// Tutorial //

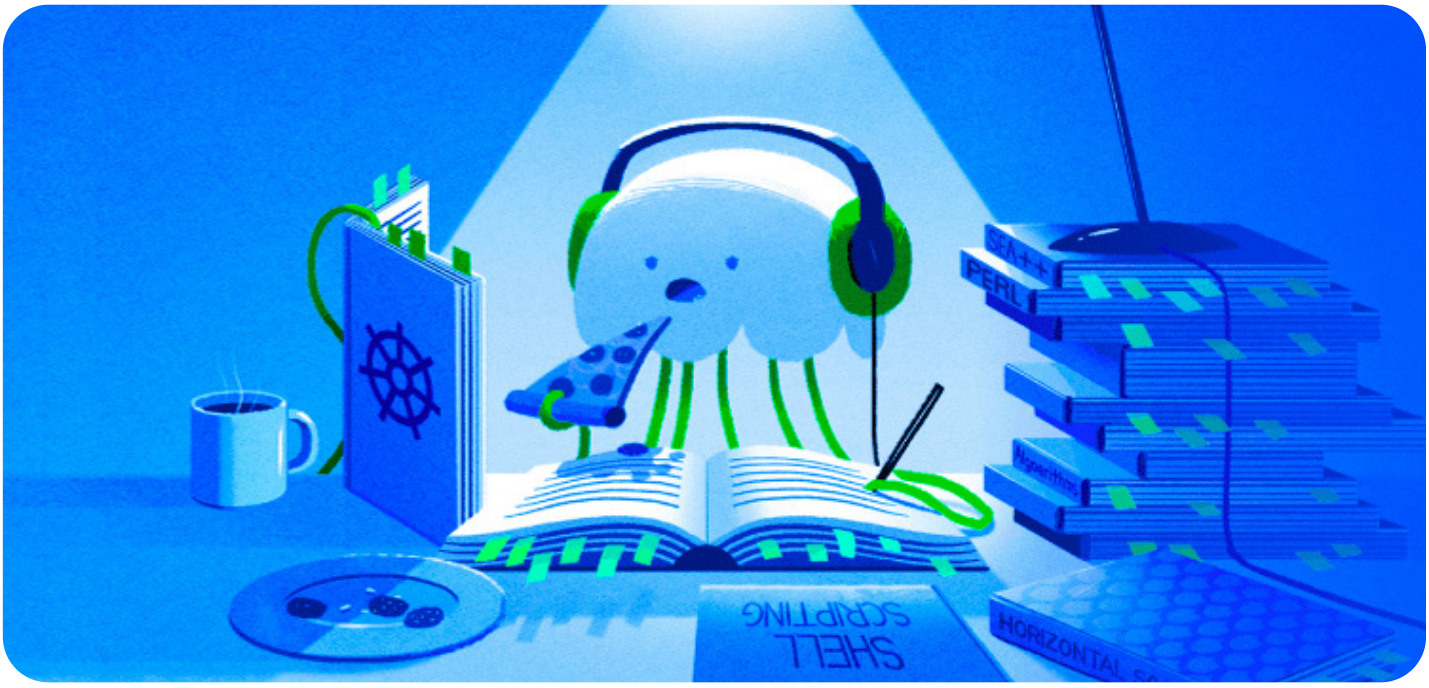
Como instalar o Docker Compose no Debian 10

Published on March 19, 2020

Docker | Debian 10 | Debian | Container

By Kathleen Juell and Brian Hogan

Português



Introdução

O Docker é uma ótima ferramenta para automatizar a implantação de aplicativos Linux dentro de contêineres de software, mas para aproveitar de todo esse potencial, cada componente de um aplicativo deve executar em seu próprio contêiner individual. Para aplicativos complexos com vários componentes, orquestrar todos os contêineres para iniciar, comunicar e fechar ao mesmo tempo pode se tornar algo rapidamente complicado.

A comunidade do Docker surgiu com uma solução popular chamada Fig, que permitiu o uso de um arquivo YAML único para orquestrar todos os seus contêineres e configurações do Docker. Ele se tornou tão popular que a equipe do Docker decidiu fazer o Docker Compose, com base na fonte Fig, que agora foi descontinuada. O Docker Compose permite que os usuários orquestrem os processos dos contêineres do Docker, incluindo a inicialização, o desligamento e a configuração de ligação intracontêineres e de volumes.

Neste tutorial, você irá instalar a versão mais recente do Docker Compose para ajudá-lo a gerenciar aplicativos multicontêineres em um servidor Debian 10.

Pré-requisitos

Para seguir este artigo, você irá precisar do seguinte:

- Um servidor Debian 10 e um usuário não raiz com privilégios sudo. Este tutorial de configuração inicial do servidor com o Debian 10 explica como configurar isso.
- O Docker instalado com as instruções do **Passo 1** e do **Passo 2** do tutorial sobre Como instalar e utilizar o Docker no Debian 10

Nota: embora os pré-requisitos dêem instruções para instalar o Docker no Debian 10, os comandos do `docker` neste artigo devem funcionar em outros sistemas operacionais, desde que o Docker esteja instalado.

Passo 1 — Instalando o Docker Compose

Embora possa instalar o Docker Compose a partir dos repositórios oficiais do Debian, ele está várias versões menores atrás do lançamento mais recente. Neste tutorial ele será instalado a partir do repositório do GitHub do Docker. O comando seguinte é ligeiramente diferente daquele que você encontrará na página dos Lançamentos . Use o sinalizador `-o` para especificar o arquivo de saída primeiro, em vez de redirecionar a saída. Essa sintaxe evita que você se depare com um erro de "permissão negada", causada ao usar o comando `sudo`.

Verifique o lançamento atual e, se necessário, atualize-o com o seguinte comando:

```
sudo curl -L https://github.com/docker/compose/releases/download/1.25.3/docker-
```

Em seguida, vamos definir as permissões:

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

Então, vamos verificar se a instalação foi bem-sucedida, verificando a versão:

```
docker-compose --version
```

Isto mostará na tela a versão que instalamos:

```
Output
docker-compose version 1.25.3, build d4d1b42b
```

Agora que temos o Docker Compose instalado, estamos prontos para executar um exemplo "Hello World".

Passo 2 — Executando um contêiner com o Docker Compose

O registro público do Docker, o Docker Hub, inclui uma imagem do *Hello World* para demonstração e teste. Ele ilustra a configuração mínima necessária para executar um contêiner utilizando o Docker Compose: um arquivo YAML que chama uma única imagem. Criaremos essa configuração mínima para executar nosso contêiner `hello-world`.

Primeiramente, crie um diretório para o arquivo YAML e troque para ele:

```
mkdir hello-world
cd hello-world
```

Copy

Depois, crie o arquivo YAML:

```
nano docker-compose.yml
```

Copy

Coloque o seguinte conteúdo no arquivo, salve o arquivo e saia do editor de texto:

docker-compose.yml

```
my-test:
 image: hello-world
```

Copy

A primeira linha no arquivo YAML é usada como parte do nome do contêiner. A segunda linha especifica qual imagem usar para criar o contêiner. Quando executarmos o comando `docker-compose up`, ele procurará uma imagem local pelo nome que especificamos, no caso, o `hello-world`. Com isso funcionando, vamos salvar e sair do arquivo.

É possível ver as imagens manualmente no nosso sistema com o comando `docker images`:

```
docker images
```

Copy

Quando não houverem imagens locais, apenas os títulos das colunas são exibidos:

```
Output
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
```

Enquanto ainda estiver no diretório `~/hello-world` , execute o seguinte comando:

```
docker-compose up
```

Copy

A primeira vez que executar o comando, se não houver uma imagem local chamada `hello-world`, o Docker Compose irá buscá-la do repositório público do Docker Hub:

```
Output
Pulling my-test (hello-world:)...
latest: Pulling from library/hello-world
9db2ca6ccae0: Pull complete
Digest: sha256:4b8ff392a12ed9ea17784bd3c9a8b1fa3299cac44aca35a85c90c5e3c7afacdc
Status: Downloaded newer image for hello-world:latest
. . .
```

Após puxar a imagem, o `docker-compose` cria um contêiner, anexa e executa o programa hello que, por sua vez, confirma que a instalação parece estar funcionando:

```
Output
. . .
Creating helloworld_my-test_1...
Attaching to helloworld_my-test_1
my-test_1 |
my-test_1 | Hello from Docker.
my-test_1 | This message shows that your installation appears to be working correctly.
my-test_1 |
. . .
```

Em seguida, ele mostra na tela uma explicação do que ele fez:

```
Output
 To generate this message, Docker took the following steps:
my-test_1  |  1. The Docker client contacted the Docker daemon.
my-test_1  |  2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
my-test_1  |     (amd64)
my-test_1  |  3. The Docker daemon created a new container from that image which runs the
my-test_1  |     executable that produces the output you are currently reading.
my-test_1  |  4. The Docker daemon streamed that output to the Docker client, which sent i
my-test_1  |     to your terminal.
```

Os contêineres do Docker apenas executam enquanto o comando estiver ativo. Portanto, assim que o `hello` terminar de executar, o contêiner será interrompido. Consequentemente, quando olharmos os processos ativos, os cabeçalhos das colunas irão aparecer, mas o contêiner `hello-world` não estará listado porque ele não estará em execução:

```
docker ps
```

Copy

```
Output
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
```

É possível ver as informações do contêiner, que serão necessárias no próximo passo, usando o sinalizador `-a`. Isso mostra todos os contêineres, não apenas os ativos:

```
docker ps -a
```

Copy

```
Output
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
06069fd5ca23        hello-world         "/hello"            35 minutes ago      Exited (0)
```

Isso mostra as informações que precisará remover do contêiner quando terminar com ele.

Passo 3 — Removendo a imagem (opcional)

Para evitar o uso desnecessário do espaço em disco, vamos remover a imagem local. Para fazer isso, precisaremos excluir todos os contêineres que referem a imagem, utilizando o comando `docker rm`, seguido ou do `CONTAINER ID` ou do `NAME`. No exemplo a seguir, estamos usando o `CONTAINER ID` a do comando `docker ps -a` que acabamos de executar. Certifique-se de substituir a ID do seu contêiner:

```
docker rm 06069fd5ca23
```

Copy

Assim que todos os contêineres que referenciam a imagem tiverem sido removidos, poderemos remover a imagem:

```
docker rmi hello-world
```

Copy

Conclusão

Você instalou o Docker Compose no Debian 10, testou sua instalação - executando um exemplo do Hello World - e removeu a imagem de teste e o contêiner.

Embora o exemplo do Hello World confirmou sua instalação, a configuração simples não mostrou um dos principais benefícios do Docker Compose — poder trazer um grupo de contêineres do Docker para cima e para baixo, todos ao mesmo tempo. Para ver como usar o Docker Compose mais detalhadamente, leia o tutorial sobre Como instalar o WordPress com o Docker Compose.