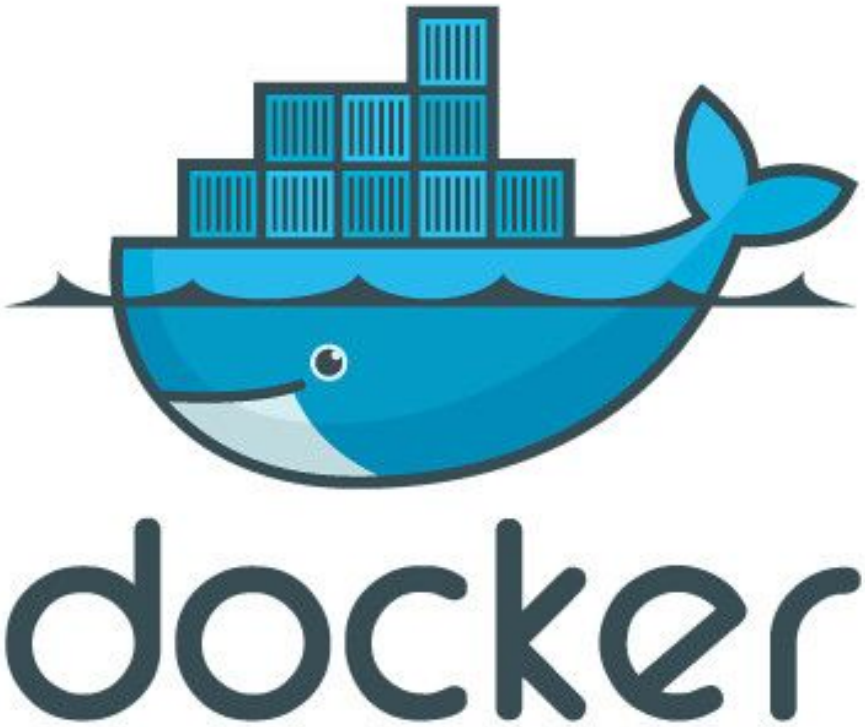
docker

O que é o Docker?

Docker é uma ferramenta projetada para facilitar a criação, a implementação e a execução de aplicativos usando contêineres.

Primeiramente temos que definir o que não é Docker. Docker não é um sistema de virtualização tradicional, enquanto em um ambiente de virtualização tradicional nós temos um Sistema Operacional completo e isolado, dentro do Docker nós temos recursos isolados que utilizando bibliotecas de kernel em comum.

Além de tudo Docker é uma plataforma Open Source escrito em Go, que é uma linguagem de programação de alto desempenho desenvolvida dentro do Google, que facilita a criação e administração de ambientes isolados.

Vantagens

- Com o uso de containers, o Docker acaba com a necessidade de executar um sistema operacional inteiro, como em uma máquina virtual por exemplo, para rodar um aplicativo, proporcionando ao usuário um ambiente portátil e com todas as necessidades básicas para sua execução.

- O Docker é perfeito para proporcionar um ambiente para a criação e compartilhamento de "Sistemas Operacionais com configurações customizadas" com alguns simples passos, e seu vasto registro de imagens e repositório de aplicações e dependências soluciona virtualmente todos os problemas relacionados à configuração de ambiente que um desenvolvedor possa ter.

- Excelente opção para se desenvolver e executar pequenas aplicações com simplicidade e maestria, garantindo aos desenvolvedores um ambiente leve e de simples configuração.

- Permite que um desenvolvedor empacote um aplicativo com todas as partes que precisa, como bibliotecas e outras dependências, e envie tudo como um único pacote, assim facilitando muito o processo para que um desenvolvedor ou até qualquer outra pessoa possa usufruir da aplicação sem muitas dificuldades, sem o problema de “A aplicação não funciona na minha máquina”.

- Uma das grandes vantagens do Docker está em resolver o problema de executar aplicativos em sistemas incapacitados, seja por incompatibilidade do sistema, ou pela falta de programas, binários, bibliotecas ou dependências necessárias para a execução do aplicativo. Assim facilitando tanto os testes de aplicações quanto o desenvolvimento também.

Desvantagens

- Uma possível desvantagem do Docker se encontra no fato de que, como este divide recursos entre outros containers e outros processos do sistema que o executa, uma aplicação que necessite por exemplo de um alto consumo de CPU terá seu desempenho prejudicado, comparado a executá-la em uma VM ou uma máquina comum.

- A comunicação entre o container e seu host, e o mapeamento de rede necessário para enviar os pacotes aos seus destinos corretos também impacta na performance de seus processos.

- Dificuldade na persistência dos dados, pois containers são elaborados para apagar totalmente seus arquivos quando são desligados, portanto qualquer armazenamento de dados para uso futuro teria que ser feito em algum outro lugar.

- O Docker foi criado com o intuito de executar aplicações de servidor que não necessitam de interfaces gráficas, portanto uma aplicação que precisa ser visualmente utilizada não seria própria para ser usada com a plataforma.

Qual é o público do Docker?

O Docker é uma ferramenta projetada para beneficiar tanto os desenvolvedores quanto os administradores do sistema, tornando-o parte de muitas ferramentas de desenvolvimento.

Instalando o Docker em seu computador:

O processo de instalação do Docker é simples, basta seguir os passos abaixo:

1º Iniciando

Primeiramente vamos atualizar o sistema para ter mais segurança e confiabilidade para a instalação do Docker. Execute os comandos abaixo em seu terminal:

$ sudo apt-get update

2º Configurar o repositório

Existem outras maneiras de instalar o docker, mas essa é a maneira mais recomendada, pois facilita as tarefas de instalação e atualização.
Instale pacotes para permitir que o apt use um repositório via HTTPS:

```
$ sudo apt-get install \ apt-transport-https \ ca-certificates \ curl \ gnupg2 \
software-properties-common
```

3º Adicionar chave GPG oficial do Docker:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4º Use o seguinte comando para configurar o repositório estável .

```
$ sudo apt-add-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
bionic stable"
```

5º Atualizar pacotes

```
$ sudo apt-get update
```

6º Instalando a versão mais recente do docker

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

7º Verificando se o Docker está instalado

```
$ docker -v
```

Caso tenha seguido esses passos e mesmo assim ocorreu erros, acesse o site do Docker (https://docs.docker.com/install/), e verifique a forma de instalação específica para para sua distribuição linux.

Agora é so criar uma conta no https://hub.docker.com/ e utilizar os comandos:

```
$ sudo su
$ docker login
```

Agora você já tem acesso a vários containers prontos, atualizações e a possibilidade de dar pull em seu containers.

Utilizando o Docker:

Para baixar uma imagem no Docker, é necessário que ela esteja no Docker Hub ou em algum outro repositório.

```
$ docker pull [nome da imagem]
```

O comando acima baixa a imagem e salva em sua máquina, assim você pode executar um ou mais containers a partir dessa imagem.

Listar todas as imagens baixadas:

```
$ docker images
```

Iniciar um container de uma imagem:

```
$ docker run [nome da imagem]
```

Há diversos parâmetros que você pode usar no comando **run**, como por exemplo o comando (-p) que indica qual porta você quer externar para que a máquina consiga fazer requisições dentro do container. Todos esses comandos podem ser vistos neste link.

Alguns comandos úteis:

```
$ docker ps # Lista os containers em execução

$ docker exec [container id] [comando] # Executa comandos dentro do container

$ docker stop $(docker ps -aq) # Para a execução de todos dos containers

$ docker rm $(docker ps -aq) # Exclui todos os containers criados
```

Docker Compose

Quando precisamos definir uma aplicação composta por diversos serviços, é necessário executar cada serviço em seu próprio container e linká-los. Isso é possível através do Docker Compose.

Executando os comandos abaixo você baixa a versão estável atual do Docker Compose e aplica as permissões executáveis ao binário:

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose
```

Utilizando o Docker Compose

No exemplo a seguir iremos criar um ambiente de desenvolvimento com PHP e MySQL no servidor Apache.

Arquivos do Projeto

O arquivo base do Docker Compose é o docker-compose.yml, onde os serviços PHP e MySQL serão linkados. No mesmo diretório, criamos o arquivo php7-apache2-dockerfile que será referenciado pelo arquivo .yml como Dockerfile para a criação e configuração do PHP e Apache. Para testarmos o ambiente de desenvolvimento, criamos o diretório html/index.php.

Então teremos a seguinte estrutura de arquivos:

docker-compose.yml
html
└── index.php
php7-apache2-dockerfile

O arquivo php7-apache2-dockerfile contém as configurações do PHP e Apache, onde em *FROM* referenciamos a versão do PHP que encontramos no repositório do Docker, não é necessário já ter a imagem do PHP baixada, o Docker tomará conta disso. O *WORKDIR* diz ao container qual será o diretório de trabalho do Apache. Precisamos, para a conexão com o MySQL, instalar o MySQLi, para que isso seja possível utilizamos o serviço *RUN,* que envia o comando de instalação para o container do PHP.

```
FROM php:7.2-apache
WORKDIR /var/www/html
RUN docker-php-ext-install mysqli
```

Em docker-compose.yml iremos linkar os serviços para a criação do ambiente de desenvolvimento, com isso temos o serviço ***apache***, onde em ***build:*** será referenciado o Dockerfile php7-apache2-dockerfile, em ***image:*** damos um nome para imagem a ser criada a partir do Dockerfile e logo após damos um nome ao container que será criado a partir dessa imagem, em ***ports:*** indicamos a porta do host que será mapeada para a porta do container e em ***volumes:*** mapeamos o diretório ./html do projeto para o diretório /var/www/html do container.

Agora para o serviço ***mysqldb*** damos um nome ao container a ser criado, podemos notar que diferente do serviço ***apache***, nós não utilizamos um Dockerfile para criar a imagem do MySQL, simplesmente referenciamos a imagem em ***image:*** com a versão encontrada no Docker Hub. Com a tag ***ports:*** Mapeamos a porta 3307 da nossa máquina real para a porta 3306 do container e em seguida enviamos as variáveis de ambiente necessárias para a criação do usuário e do banco de dados no MySQL.

```yaml
version: '3'

services:
  apache:
    build:
      dockerfile: php7-apache2-dockerfile
      context: .
    image: seu-usuario/php7-apache2-dockerfile
    container_name: php7-apache2
    restart: always
    ports:
      - '80:80'
    volumes:
      - ./html:/var/www/html
    depends_on:
      - mysqldb
    links:
      - mysqldb

  mysqldb:
    container_name: mysqlServer
    image: mysql:5.7
    restart: always
    ports:
      - '3307:3306'
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=banco
```

É no index.php onde realizamos a conexão com o banco de dados utilizando o MySQLi.

```php
<html>
<?php

echo 'Versão Atual do PHP: ' . phpversion();

$servername = "mysqlServer";
$username = "root";
$password = "root";

// Criando Conexão
$conn = new mysqli($servername, $username, $password);

// Testando Conexão
if ($conn->connect_error) {
    die("Falha na Conexão: " . $conn->connect_error);
}
echo "<br /> Conexão Bem Sucedida";
?>

</html>
```

Testando a Aplicação

Após termos entendido o funcionamento do Docker Compose, podemos agora rodar nossa aplicação. Basta executar o comando $ docker-compose up no seu terminal (é necessário estar no diretório do docker-compose.yml). Serão exibidas uma série de mensagens relacionadas ao Docker baixando as imagens dos containers pela primeira vez

Se tudo der certo, ao acessar http://localhost no seu navegador você verá:

Referências:

https://tableless.com.br/iniciando-com-o-docker-dicas-praticas-para-comecar-usar-agora-mesmo/

https://tableless.com.br/iniciando-com-o-docker-criando-suas-proprias-imagens/

https://medium.com/operacionalti/ambiente-de-desenvolvimento-php-com-docker-46e0eb2fac3d

https://docs.docker.com/docker-hub/

https://imasters.com.br/desenvolvimento/vms-vs-containers-quais-diferencas-e-usos

https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2017_2/docker/conclusion.html

https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2017_2/docker/network.html

https://opensource.com/resources/what-docker