Introdução ao Docker

Jean Phelipe de Oliveira Lima

jpdol.eng16@uea.edu.br

/jpdol

Trabalhando com Volumes

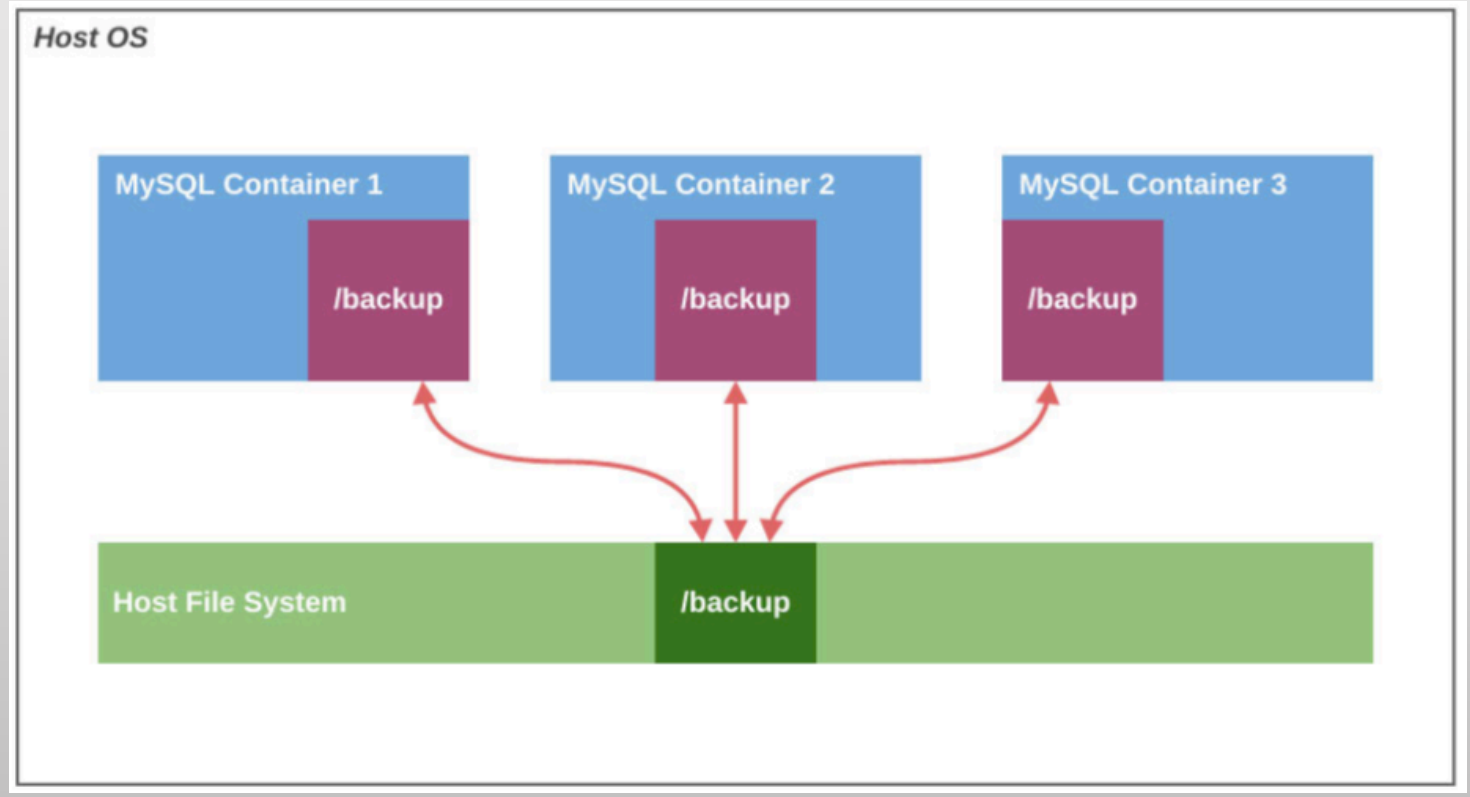
- Volumes
- Gerenciamento de Dados
- Utilizando Volumes no Docker

Volumes

- Um volume pode ser um diretório localizado fora do sistema de arquivos de um *container*. O Docker permite especificar diretórios no *container* para que possam ser mapeados no sistema de arquivos do *host*. Com isso, temos a capacidade de manipular dados no *container* sem que tenham relação alguma com as informações da imagem. Um volume pode ser compartilhado entre vários *containers*.

Volumes

- A figura a seguir ilustra como os volumes funcionam:



Gerenciamento de Dados

- Volumes são diretórios configurados dentro de um *container* que fornecem o recurso de compartilhamento e persistência de dados.

- Características:
  - Os volumes podem ser compartilhados ou reutilizados entre containers;
  - Toda alteração feita em um volume é de forma direta;
  - Volumes alterados não são incluídos quandoa tualizamos uma imagem.

Gerenciamento de Dados

- Ainda utilizando a imagem criada para o Nginx, podemos testar o uso de volumes com a opção **-v**. Com ela, é possível informar um diretório no *host* local que poderá ser acessado no *container*, funcionando de forma parecida com um mapeamento. Além disso, vamos informar o tipo de permissão que o container terá sob o diretório local, sendo: **ro** para somente leitura, e **rw** para leitura e escrita.

- Utilizando a opção **-v:**

```
$ docker run -d -p 8080:8080 –v /tmp/nginx:/usr/share/nginx/html:ro nginx
```

- O diretório /tmp/nginx pertence ao *host* local, e quero que ele seja mapeado para /usr/share/nginx/html dentro do *container* que está sendo inicializado. Além disso, estou passando a instrução **ro** no final para informar ao *container* que ele só poderá ler o conteúdo em /tmp/nginx.

Gerenciamento de Dados

- Para testar o volume criado, vamos gerar um arquivo em /tmp/nginx e inserir algum texto. Entretanto, antes note o retorno de um *request* feito ao acessarmos o *web server*:

```
$ curl -IL http://localhost:8080
→HTTP/1.1 403 Forbidden Server: nginx/1.4.6 (Ubuntu)
```

- Agora, criando o arquivo no host e testando novamente:

```
$ echo "It works!" > /tmp/nginx/index.html
$ curl http://localhost:8080
→ It works!
```

Utilizando Volumes no Docker

- Imagine que temos um *container* com um banco de dados em execução. Nele, é possível controlar os dados que são armazenados em disco para fazer *backups* e até restaurações.
- Para isso, devemos separar a construção deste novo *Dockerfile*. Antes, crie uma pasta chamada mysql e um novo arquivo *Dockerfile* com as seguintes instruções de instalação:

```
FROM ubuntu
DEBIAN_FRONTEND noninteractive
RUN apt-get update -qq && apt-get install -y

mysql-server-5.5
ADD my.cnf /etc/mysql/conf.d/my.cnf
```

Utilizando Volumes no Docker

- RUN chmod 664 /etc/mysql/conf.d/my.cnf ADD run /usr/local/bin/run
  RUN chmod +x /usr/local/bin/run
  VOLUME ["/var/lib/mysql"]
- EXPOSE 3001
  CMD ["/usr/local/bin/run"]

Utilizando Volumes no Docker

- Aqui temos duas novidades. A primeira é o uso da opção **ENV** para declarar uma variável ambiente, que neste caso será utilizada no processo de instalação do MySQL, a fim de evitar telas interativas que são exibidas durante a instalação para cadastrar usuários e configurar o banco de dados.
- E a segunda é que estamos informando que o diretório /var/lib/mysql será um **VOLUME**; assim, poderemos manipular os arquivos de dados salvos pelo MySQL.
- Agora, vamos gerar uma nova imagem chamada mysql, e inicializar um *container*:

Utilizando Volumes no Docker

```
$ sudo docker build -t mysql .
$ sudo docker run -d -p 3001:3001 -e MYSQL_ROOT_PASSWORD=<senha> mysql
```

- A opção **-e** serve para designar valores a variáveis ambiente – neste caso, o valor que ela recebe é a senha de root para o acesso ao MySQL.

- Para testar se o *container* está funcionando perfeitamente, podemos aces- sar o MySQL partindo do host local:

```
$ mysql -h 127.0.0.1 -u root -p
```

Utilizando Volumes no Docker

- Agora, com o *container* em execução e o teste feito, vamos focar no uso de volumes.

- Por padrão, os dados armazenados pelo MySQL residem em /var/lib/mysql, justamente onde inicializamos o nosso volume. Assim, é possível criarmos um novo *container* que terá acesso a esses dados. Para isso, podemos utilizar a opção **--volumes-from**, que recebe como argumento o ID ou nome do *container* a cujos dados queremos ter acesso.

Utilizando Volumes no Docker

- No exemplo a seguir, usaremos uma imagem mínima chamada busybox para ter acesso aos dados do *container* que está em execução com MySQL:

```
$ sudo docker run -i -t --volumes-from <ContainerID> busybox
```

- O Docker inicializou o novo *container* utilizando a imagem busybox, e mapeou o volume do container mysql para a nova instância que foi criada.

Utilizando Volumes no Docker

- Observações:
  - A opção **--volumes-from** pode ser usada para replicar esse volume em múltiplos *containers*.
  - Se você remover o *container* que monta o volume inicial (ou seja, o do mysql), os posteriores que foram inicializados com a opção **--volumes-from** não serão excluídos.
  - Para remover o volume do disco, você pode utilizar a opção **-v** combi- nada com o **rm**, como **docker rm -v volume_name**.

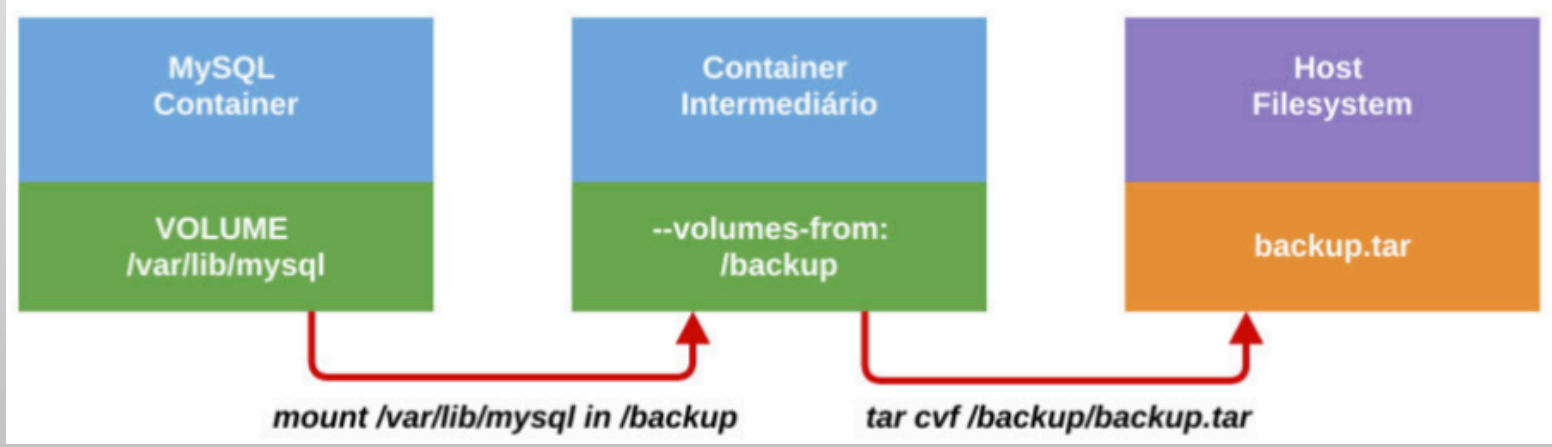
Backup e Restore de Volumes

- Outra forma de uso para **--volumes-from** pode ser para a construção de *backups*. Imagine que você queira realizar um *backup* do diretório /var/lib/mysql do *container* para o seu *host* local:

```
$ docker run --volumes-from <ContainerID>-v \
$ (pwd):/backup ubuntu tar cvf /backup/backup.tar /var/lib/mysql
```

Backup e Restore de Volumes

- Explicando:
  - Primeiro estamos criando um novo *container* com acesso ao volume de dados do mysql;
  - Em seguida, estamos criando um mapeamento entre o nosso *host* local e o novo *container*, de forma que tudo o que for criado no diretório mapeado dentro deste novo *container* esteja disponível para o nosso *host* local.
  - Resumo: o novo *container* copia os dados da instância que está com o mysql ativo, compacta tudo e disponibiliza para o nosso *host* local.



Introdução ao Docker

Jean Phelipe de Oliveira Lima

jpdol.eng16@uea.edu.br

/jpdol