

TUTORIAL: DOCKER

DevOps básico

Instalação, configuração e uso do Docker

Docker

O Docker oferece uma eficiente e rápida maneira de portar aplicações entre sistemas e máquinas. Ele é leve e enxuto, permitindo rapidamente conter aplicações e as executar dentro de seus próprios ambientes seguros (via Contêineres Linux: LXC).

O Docker é um dos mais interessantes e poderosos projetos open-source. Ao mesmo tempo ele é uma forma de abstração da infraestrutura que beneficia tempo de desenvolvimento quanto em produção.

O que é o Docker?

O Docker abstrai a infraestrutura das aplicações através de imagens prontas. O que temos com o uso do Docker, são máquinas virtuais de baixo custo rodando somente com a infraestrutura de aplicações.

Ex.: Uma imagem muito utilizada no Docker é a Alpine, ela contém apenas 5MB. Essa imagem contém apenas o core do Sistema Operacional, no caso Linux, que executa a maioria das aplicações. O uso dela possibilita a adição do que é necessário para instalar pacotes SDK do .NET, Node.js, PHP e, logo após isso, fazer um pacote. Com o pacote montado, pode-se enviar o pacote completo para a produção, sem risco de erros.

Docker para desenvolvedores

Embora o Docker seja voltado para infraestrutura, pode-se utilizá-lo também em ambientes de desenvolvimento, pois atualmente trabalhamos com vários bancos de dados, diversas tecnologias e serviços, isso gera um acúmulo de instalações grandes.

Para que esse acúmulo não ocorra, podemos utilizar imagens prontas de máquinas com SQL Server, MySQL e MongoDB e executá-las quando necessário.

Ex.: Para criar um projeto com SQL Server, faça o download da imagem oficial para o Docker e execute, com isso teremos o serviço rodando.

O Docker pode ser utilizado em MAC, Windows e Linux. No caso do Windows, recomenda-se a utilização da versão 10, pois esta versão tem suporte à subsistemas Linux, que são a base do Docker.

Requisitos

As questões de memória e CPU variam de acordo com a quantidade de imagens que estão sendo executadas, com isso, o disco sempre será necessário.

Embora se possa realizar downloads de imagens a qualquer momento, é muito mais prático manter as imagens locais. Assim, tem-se economia de banda e tempo, porém perde-se em disco. Logo, fica a critério do desenvolvedor manter ou baixar novamente as imagens.

As recomendações a seguir indicam que é útil, ter, ao menos um processador Dual Core e 8GB de RAM.

Em casos de discos de 128GB, com pouco espaço, pode-se excluir as imagens sempre que não estiverem em uso.

A instalação do Docker ocupa, cerca de 2GB de disco.

Obs.: Caso o sistema Operacional utilizado seja o Windows, recomenda-se o uso do Windows 10, pois ele possui suporte a subsistemas Linus, base do Docker.

Versões do Docker

Atualmente, o Docker é dividido em dois produtos: Community Edition (CE) e Enterprise Edition (EE).

O Community é gratuito, voltado a comunidades, já o Enterprise é recomendado para uso empresarial, mas para trabalhar com Docker opte pela versão Community que fornece o que é preciso, além de ser gratuita.

Criando a conta

O primeiro passo a ser feito é criar uma conta no **Docker Hub**. Esta conta será utilizada tanto para ativação do Docker nas máquinas quanto para gerenciar imagens.

Preencha todas as informações. É importante ressaltar que sua conta de e-mail será verificada, logo, após criar a conta, verifique seu e-mail para validá-la.

Instalando o Docker

O Docker possui uma versão desktop que facilita muito. É necessário utilizar as linhas de comando para a instalação do CLI.

Para instalar o Docker, [**clique aqui**](#). Vá em Download Desktop and Take a Tutorial. Este link nos leva para a página de Downloads.

Selecione a versão que atende seu Sistema Operacional, como Docker for Windows ou Docker for MAC, na tela seguinte selecione Get Docker.

Siga a instalação dos pacotes para realizar o processo e, ao término reinicie seu computador.

Obs.: Durante a instalação do Docker para Windows, você será questionado sobre o uso de contêiners Windows ao invés de Linux. Recomenda-se que está opção não fique marcada, pois o padrão de utilização é o contêiner Linux.

Executando o Docker

Após baixar o pacote do seu Sistema Operacional, instalar, reiniciar sua máquina, inicie o Docker. Para confirmar que o Docker está em execução verifique o ícone (uma baleia) próximo ao relógio (System Tray).

Realize o login, caso tenha problemas para realizar o login, clique novamente no ícone do Docker (baleia) e selecione a opção Sign In.

Com esses passos realizados teremos o Docker rodando em nossas máquinas. Recomenda-se que clique novamente no ícone do Docker (baleia) e vá até a opção Settings.

Na aba General, desmarque a opção Start Docker for Desktop when you Login – Isso fará com que o Docker não seja iniciado junto ao Sistema Operacional, deixando a inicialização mais rápida.

Testando a instalação

Ao instalar o Docker automaticamente fazemos a instalação do seu CLI, que disponibiliza o comando Docker no terminal.

Abra um terminal e digite o comando abaixo para verificar a versão do Docker instalada:

docker -version

Obs.: Se estiver utilizando Windows, recomenda-se o uso do Windows Terminal ou Power Shell.

Para exemplificar, execute o Hello World para baixar e executar um contêiner para testar se o funcionamento está correto:

docker run hello-world

Ao executar o comando, veremos a mensagem Unable to find image hello-world:latest locally, o que significa que a imagem não foi encontrada localmente.

Isso ocorre, pois, o download da imagem não foi realizado ainda. Ainda assim, ele fará uma busca online e encontrará a imagem.

Após a busca, o download e execução da imagem será realizado. Será exibida a mensagem This message shows that your installation appears to be working correctly, caso tudo tenha sido feito corretamente.

Listando as imagens

Durante a execução destes primeiros comandos, duas coisas importante ocorrem, o download da imagem e a criação do contêiner. Uma imagem é a informação crua que temos enquanto o contêiner é uma representação desta imagem local.

Os contêiners se baseiam nas imagens, mas nunca executamos as imagens e sim os contêiners. Assim, com a imagem do hello-world, temos a base par construir diversos contêiners.

Para listar as imagens locais, execute o comando:

docker image ls

Como é previsível, essas imagens ocupam espaço em disco, assim, podemos removê-las e baixá-las a qualquer momento.

Para remover uma imagem baixada, utiliza o comando:

docker image rmi ID_DA_IMAGEM

Obs.: É importante ressaltar que uma imagem não poderá ser removida ou apagada, caso esteja em uso.

Listando os contêiners

Para listar os contêiners utilize o comando abaixo:

docker container ls

Obs.: Neste caso o hello-world não aparecerá, pois, este contêiner foi criado apenas para teste. Para vê-lo listado execute o comando com -all no final:

docker container ls -all

Com este comando é possível verificar todos os contêiners que estão na máquina.

Iniciando e parando um contêiner

Para executar ou parar um contêiner, pode-se usar o comando start/stop:

docker container start ID_DO_CONTAINER

docker container stop ID_DO_CONTAINER

Para verificar a lista completa do que podemos fazer com um contêiner execute o comando:

docker container -help

Removendo um contêiner

Com o contêiner parado, pode-se removê-lo utilizando o comando abaixo:

docker container rm ID_DO_CONTAINER

Obs.: Após a execução deste comando pode-se remover a imagem baixada.

Publicando uma API

Para publicar uma API criada utilizando Node/JavaScript em um contêiner e torná-la acessível pelo Browser, utilize o GIT e o Visual Studio Code. Abra um terminal e navegue para a pasta segura, sem caracteres especiais ou com o caminho. Execute o comando:

git clone https://github.com/nomedousuarioqualquer/docker-sample-api.git

Este comando fará o download do código fonte de uma API de exemplo que está no Git Hub, gerando assim, uma nova pasta chamada docker-sample-api.

Docker File

Com o código baixado, gere um contêiner para a aplicação, e neste processo informar alguns detalhes.

O Docker File é o arquivo de instruções que o Docker utilizará para saber como publicar uma aplicação em um Contêiner, bem como qual imagem utilizar.

Assim, abra a pasta docker-sample-api com o Visual Studio Code e crie um arquivo chamado Dockerfile na raiz.

Note que este arquivo não tem extensão, apenas Dockerfile. Logo, o primeiro passo a ser feito neste arquivo é definir qual imagem será utilizada na aplicação.

Vale ressaltar que a imagem é a base para a construção, contendo apenas o necessário do Sistema Operacional para executar a aplicação.

Como exemplo, utilize a [**imagem oficial do Node para Docker**](#). Neste link temos imagens oficiais de diversas empresas do Docker Hub.

Para definir que estamos utilizando uma imagem como base, utilizamos a palavra FROM, seguida da imagem que queremos utilizar:

FROM node:current-slim

Observe que após o nome da imagem temos dois pontos e current-slim, O que vem após os dois pontos é a TAG da imagem.

As TAGs são utilizadas para definir versões de imagens, o current dos que estamos pegando a versão mais atual do Node (LTS - Long Term Support) e slim que é a versão enxuta.

Cada imagem tem suas TAGs, por isso, olhe no Docker Hub a versão que precisa utilizar.

Após isso, devemos definir o caminho do arquivo, dentro da imagem, onde ficará o código fonte da aplicação. Isso é feito utilizando a palavra WORKDIR:

WORKDIR /usr/src/app

É necessário copiar o arquivo Package.json, que contém, os pacotes necessários para executar a API.

Obs.: No caso do Node, não é necessário instalar uma SDK ou algo do tipo, porém toda aplicação tem a pasta node_modules com as bibliotecas que foram utilizadas durante a construção da API.

Para executar esta cópia, utilize a palavra COPY:

COPY package.json .

Note que no comando há um ‘‘.‘‘, definindo que o arquivo será copiado para a raiz da aplicação definido no WORKDIR.

Agora, execute o comando NPM INSTALL para instalar todos os pacotes necessários, via terminal. Isto é feito com a palavra RUN:

RUN npm install

O código da aplicação é copiado para a imagem e em seguida os pacotes são instalados para que a API rode.

Porém é necessário definir em qual porta está API rodará, para isso utiliza a palavra EXPOSE:

EXPOSE 8080

Assim, estamos expondo a API na porta 8080, porém pode-se utilizar outra porta.

É importante ressaltar que a API, internamente (dentro da imagem) roda na porta 3000 - Isto está definido na linha 15 do arquivo bin/www.

Ainda assim, embora tenhamos copiado os arquivos e instalado os pacotes, a API ainda não está executando. Para isso execute o comando NPM START. Utilize a palavra CMD para passar uma lista de comandos a serem executados:

CMD [‘npm’, ‘start’]

Agora, execute mais um COPY para copiar o resto da aplicação com a imagem:

COPY ..

Observe que utilizamos o mesmo esquema anterior, copiando os arquivos da raiz (‘.’) da aplicação para a raiz (‘.’) da imagem.

O arquivo final Dockerfile fica com o conteúdo:

Copy the file from your host to your current location

COPY package.json .

Run the command inside your image filesystem

RUN npm install

Inform Docker that the container is listening on the specified porta at runtime.

EXPOSE 8080

Run the specified command within the container.

CMD [“npm”, “start”]

Copy the rest of your app’s source code from your host to your image filesystem.

COPY ..

OBS.: Ao fazer Control + C, Control + V, verifique as aspas, pois, caso não sejam alteradas manualmente o script ocorrerá erro.

Criando uma imagem

Para criar uma imagem customizada a partir desta API/Dockerfile junte a imagem padrão definida no Dockerfile com o restante da execução do script.

Dentre os comandos que podemos executar, temos o docker image build, para criar uma imagem.

Certifique-se de que estas na raiz da API, na pasta docker-sample-api e execute o comando:

docker image build -t dockerapi:1.0 .

Para facilitar futuras execuções, dê um nome para esta imagem, adicionando o parâmetro -t, seguido pelo nome:versão da imagem.

Agora, defina o diretório onde está o Dockerfile, no caso a raiz da API, definido no comando pelo “.”.

Para testar, basta executar o comando docker image ls para listar as imagens que estão na máquina.

Rodando o contêiner

Agora, pegue a imagem gerada e execute ela como um contêiner. Faça isso com o comando docker container run.

Certifique-se que você está na raiz da API, na pasta docker-sample-api e execute o comando:

```
docker container run --publish 8080:3000 --detach --name api dockerapi:1.0
```

O primeiro parâmetro do comando é o --publish, que vai fazer o direcionamento do tráfego para a porta onde a API está rodando.

Vamos rodar o contêiner na porta 8080, ou seja, acessaremos a API no Browser pela URL http://localhost:8080, porém, internamente a API estará rodando na porta 3000.

O --publish cuidará exatamente deste direcionamento de tráfego. Lembre-se que a porta 8080 pode ser alterada, como, por exemplo 8181.

O --detach diz para o Docker executar este contêiner em background, e não trava o processo no terminal ou Visual Studio Code que está executando.

O --name especifica o nome do contêiner – coloque o nome que quiser – desde que não tenha espaços ou caracteres especiais.

Agora, abra o Browser no endereço http://localhost:8080 para ver a mensagem ‘Docker Node API Running’ na tela, isso significa que a API está rodando.

Por fim, temos o nome da imagem especificado ou que acabamos de criar. Use o comando docker container ls para ver este contêiner em execução, além do docker container start ID e docker container stop ID para iniciar ou parar ele.

Acessando os logs

Par visualizar os logs de execução da API clique no ícone do Docker, na opção Dashboard. Uma nova tela se abrirá listando o contêiner que foi criado recentemente.

Clique sobre o Contêiner e em seguida clique na opção LOGS para visualizar o que está sendo executado neste contêiner.

PUCRS online  **uol**edtech