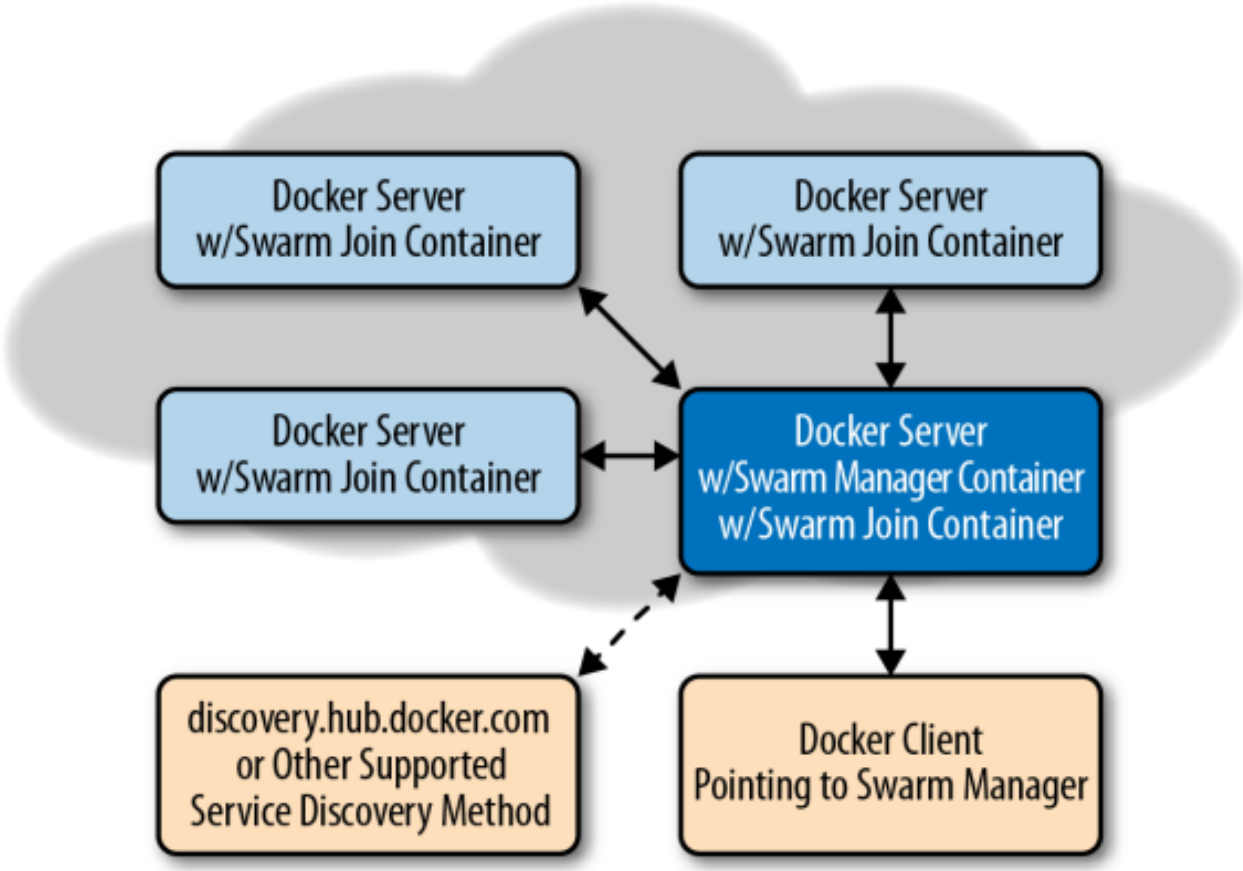
- O Docker possui um comando interno para mostrar o que está sendo executado dentro de um contêiner

```
docker top ID
```

Docker Swarm

- `libswarm` project/released 2015
- A ideia por trás do Swarm é apresentar uma única interface para a ferramenta cliente do docker, mas fazer com que essa interface seja apoiada por um `cluster` inteiro, em vez de um único daemon do Docker
- O Swarm é implementado como um único contêiner do Docker que atua como o hub de gerenciamento central do cluster do Docker e também como o agente que é executado em cada host do Docker

Docker Swarm



Dockerfile

- Os `Dockerfile` permitem criar suas próprias imagens
- Um `Dockerfile` descreve o software que compõe uma imagem
- Os `Dockerfile` contêm um conjunto de instruções que especificam qual ambiente usar e quais comandos executar

Dockerfile

- `FROM` inicializa um novo estágio de construção e define a imagem base para instruções subsequentes
- `LABEL` define o campo Autor das imagens geradas. Você pode usar qualquer par de valor-chave nos rótulos
- `ADD` copia novos arquivos, diretórios ou URLs de arquivos remotos
- `EXPOSE` informa ao Docker que o contêiner escuta nas portas de rede especificadas em tempo de execução
- `CMD` especifica o que executar quando o contêiner (não a imagem) for executado

Docker Prática

First Dockerfile

1. Criar um diretorio com o arquivo `Dockerfile` (não devo colocar uma extensão)

```
FROM ubuntu:14.04
ENTRYPOINT ["/bin/echo"]
```

2. Build da nova imagem com

```
docker build -t basic-bash .
```

First Dockerfile

3. Agora estamos prontos para executar este contêiner, especificando o ID da imagem recém criada

```
docker images
```

4. Passando uma entrada/string para ele

```
docker run ID Super Docker Echo
```

First Dockerfile

5. O contêiner executou o comando definido pela instrução ENTRYPOINT.
6. Como o valor ENTRYPOINT não pode ser substituído esta imagem só poderá executar `echo`
7. Outro exemplo:

```
FROM ubuntu:14.04
ENTRYPOINT ["/bin/ls"]
```

First Dockerfile

8. Parece o mesmo, mas se passarmos um novo executável como argumento para o comando docker run, esse comando será executado

```
FROM ubuntu:14.04
CMD ["/bin/echo", "Hi Docker :)"]
```

9. Esta execução não necessita de uma entrada do usuário

```
docker build -t third-cmd .
docker run third-cmd
```

Flask application

- O Flask é um popular framework web de Python (desenvolvimento de aplicativos da web)
  - Design de API e trabalho com APIs de terceiros
  - Bases de dados
  - Armazenamento em cache

Flask application

```
#!/usr/bin/env python

from flask import Flask
app = Flask(__name__)

@app.route('/hi')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port
        =5000)
```

Flask application

1. Para que esse aplicativo seja executado dentro de um contêiner do Docker, precisamos escrever um Dockerfile que instale os pré-requisitos necessários para executar o aplicativo

```
FROM ubuntu:14.04

RUN apt-get update
RUN apt-get install -y python
RUN apt-get install -y python-pip
RUN apt-get install clean all
RUN pip install flask

ADD hello.py /tmp/hello.py
EXPOSE 5000
CMD ["python","/tmp/hello.py"]
```

Flask application

2. Para executar o aplicativo, usamos a opção -d da docker run que `daemonizes` o contêiner e também a opção -P da docker run para permitir que o Docker escolha uma porta no host do Docker que será mapeada para a porta exposta especificada no Dockerfile

```
docker build -t flask .
docker run flask

docker run -d -P flask
```

Flask application

3. No entanto, como usamos o CMD e não o ENTRYPOINT, podemos substituí-lo e iniciar o contêiner no modo interativo, para explorá-lo

```
docker run -t -i -P flask /bin/bash
```

Docker Images Tags

1. Você está criando várias imagens e várias versões da mesma imagem. Você deseja acompanhar facilmente cada imagem e suas versões, em vez de usar um ID da imagem.
2. Marque a imagem com o comando docker tag. Isso permite renomear uma imagem existente ou criar uma nova marca com o mesmo nome

```
docker images
# Rename
docker tag ubuntu:14.04 foobar
docker tag ubuntu:14.04 foobar:book
```

Segunda opção com Python2

```
FROM python:2

RUN pip install flask

ADD hello.py /

EXPOSE 5000

CMD ["python","./hello.py"]
```

Python3 dockerfile

1. Para criar uma aplicação simples no python3, basta aplicar o mesmo conceito

```
FROM python:3

ADD my_script.py /

RUN pip install pystrich

CMD [ "python", "./my_script.py" ]
```

```
docker build -t python-barcode .
docker run python-barcode
```

Dockerfile Web Server

1. Criar uma aplicação utilizando o Nginx
2. Desenvolver a estrutura que será utilizada

```
FROM nginx

COPY wrapper.sh /

COPY html /usr/share/nginx/html

CMD ["./wrapper.sh"]
```

Dockerfile Web Server

3. Agora para criar e executar a imagem

```
docker build -t static-site .
docker run -p 8888:80 static-site
docker port ID
http://172.17.0.1:8888/
```