Volumes – Exercícios - Resolvidos

1 – Dê um exemplo prático de como podemos compartilhar dados entre um Contêiner e o Host usando volumes. **Resposta:**

Como exemplo vamos usar a imagem do Nginx .

Por padrão a imagem do Nginx armazena os logs da aplicação no diretório **/var/log/nginx** no contêiner e escuta na porta 80. Esta pasta não pode ser acessada de fora do contêiner.

Vamos tornar os dados dessa pasta no contêiner acessível a partir do Host.

Para isso vamos criar um contêiner e definir um mapeamento de volumes usando a tag -v

docker container run --name=nginx1 -d -v ~/nginxlogs:/var/log/nginx -p 5000:80 nginx

Onde:
--name=nginx1 é o nome do contêiner
-d roda o processo em segundo plano
-v ~/nginxlogs:/var/log/nginx monta um volume e o vincula o diretório **/var/log/nginx** do contêiner ao diretório ~/nginxlogs do host
-p 5000:80 mapeia a porta 80 do contêiner para a porta 5000 do host
nginx é o nome da imagem usada

Ao executar este comando o contêiner nginx1 será criado
A pasta **nginxlogs** será criada na sua pasta local no host

Ao entrar nesta pasta e digitar ls -g você verá os logs do nginx(*access.log e error.log*) que foram copiados da pasta do contêiner

Acesse o navegador em http://localhost:5000 e depois no terminal digite o comando **cat ~/nginxlogs/access.log** para ver o resultado.

Volumes – Exercícios - Resolvidos

2 - Crie um volume de dados chamado **Volume1**. A seguir crie um contêiner e mapeie este volume para uma pasta do contêiner.

Resposta:

A partir da versão 1.9 do Docker o comando **docker volume create** permite criar um volume se relação com qualquer contêiner.

Obs: Este comando é equivalente ao usar a **tag -v** na criação contêiner da seguinte forma: **-v path:/path/conteiner**

Podemos então criar o volume **Volume1** usando o comando:

docker volume create –name Volume1

A seguir vamos criar um contêiner usando a imagem do **alpine** mapeando este volume para uma pasta do contêiner:

docker container run --rm -it --name ct1 -v Volume1:/Volume1 alpine

Nota: a flag --rm vai excluir o contêiner quando você sair dele

Este comando cria o contêiner **ct1** e entra no contêiner.

Digitando o comando **ls -g** no contêiner você verá a pasta **Volume1** criada no contêiner.

Agora digite os comandos a seguir no contêiner:

```
/ # echo ‘exemplo 1’ > /Volume1/teste.txt
/ # exit
```

O contêiner **ct1** foi excluído e seus dados foram perdidos mas vamos criar um novo contêiner chamado **ct2** mapeando par ao volume **Volume1**

docker container run --rm -it –name ct2 Volume1:/NovoVolume1 alpine

Agora dentro do contêiner digite **ls -g** e você verá a pasta **NovoVolume1** no contêiner

Entre na pasta digitando **cd NovoVolume1** e digite **ls -g** e você verá o arquivo **teste.txt** criado no primeiro contêiner **ct1** que foi excluído.

3 – Dê um exemplo prático de como podemos compartilhar dados entre múltiplos contêineres usando volumes.

Como exemplo poderíamos criar um volume independente usando o comando **docker volume create <volume>** e depois criar vários contêineres mapeando este volume. *(Como no exemplo anterior)*

Esta abordagem é a mais indicada pois o volume criado não depende de nenhum contêiner mas lembre-se o Docker não trata o bloqueio de arquivos e se você precisar de múltiplos contêineres escrevendo para o volume você deve gerenciar este aspecto.

Outra abordagem é mapear um volume usando a **tag -v** e criar um contêiner que vai possuir o volume que você vai compartilhar com os demais contêineres.

Essa abordagem tem o problema de depender do contêiner que contêm o volume, ou seja, se este contêiner for excluído ou cair os demais contêineres perdem o acesso aos dados.
Exemplo:

docker container create --name ct3 -v /Volume2 alpine

Este comando cria um contêiner e define a pasta **Volume2** que pode ser consumida por outros contêineres.

Vamos criar outro contêiner chamado **ct4** que vai usa a pasta **Volume2** que usa imagem do **alpine**:

docker container run -d --volumes-from ct3 --name ct4 alpine

Usando a tag **--volumes-from ct3** agora temos o contêiner **ct4** que tem uma pasta **/Volume2** que é a mesma pasta do contêiner **ct3**

Poderíamos continuar a mapeando contêineres para usar a pasta **Volume2** do contêiner **ct3**.

docker container run -d --volumes-from ct3 --name ct5 alpine

Assim temos os contêineres **ct4 e ct5** compartilhando a pasta **Volume2** que esta no contêiner **ct3**.