Virtualização de Redes

TP1 - Docker

Versão 2

Data de vencimento: 14/03/2021 23:59

MIEI / MEI / MERSTEL
Departamento de Informática
Universidade do Minho

Fevereiro de 2021

criado por: Fábio Gonçalves mantido
por: João Pereira

Metas

Este trabalho deve permitir adquirir o conhecimento básico para trabalhar com virtualização Docker. O objetivo principal deste trabalho é obter as ferramentas necessárias para:

- Criar, excluir e gerenciar contêineres docker;
- Crie imagens docker;
- Use docker-compose;
- Crie compilações automatizadas.

Instalação do Docker

O Docker pode ser instalado em qualquer sistema operacional, embora, no Windows, isso só seja possível na versão Pro. Os guias apresentados aqui foram feitos em uma máquina Ubuntu, mas devem ser independentes de plataforma. O Docker pode ser instalado seguindo um dos guias apresentados, dependendo do seu sistema operacional:

- Linux - https://docs.docker.com/install/linux/docker-ce/ubuntu/
- Mac - https://store.docker.com/editions/community/docker-ce-desktop-mac/plans/docker- ce-desktop-mac-tier? tab = instruções
- Janelas - https://store.docker.com/editions/community/docker-ce-desktop-windows/ Plans / docker-ce-desktop-windows-tier? tab = instruções

Após a instalação, teste o contêiner executando docker run hello-world.

NÃO SE ESQUEÇA DE ATUALIZAR AS PERMISSÕES

sudo usermod -aG docker

Comandos Básicos

Há um conjunto de comandos básicos que permitem fazer a maioria das operações de gerenciamento no docker: criar, remover e listar contêineres. Alguns desses comandos são:

- docker run - executa um comando em um novo contêiner:
    - docker run ubuntu: mais recente
- docker exec - executa um comando em um contêiner existente:
    - docker exec existing_container ls
- parada do dock - para o contêiner;
- docker start - inicia um contêiner parado;
- docker rm - exclui um contêiner interrompido;
- imagens docker - lista todas as imagens existentes;
- docker ps -a - lista os contêineres existentes;
- registros do docker - veja os registros do contêiner;
- docker rmi - exclui uma imagem.

1 Noções básicas de contêineres

Os contêineres do Docker são criados a partir de imagens pré-construídas. Eles podem ser obtidos diretamente no repositório do hub do docker (https://hub.docker.com) ou criado manualmente usando um Dockerfile.

1.1 Gerenciando imagens docker

O comando docker permite puxar, excluir ou criar novas imagens. Para este exercício, uma imagem será extraída do repositório do docker. Para fazer isso, escolha uma imagem do Linux do https://hub.docker.com local na rede Internet. Faça o download emitindo:

$ docker pull image_name: tag

O comando anterior recebe o image_name e a tag como argumentos. O image_name é o nome da imagem a ser baixada. A tag é a versão desta imagem. Se nenhuma tag ou palavra-chave mais recente for usada, a imagem mais recente será baixada. Para listar as imagens baixadas, use o comando:

imagens de $ docker

Eles podem ser excluídos com o comando:

$ docker rmi image_id

1.2 Gerenciando contêineres

Usando uma das imagens baixadas, agora é possível criar um contêiner a partir dela. Neste exemplo, um contêiner Linux será criado. O comando emitido abrirá um terminal dentro do contêiner. Isso permitirá explorar o contêiner como um sistema operacional normal. Nesta etapa, a imagem baixada deve ser usada.

$ docker run -it ubuntu: latest / bin / bash

O estado dos contêineres pode ser visto com o comando:

$ docker ps -a

Um dos campos no resultado do comando acima é o nome do contêiner. Se nenhum foi atribuído, o contêiner terá um nome aleatório. Usando o nome na tabela, o contêiner pode ser interrompido e excluído:

$ docker stop container_name $ docker rm

container_name

Para atribuir um nome de container, facilitando seu gerenciamento, o flag –name deve ser adicionado ao criar o container:

$ docker run --name test -it ubuntu: latest / bin / bash

Nos exemplos anteriores, todos os contêineres foram criados usando docker run. Isso permite criar simultaneamente um contêiner e abrir um shell. O próximo comando pode ser usado para abrir um shell em um contêiner já em execução.

$ docker exec -it nome_do_container / bin / bash

Neste exemplo, o nome do contêiner será o teste criado anteriormente. Depois de terminar com sucesso este exemplo, o recipiente deve ser parado e removido. Em seguida, crie um novo container usando uma imagem que ainda não foi baixada, outro tipo de Linux, por exemplo.

1.3 Persistência

Os dados do contêiner não são persistentes. O que significa que todos os dados do contêiner são removidos quando são excluídos.

Existem três maneiras de persistir os dados do contêiner:

- **Volumes:** Esse tipo de persistência permite que os contêineres compartilhem dados entre os contêineres. Eles são criados no sistema host e gerenciados diretamente pelo docker;
- **Bind Mount:** Isso permite que os contêineres montem diretórios no sistema host. Assim, possibilitando o acesso aos dados diretamente do host. Normalmente usado para compartilhar dados entre o host e os contêineres;
- **Tmpfs:** Eles são armazenados apenas na memória e nunca no disco host;

1.3.1 Persistência - Volumes

Para usar os volumes do docker, a primeira etapa é criar o próprio volume:

$ docker volume criar test-vol

Então, o volume pode ser visto com os seguintes comandos:

$ docker volume ls

$ docker volume inspect test-vol

O primeiro comando mostra os volumes existentes. A próxima, permite ver todas as características do volume selecionado. Os volumes do Docker permitem o uso de vários drivers. Seu uso pode ser visto na documentação do docker.

Para remover um volume:

$ docker volume rm test-vol

Para exemplificar os recursos dos volumes do docker, vamos começar criando um novo volume:

$ docker volume criar novo-test-vol

Em seguida, um container será criado com a montagem do volume no diretório do container / home / test.

$ docker run --name test-vol-container --mount
source = new-test-vol, target = / home / test

O comando acima abrirá um terminal shell dentro do contêiner. Em seguida, um arquivo deve ser criado no diretório montado:

$ echo “test 123” >> /home/test/test.txt

Para verificar se a persistência está funcionando, remova o container e crie-o novamente com o mesmo comando.

Se em outro terminal o próximo comando for executado:

$ docker run --name test-vol-container-2 --mount
source = new-test-vol, target = / home / test2

É possível ver que este arquivo existe também neste novo container. Desta vez, no diretório / home / test2. O conteúdo dos diretórios é compartilhado entre os contêineres.

1.3.2 Persistência - Montar ligação

Para montar diretamente um diretório de host no contêiner, emita o seguinte comando (Ajustar para seu usuário e sistema):

$ docker run --name test-mount-container -v
/ home / nome_do_user / test-dir: / home / test -it ubuntu: mais recente
/ bin / bash

É possível ver que o host do sistema agora possui um novo diretório chamado test-dir. Se um arquivo for criado neste diretório, ele também existirá dentro do contêiner. Executando o seguinte comando, é possível visualizar os arquivos recém-criados dentro do container:

$ ls / home / test

1.4 Mapeamento de porta

Um contêiner é um sistema fechado sem conexão de rede externa. No entanto, é possível mapear portas internas para as portas do host. Para fazer isso, o - **p** bandeira é usada:

$ docker run -it --name test-port -p 8888: 9999 ubuntu: mais recente
/ bin / bash

Usando a ferramenta netcat é possível verificar esta funcionalidade. Para isso, os próximos comandos podem ser executados:

atualização de $ apt

$ apt install netcat $ nc -l -vv -p

9999

E, no sistema host, execute:

$ wget localhost: 8888

No contêiner, deve ser possível ver a conexão e os dados recebidos. Assim, é possível verificar que o comando mapeou a porta interna 9999 do docker para o host 8888. Porém, ele não permite a comunicação por meio de pacotes udp. Para fazer isso, o seguinte comando pode ser usado:

$ docker run -it --name test-port-udp -p 8888: 9999 / udp
ubuntu: / bin / bash mais recente

1.5 Rede

Existem vários tipos de redes docker. No entanto, as conexões de ponte são usadas por padrão. O comando para listar todas as redes docker é:

$ docker network ls

Por padrão, devem existir 3 redes: bridge, host e null. A rede de ponte permite que os contêineres sejam conectados entre si. A rede do host é usada para o contêiner se conectar ao host. Finalmente, a rede nula existe para permitir que um contêiner seja executado sem nenhum dispositivo de rede.

Para testar as redes docker, primeiro serão criados dois contêineres:

$ docker run -dit --name test-net-1 ubuntu: latest / bin / bash $ docker run -dit --name test-net-2

ubuntu: latest / bin / bash

Nesse caso, o sinalizador -d é usado com os sinalizadores -it. Ele permite que o contêiner seja iniciado em segundo plano no modo daemon. Para se conectar a este contêiner, o seguinte comando pode ser usado:

$ docker attach test-net-1

Após a criação dos containers, é possível verificar a configuração da rede. O seguinte comando é usado para fazer isso:

$ docker rede inspecionar ponte

Agora é possível comparar as configurações obtidas com o comando anterior e emissão **ipconfig** dentro do contêiner.

Com um terminal shell dentro do container, é possível perceber que se trata de uma conexão com a rede externa e com a internet. Ele pode ser testado executando ping no Google, por exemplo. No entanto, se tentar fazer ping entre o contêiner pelo nome, não será possível:

$ docker attach test-net-1

$ ping test-net-2

Para isso, eles deveriam estar conectados à mesma rede, como no próximo exemplo:

$ docker network criar user-net

Os contêineres são conectados à rede usando a bandeira **-rede:**

$ docker run -dit --name test-user-net-1 --network user-net
ubuntu: / bin / bash mais recente

$ docker run -dit --name test-user-net-2 --network user-net
ubuntu: / bin / bash mais recente

Após inspecionar esta nova rede, pode-se verificar que ela possui a configuração para os dois containers. A conexão pode ser testada com:

$ docker attach test-user-net-1

$ ping test-user-net-2

As redes podem ser excluídas com o seguinte comando:

$ docker network rm network_name

2 docker-compose

Docker-compose permite definir e executar várias configurações e também configurar as aplicações ou serviço que irá iniciar. Portanto, apenas um comando é necessário para iniciar todos os contêineres. As instruções de instalação podem ser encontradas em https: // docs.
docker.com/compose/install/#install-compose .
Os comandos básicos do Docker-compose são:

- **docker-compose up:** permite iniciar os contêineres
- **docker-compose stop:** encerra todos os contêineres em execução;
- **docker-compose rm:** apaga os contêineres;
- **registros docker-compose:** para ver os registros de todos os serviços descritos no dockercompose.

O exemplo a seguir mostra um exemplo de um **docker-compose.yml.** Para testá-lo, crie um arquivo chamado **docker-compose.yml** e copie todo o conteúdo do exemplo abaixo.

```
versão: '3'
Serviços:
    serviço1:
        imagem: httpd: lastest
        portas:
            - "8080: 80"
        volumes:
            - httpd-vol: / usr / local / apache2
        redes:
            - container-net
volumes:
    httpd-vol:
redes:
    container-net:
```

Este arquivo contém a versão do docker-compose a ser utilizada, a lista de serviços a serem executados e sua configuração, os volumes e as redes. Em cada serviço encontra-se a sua configuração, onde se encontram todos os parâmetros necessários a cada serviço.
Após a criação do arquivo, no mesmo diretório onde se encontra o arquivo, deve-se executar o seguinte comando:

$ docker-compose up -d

Este comando inicia todos os contêineres no docker-compose. O sinalizador -d significa que os contêineres serão enviados para segundo plano no modo daemon.
Depois que os contêineres são iniciados, no host um navegador deve ser aberto no url http: // localhost: 8080 . Um serviço da web deve estar em execução na porta 8080.
Em seguida, os comandos docker ls, docker network inspect, docker volume ls e docker volume inspect devem ser usados para verificar todos os parâmetros do contêiner.

3 Dockerfile

Dockerfile é um arquivo de texto que contém uma lista de comandos que descrevem como uma imagem docker será criada. Isso será executado sequencialmente.
Normalmente, a primeira linha indica com a imagem que vai ser usada. O exemplo a seguir mostra um exemplo de dockerfile.

```
DO ubuntu: mais recente
EXPOR 8888
EXECUTE apt-get update && apt-get -y install netcat VOLUME / home / output

ENTRYPOINT ["/ bin / nc", "-k", "-l", "-p", "8888", "-vv"]
```

A primeira linha indica que a imagem base que será usada é um Ubuntu. O, o
EXPOR palavra-chave significa que a porta 8888 será exposta. Isso significa que este último pode ser mapeado para uma porta do host.
A próxima linha serve para instalar a ferramenta netcat. Em seguida, é criado o / **casa / saída** volume. Esta última etapa permite criar um volume que pode ser persistido posteriormente.
finalmente, o **PONTO DE ENTRADA** palavra-chave é o comando que será executado quando o container for iniciado. Para testar o Dockerfile criado, primeiro a imagem deve ser criada. No diretório de arquivos do Docker, execute:

construção de $ docker. -t teste: 0,1

O - **t** flag permite atribuir um nome a esta imagem. Nesse caso **teste: 0,1.**
Depois de criar a imagem, um contêiner pode ser criado. Para fazer isso:

$ docker run --name test-dockerfile -p 8888: 8888 -d test: 0.1

Isso permite que um contêiner seja iniciado no modo daemon. O container terá o porto
8888, conforme indicado no arquivo Docker. Em seguida, execute o seguinte comando:

$ docker registra test-dockerfile

Isso permite ver os logs do contêiner. Em outro terminal agora execute:

$ wget localhost: 8080

Nos logs do docker, deve ser possível ver se a conexão foi feita com êxito.

4 construções automáticas

O hub Docker permite construir imagens automaticamente. Para fazer isso, um dockerfile deve ser fornecido. Para fazer isso, é necessário criar uma conta github e docker hub. As etapas necessárias para criar uma compilação automática são:

1. Crie um projeto github:
2. Enviar o arquivo docker para o github;
3. Crie um projeto no hub docker;
4. Vinculando os dois projetos;

5 exercícios práticos

1. Como criar um contêiner a partir de um Linux alpine que tem a porta do contêiner 9999 mapeada no host 8668. Ele também deve ter uma montagem de ligação, montando o contêiner / home / internal_dir em hosts / home / user / docker_dir /.
2. Como criar o volume “my-volume-1”?

   (a) Qual é o ponto de montagem do volume? (b) Qual
   driver está sendo usado?

3. Crie dois contêineres básicos (eles não devem ser anexados a nenhuma rede criada manualmente).

(a) É possível inspecionar a rede de ponte e encontrar seus endereços IP?

(b) É possível que os contêineres se comuniquem entre si usando seus nomes?

(c) E com seus IPs?

4. Qual é o comando para criar a rede “my-network-1”?

(a) Como conectar dois contêineres a essa rede?

(b) Qual parâmetro relevante é possível encontrar sobre essa rede? (c) Os contêineres podem executar ping uns nos outros usando seus nomes?

5. Quais foram os resultados obtidos nos comandos **docker network ls, docker net-inspecionar trabalho** e **volume docker ls** na seção 1 deste documento? Que conclusões podemos tirar do resultado desses comandos?

6. Crie um docker-compose que inicie pelo menos dois serviços:

(a) Devem estar na mesma rede docker; (b) Devem compartilhar o mesmo volume do docker:

(c) Um dos serviços também deve ter uma montagem de ligação;

(d) Um dos serviços deve ter sua porta 9999 exposta na porta do host 8888;

(e) Usando os comandos inspect e ls, que conclusões você pode tirar sobre seus resultados?

7. Crie um arquivo Docker que:

(a) Instale alguma ferramenta. Qualquer um escolhido pelo aluno. Pontos extras para qualquer aplicativo que está ouvindo solicitações externas (servidor da web, por exemplo);

(b) Tem volume, para persistência posterior;

8. Crie uma construção automática. O arquivo docker criado no exercício anterior pode ser usado.

Os exercícios devem ser respondidos em um arquivo de relatório, adicionando, quando necessário, capturas de tela. Nas questões 5 e 6, o docker-compose e os Dockerfiles também devem ser adicionados. Para a pergunta 7, apenas o url do repositório do hub do docker precisa ser indicado.

Um arquivo zip com todo o conteúdo (report.pdf, docker-compose e dockerfile) deve ser enviado via quadro-negro até a data de vencimento.