Docker

O que é docker?

- Iniciado em 2013, ***Docker*** é uma plataforma de código aberto que permite que você crie, teste e implante aplicativos rapidamente.
- Permite que você empacote um aplicativo com todas as partes de que ele precisa, como bibliotecas e outras dependências, e o **envie** como um único pacote.
- Similares a máquinas virtuais, mas mais leves e rápidas, pois compartilham o kernel do sistema operacional.

Vantagens do Docker

- ***Portabilidade***: O Docker é executado em qualquer lugar, seja na nuvem, no seu laptop, em um servidor ou data center.
- ***Consistência***: O Docker garante que o ambiente de desenvolvimento seja o mesmo que o ambiente de produção.
- ***Eficiência***: O Docker permite que você execute mais aplicativos em um único servidor, economizando recursos.
- ***Escalabilidade***: O Docker facilita a escalabilidade de aplicativos pela sua leveza, adicionando ou removendo contêineres eficientemente.

Desvantagens do Docker

- ***Complexidade***: O Docker pode ser complexo para iniciantes, pois envolve a criação de imagens, contêineres, redes, volumes, etc.
- ***Desempenho***: O Docker pode ser mais lento do que a execução de aplicativos diretamente no sistema operacional, devido à virtualização.

O que é uma máquina virtual?

- Uma ***máquina virtual*** é um software que emula um computador físico, permitindo que você execute vários sistemas operacionais em um único hardware.
- Cada máquina virtual inclui um sistema operacional, aplicativos e bibliotecas, é isolada de outras máquinas virtuais.
- As máquinas virtuais são mais pesadas e lentas, pois incluem um sistema operacional completo.

O que é um container?

- Um ***container*** é uma instância de uma imagem Docker, um pacote de leitura somente com um conjunto de instruções para criar um contêiner.
- Um contêiner é uma instância isolada de um sistema operacional, que pode ser executado, parado, movido e deletado.
- Os contêineres são leves e rápidos, pois compartilham o kernel do sistema operacional com outros contêineres.

Casos de uso comuns do Docker:

- ***Desenvolvimento e testes***: O Docker é amplamente utilizado para desenvolvimento e testes de aplicativos, pois permite que você crie ambientes isolados e consistentes.
- ***Estudo e aprendizado***: Permite que você crie e inicie ambientes de desenvolvimento com facilidade, sem afetar o sistema operacional.
- ***Microserviços***: O Docker é uma escolha popular para arquiteturas de microsserviços, pois permite que você empacote e implante serviços independentes.
- ***CI/CD***: O Docker é frequentemente usado em pipelines de CI/CD para criar, testar e implantar aplicativos de forma automatizada.

Terminologia do Docker

- ***Docker Client***: A interface de linha de comando que permite que você interaja com o Docker.
- ***Docker Container***: Um pacote de software com todas as dependências necessárias para executar um aplicativo específico.
- ***Docker Image***: Um ou mais arquivos de leitura, que contém um conjunto de instruções para criar um contêiner. Pode ser compartilhado e reutilizado em diferentes ambientes.

- ***Docker Server***: Também conhecido como Docker daemon, é o comando `dockerd`, responsável por criar e gerenciar contêineres pelo client. Faz o trabalho de construir, executar e distribuir contêineres.
- ***Docker Registry***: Um serviço que armazena e distribui imagens do Docker. O Docker Hub é o registro público mais conhecido.

Glossário Docker

Utilizando containeres docker

Os containeres docker são executados a partir de imagens, que são arquivos de leitura somente, que contém um conjunto de instruções para criar um contêiner. Imagens podem ser encontradas no site do Docker Hub.

Ou também através do comando

```
docker search {nome da imagem}
```

que busca imagens do Docker Hub.

Rodando containers

Alguns comandos úteis do Docker:

```
# Lista comandos da CLI do Docker
docker --help
```

```
# Verifica se a imagem do hello-world está disponível no daemon.
# Caso não esteja, baixa a imagem do Registry - Docker Hub
docker container run hello-world
# ou
docker run hello-world
```

```
# Lista todos os contêineres em execução
# Pode retornar containers finalizados se passado o argumento -a
docker container ls
# ou
docker ps
```

```
# Atribui um nome ao contêiner
docker container run --name c-hello-world hello-world
```

```
# Inicia um contêiner ubuntu no modo interativo
# O comando /bash abre um shell no contêiner
docker container run -it ubuntu /bash
```

```
# Detalhes do contêiner, baseado no ID.
docker container inspect {Id do container}
```

```
# Remove um contêiner baseado no ID
docker container rm {Id do container}
```

```
# -d: Executa o contêiner em segundo plano (modo daemon)
docker container run -it -d ubuntu
```

```
# Para um contêiner baseado no ID
docker container stop {I do container}
```

```
# Acessa um contêiner em execução, no modo interativo
docker container exec -it {Id do container}
```

```
# -p 8080:80: Mapeia a porta 8080 do host para a porta 80 do contêiner
docker container run -d -p 8080:80 nginx
```

Utilizando imagens docker

Imagens docker são criadas a partir de um arquivo chamado ***Dockerfile***, que contém instruções para criar a imagem.
Alguns exemplos de instruções comuns em um ***Dockerfile*** são:

- ***FROM***: Especifica a imagem base.
- ***ENV***: Define variáveis de ambiente.
- ***WORKDIR***: Define o diretório de trabalho.
- ***RUN***: Executa um comando no contêiner.
- ***COPY***: Copia arquivos do host para o contêiner.
- ***CMD***: Especifica o comando a ser executado quando o contêiner é iniciado.

Resumo Dockerfile

Criando uma imagem ubuntu

```
# Busca uma imagem do ubuntu na sua versão mais recente
FROM ubuntu:latest
# Roda os comandos update e install do apt-get
RUN apt-get update && \
  apt-get install curl -y
```

```
# -t: Tag, nome da imagem a ser criada, satin-ubuntu
# e o diretório onde está o Dockerfile
docker image build -t satin-ubuntu .
```

```
# Exibe todas as baixadas localmente
docker image ls
```

```
# Exibe o histórico de criação da imagem
docker history satin-ubuntu
```

```
# Exibe detalhes da imagem
docker image inspect {nome da imagem}
```

Publicando uma imagem no Docker Hub

```
# Loga no Docker Hub
docker login
# Será solicitado usuário e senha
```

```
# Cria uma nova tag para a imagem
docker tag satin-ubuntu psatin/ubuntu
# psatin é o nome do usuário no Docker hub
# ubuntu é o nome da imagem
```

```
# Publica a imagem no Docker Hub
docker push psatin/satin-ubuntu
```

Referências:

Livros:

- Learn all about React Native - Innoware PJP
- Docker: Up & Running

Sites/Documentações/Bibliotecas:

- React Native Doc, Expo doc, React navigation,Async Storage,
  Docker Docs