Relatório CARD 16 - Prática: Docker e Containers para Aplicações (III)

Willian Augusto Soder de Souza

O objetivo deste relatório é descrever os principais conhecimentos aprendidos ao assistir o curso “Introdução a Docker Containers (Hands On)”, onde o apresentador visa passar uma visão geral sobre docker, desde a sua instalação até os principais comandos, abaixo segue um resumo sobre os conceitos apresentados e uma lista dos principais comandos do docker.

- **Docker**: é uma plataforma de software que permite criar, testar e implantar aplicativos rapidamente em ambientes isolados chamados "containers".
- **Containers:** são unidades padronizadas de software que encapsulam o código de um aplicativo e todas as suas dependências, como bibliotecas, ferramentas e configurações, para que o aplicativo possa ser executado de forma consistente em qualquer ambiente.
- **Imagem:** é um arquivo de leitura única que contém tudo o que é necessário para executar um aplicativo. Isso inclui o código-fonte do aplicativo, bibliotecas, dependências, configurações e o sistema de arquivos necessário. As imagens são usadas como "modelos" para criar e executar containers.

Principais comandos:

- **docker image load:** importa uma imagem Docker.
- **docker image save:** salva uma imagem Docker.
- **docker image tag:** coloca uma tag a uma imagem Docker.
- **docker image pull:** baixa uma imagem Docker do Docker Hub.
- **docker (image, container) ls:** lista todos os container ou imagens ativas, para exibir os inativos é necessário adicionar um ‘-a’.
- **docker (image, container) inspect:** mostra informações sobre o container ou imagem.
- **docker (image, container) rm:** remove o container ou imagem.
- **docker container create:** cria um container (deve-se especificar a imagem).
- **docker container start:** inicia um container**.**
- **docker container stop:** encerra um container.
- **docker container run:** cria, inicia e conecta a um container (define-se nome, imagem e modo), se adicionar ‘-v’ após o ‘run’ é possível montar volumes no container (Volumes são uma maneira de persistir dados e compartilhar arquivos entre o container e o sistema host, ou entre containers.).
- **docker container rename:** permite renomear um container.
- **docker container cp:** permite copiar arquivos ou pastas do sistema para o container.

- **docker logs (container_id):** mostra os logs de um container.

Exemplo prático:

Neste exemplo prático, primeiramente foi criada uma pasta com o nome 'willian' no sistema, e dentro dessa pasta foi adicionado um arquivo chamado 'arquivo.txt'.

```
williansoder@pop-os:~$ mkdir willian
williansoder@pop-os:~$ touch willian/arquivo.txt
williansoder@pop-os:~$ echo "arquivo de texto dentro do sistema">willian/arquivo
.txt
williansoder@pop-os:~$ docker conatiner run --name exemplo -it alpine sh
```

Após isso, foi criado e iniciado um container chamado 'containerex' usando a imagem 'alpine'. Dentro desse container, foi criada a pasta 'exemplo', que também contém um arquivo de texto.

```
williansoder@pop-os:~$ docker container run --name containerex -it alpine sh
/ # mkdir exemplo
/ # cd exemplo
/exemplo # touch carro.txt
/exemplo # echo "Quero uma porsche">carro.txt
/exemplo # cat carro.txt
Quero uma porsche
/exemplo #
```

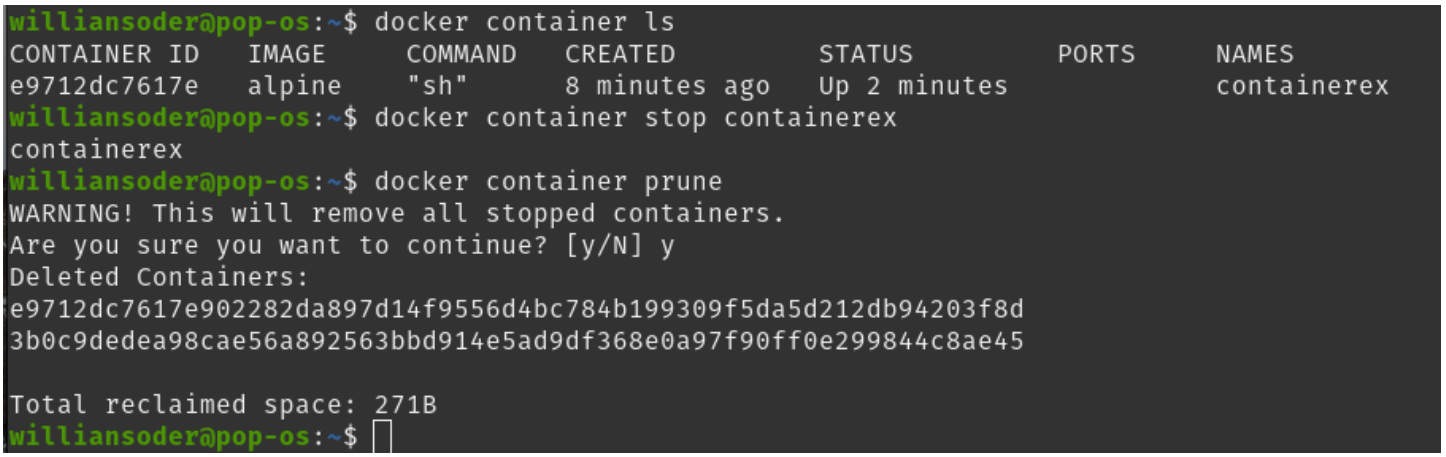
Em seguida, o diretório 'willian' foi copiado do sistema para dentro do container criado.

```
williansoder@pop-os:~$ docker container cp willian containerex:/
Successfully copied 2.56kB to containerex:/
```

Exibindo os diretórios dentro do container criado, é possível observar que tanto o diretório 'willian', que foi copiado do sistema, quanto o diretório 'exemplo', que foi criado diretamente no container, foram corretamente criados e adicionados.

```
williansoder@pop-os:~$ docker container exec containerex ls
bin
dev
etc
exemplo
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
willian
williansoder@pop-os:~$
```

Por fim, o container criado foi parado e excluído usando o comando ‘prune’."



CONCLUSÃO:

Conhecer o básico sobre Docker é essencial para otimizar o desenvolvimento e a gestão de aplicativos. Docker proporciona consistência entre ambientes, eficiência no uso de recursos e facilidade de escalabilidade e portabilidade. Entender como criar e gerenciar imagens e containers, bem como usar comandos fundamentais, melhora significativamente a eficácia no desenvolvimento e na operação de software, garantindo maior controle e flexibilidade.