Nome: Francisco Felipe de Oliveira
Professor: Ricardo Duarte Taveira
Avaliação 9 – Docker Compose

A Arquitetura Docker é feita de camadas. A camada inferior é o servidor físico que usamos para hospedar máquinas virtuais. Isso é o mesmo que uma arquitetura de virtualização tradicional. A segunda camada é o sistema operacional host, que é a máquina base (ou seja, Windows ou Linux). Em seguida, é o Docker Engine, que usamos para executar o sistema operacional. Acima disso estão os aplicativos que são executados como contêineres do Docker. Esses objetos do Docker são compostos de imagens e contêineres.

Você pode definir sua pilha de aplicativos em um arquivo e manter esse arquivo na raiz do repositório do projeto, sob controle de versão. Essa abordagem permite que outras pessoas contribuam com seu projeto. Eles só precisariam clonar seu repositório.

A estrutura básica do Docker depende de imagens e contêineres. Pense em imagens e contêineres como dois estados diferentes do mesmo conceito subjacente. Um contêiner é como um objeto e uma imagem é como sua classe. Pense em um contêiner como um sistema isolado que contém tudo o que é necessário para executar um determinado aplicativo. É uma instância de uma imagem que simula o ambiente requerido.

O Docker Compose é uma ferramenta do Docker usada para definir e executar aplicativos de vários contêineres. Com o Compose, você usa um YAMLarquivo para configurar os serviços do seu aplicativo e cria todos os serviços do aplicativo a partir dessa configuração.

O Docker Compose ajuda a definir e compartilhar aplicativos de vários contêineres. Com o Docker Compose, você pode criar um arquivo para definir os serviços. Com um único comando, você pode criar tudo ou derrubá-lo.

Pense em docker-compose um fluxo de trabalho automatizado de vários contêineres. O Compose é uma excelente ferramenta para desenvolvimento, teste, fluxos de trabalho de CI e ambientes de preparação. De acordo com a documentação do Docker, os recursos mais populares do Docker Compose são:

- Vários ambientes isolados em um único host.
- Preservar os dados de volume quando os contêineres são criados.
- Recrie apenas contêineres que foram alterados.
- Variáveis e mover uma composição entre ambientes.
- Orquestre vários contêineres que funcionam juntos.

Como usar e instalar o Docker Compose

O Compose usa o Docker Engine, então você precisará ter o Docker Engine instalado no seu dispositivo. Você pode executar o Compose no Windows, Mac e Linux de 64 bits. Instalar o Docker Compose é realmente muito fácil. Em sistemas de desktop, como o Docker Desktop para Mac e Windows, o Docker Compose já está incluído. Nenhuma etapa adicional é necessária. Em sistemas Linux, você precisará:

1. Instalar o Docker Engine
2. Execute o seguinte comando para baixar o Docker Compose: *sudo curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose*
3. Aplique permissões ao binário, assim: *sudo chmod +x /usr/local/bin/docker-compose*

4. Teste a instalação para verificar se funcionou corretamente: *$ docker-compose --version docker-compose version 1.26.2, build 1110ad01*

Independentemente de como você escolheu instalá-lo, depois de baixar e executar o Docker Compose corretamente, você pode começar a usá-lo com seus Dockerfiles. Este processo requer três etapas básicas:

1. Defina o ambiente do seu aplicativo usando um Dockerfile. Dessa forma, ele pode ser reproduzido.
2. Defina os serviços para seu aplicativo em um docker-compose.ymlarquivo. Dessa forma, eles podem ser executados em um ambiente isolado.
3. Corra docker-composepara iniciar seu aplicativo.

Você pode adicionar facilmente o Docker Compose a um projeto pré-existente. Se você já tiver alguns Dockerfiles, adicione os arquivos Docker Compose abrindo a Paleta de Comandos. Use o Docker: Docker Compose Files para o comando Workspace e, quando promovido, escolha os Dockerfiles que deseja incluir.

Você também pode adicionar arquivos do Docker Compose ao seu espaço de trabalho ao adicionar um Dockerfile. Da mesma forma, abra a paleta de comandos e use o comando Docker: Adicionar arquivos do Docker ao espaço de trabalho.

Você será perguntado se deseja adicionar algum arquivo do Docker Compose. Em ambos os casos, a extensão Compose adicionará o docker-compose.ymlarquivo ao seu espaço de trabalho.

Estrutura de arquivos do Docker Compose

Agora que sabemos como baixar o Docker Compose, precisamos entender como funcionam os arquivos do Compose. Na verdade, é mais simples do que parece. Resumindo, os arquivos do Docker Compose funcionam aplicando vários comandos que são declarados em um único docker-compose.ymlarquivo de configuração.

Estrutura básica de um arquivo YAML do Docker Compose:

```yaml
versão: 'X'

Serviços:
  rede:
    construir: .
    portas:
      - "5000:5000"
    volumes:
      - .:/código
  redis:
    imagem: redis
```

Agora, vamos ver um exemplo do mundo real de um arquivo do Docker Compose e dividi-lo passo a passo para entender tudo isso melhor. Observe que todas as cláusulas e palavras-chave neste exemplo são palavras-chave comumente usadas e padrão do setor.

Com apenas isso, você pode iniciar um fluxo de trabalho de desenvolvimento. Existem algumas palavras-chave mais avançadas que você pode usar na produção, mas, por enquanto, vamos começar com as cláusulas necessárias.

```yaml
versão: '3'
Serviços:
  rede:
    # Caminho para dockerfile.
    # '.' representa o diretório atual no qual
    # docker-compose.yml está presente.
    construir: .

    # Mapeamento da porta do contêiner para o host

    portas:
      - "5000:5000"
    # Montar volume
    volumes:
      - "/código de usuário/:/código"

    # Vincule o contêiner do banco de dados ao contêiner do aplicativo
    # para acessibilidade.
    links:
      - "banco de dados:backenddb"

  base de dados:

    # imagem para buscar no hub do docker
    imagem: mysql/mysql-server:5.7

    # Variáveis de ambiente para script de inicialização
    # container usará essas variáveis
    # para iniciar o contêiner com essas variáveis definidas .
    meio Ambiente:
      - "MYSQL_ROOT_PASSWORD=raiz"
      - "MYSQL_USER = usuário de teste "
      - "MYSQL_PASSWORD=admin123"
      - "MYSQL_DATABASE=back-end"
    # Monte o arquivo init.sql para executar automaticamente
    # e crie tabelas para nós.
    # tudo na pasta docker-entrypoint-initdb.d
    # é executado assim que o container está funcionando .
    volumes:
      - "/usercode/db/init.sql:/docker-entrypoint-initdb.d/init.sql "

```

Comandos de composição do Docker

Agora que sabemos como criar um *docker-compose* arquivo, vamos examinar os comandos mais comuns do Docker Compose que podemos usar com nossos arquivos. Lembre-se de que discutiremos apenas os comandos usados com mais frequência.

docker-compose: Todo comando Compose começa com este comando. Você também pode usar *docker-compose <command> --help* para fornecer informações adicionais sobre argumentos e detalhes de implementação.

```
$ docker-compose --help
Define and run multi-container applications with Docker.
```

docker-compose build: Este comando cria imagens no *docker-compose.yml* arquivo. O trabalho do *build* comando é preparar as imagens para criar contêineres, portanto, se um serviço estiver usando a imagem pré-construída, ele ignorará esse serviço.

```
$ docker-compose build
database uses an image, skipping
Building web
Step 1/11 : FROM python:3.9-rc-buster
 ---> 2e0edf7d3a8a
Step 2/11 : RUN apt-get update && apt-get install -y docker.io
```

docker-compose images: Este comando listará as imagens que você construiu usando o *docker-compose* arquivo atual.

```
$ docker-compose images
          Container                  Repository        Tag       Image Id       Siz
e
--------------------------------------------------------------------------------
---
7001788f31a9_docker_database_1   mysql/mysql-server   5.7      2a6c84ecfcb2   333.
9 MB
docker_database_1                mysql/mysql-server   5.7      2a6c84ecfcb2   333.
9 MB
docker_web_1                     <none>               <none>   d986d824dae4   953 M
B
```

docker-compose stop: este comando interrompe os contêineres em execução dos serviços especificados.

```
$ docker-compose stop
Stopping docker_web_1      ... done
Stopping docker_database_1 ... done
```

docker-compose run: Isso é semelhante *ao docker run* comando. Ele criará contêineres a partir de imagens construídas para os serviços mencionados no arquivo de composição.

```
$ docker-compose run web
Starting 7001788f31a9_docker_database_1 ... done
 * Serving Flask app "app.py" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 116-917-688
```

docker-compose up: este comando faz o trabalho dos *comandos docker-compose buil* de *docker-compose run*. Ele constrói as imagens se elas não estiverem localizadas localmente e inicia os contêineres. Se as imagens já estiverem construídas, ele irá bifurcar o contêiner diretamente.

```
$ docker-compose up
Creating docker_database_1 ... done
Creating docker_web_1      ... done
Attaching to docker_database_1, docker_web_1
```

docker-compose ps : este comando lista todos os contêineres no *docker-compose* arquivo atual. Eles podem então estar em execução ou parados.

```
$ docker-compose ps
      Name                 Command            State              Ports
-------------------------------------------------------------------------------
docker_database_1   /entrypoint.sh mysqld   Up (healthy)   3306/tcp, 33060/tcp
docker_web_1        flask run               Up             0.0.0.0:5000->5000/tcp

$ docker-compose ps
      Name                 Command          State    Ports
-----------------------------------------------------------
docker_database_1   /entrypoint.sh mysqld   Exit 0
docker_web_1        flask run               Exit 0
```

docker-compose down : Este comando é semelhante ao *docker system prune* comando. No entanto, no Compose, ele interrompe todos os serviços e limpa os contêineres, redes e imagens.

```
$ docker-compose down
Removing docker_web_1      ... done
Removing docker_database_1 ... done
Removing network docker_default
(django-tuts) Venkateshs-MacBook-Air:Docker venkateshachintalwar$ docker-compose im
ages
Container   Repository   Tag   Image Id   Size
------------------------------------------------
(django-tuts) Venkateshs-MacBook-Air:Docker venkateshachintalwar$ docker-compose ps
Name   Command   State   Ports
------------------------------
```