

Como Instalar e Usar o Docker no Ubuntu 16.04

Posted February 21, 2017  83k DOCKER UBUNTU UBUNTU 16.04



By: finid

Introdução

O Docker é uma aplicação que torna simples e fácil executar processos de aplicação em um contêiner, que é como uma máquina virtual, apenas mais portátil, mais amigável, e mais dependente do sistema operacional do host. Para uma introdução detalhada aos diferentes componentes do contêiner Docker, confira [The Docker Ecosystem: An Introduction to Common Components](#).

Existem dois métodos para a instalação do Docker no Ubuntu 16.04. Um dos métodos envolve instalá-lo em uma instalação existente do sistema operacional. A outra envolve lançar um servidor com uma ferramenta chamada [Docker Machine](#) que instala o Docker automaticamente nele.

Nesse tutorial, vamos aprender como instalá-lo e utilizá-lo em uma instalação existente do Ubuntu 16.04.

Pré-requisitos

Para seguir esse tutorial, você vai precisar do seguinte:

- Droplet do Ubuntu 16.04 64 bits
- Usuário não-root com privilégios sudo ([Initial Setup Guide for Ubuntu 16.04](#) explica como configurar isto).

Nota: O Docker requer uma versão de 64 bits do Ubuntu, bem como uma versão de kernel maior ou igual a 3.10. O Droplet padrão de 64 bits do Ubuntu 16.04 atende a esses requisitos.

Todos os comandos nesse tutorial devem ser executados como usuário não-root. Se o acesso de root for requerido para o comando, ele será precedido pelo `sudo`. [Initial Setup Guide for Ubuntu 16.04](#) explica como adicionar usuários e dar a eles o acesso ao sudo.

Passo 1 — Instalando o Docker

O pacote de instalação do Docker disponível no repositório oficial do Ubuntu 16.04 pode não ser a última versão. Para obter a maior e mais recente versão, instale o Docker a partir do repositório oficial do Docker. Essa seção lhe mostra como fazer isso.

Mas primeiro, vamos atualizar o banco de dados de pacotes:

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

Agora, vamos instalar o Docker. Adicione ao sistema a chave GPG oficial do repositório do Docker:

```
$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C07
```

Adicione o repositório do Docker às fontes do APT:

```
$ sudo apt-add-repository 'deb https://apt.dockerproject.org/repo ubuntu-xenial main'
```

Atualize o banco de dados de pacotes com os pacotes do Docker a partir do novo repositório adicionado:

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

Certifique-se de que você está instalando a partir do repositório do Docker em vez do repositório padrão do Ubuntu 16.04:

```
$ apt-cache policy docker-engine
```

Você deverá ver uma saída semelhante à seguinte:

Output of apt-cache policy docker-engine

```
docker-engine:
  Installed: (none)
  Candidate: 1.11.1-0~xenial
  Version table:
     1.11.1-0~xenial 500
        500 https://apt.dockerproject.org/repo ubuntu-xenial/main amd64 Packages
     1.11.0-0~xenial 500
        500 https://apt.dockerproject.org/repo ubuntu-xenial/main amd64 Packages
```

Observe que o `docker-engine` não está instalado, mas o candidato para instalação é do repositório Docker do Ubuntu 16.04. O número da versão do `docker-engine` pode ser diferente.

Finalmente, instale o Docker:

```
$ sudo apt-get install -y docker-engine
```

O Docker agora será instalado, o daemon iniciado, e o processo habilitado para iniciar no boot. Verifique que ele está executando:

```
$ sudo systemctl status docker
```

A saída deve ser similar ao seguinte, mostrando que o serviço está ativo e em execução:

```
[ secondary_label Output]
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2016-05-01 06:53:52 CDT; 1 weeks 3 days ago
     Docs: https://docs.docker.com
   Main PID: 749 (docker)
```

A instalação do Docker lhe fornece não apenas o serviço Docker (daemon) mas também o utilitário de linha de comando `docker`, ou o cliente Docker. Vamos explorar como utilizar o comando `docker` mais tarde nesse tutorial.

Passo 2 — Executando o Comando Docker sem Sudo (Opcional)

Por padrão, executar o comando `docker` requer privilégios de root - isto é, você tem de prefixar o comando com `sudo`. Ele também pode ser executado por um usuário no grupo **docker**, que é automaticamente criado durante a instalação do Docker. Se você tentar executar o comando `docker` sem prefixá-lo com `sudo` ou sem ser do grupo `docker`, você obterá uma saída semelhante a essa:

```
Output
docker: Cannot connect to the Docker daemon. Is the docker daemon running on this host?.
See 'docker run --help'.
```

Se você quiser evitar digitar `sudo` sempre que você executar o comando `docker`, adicione o seu usuário ao grupo `docker`:

```
$ sudo usermod -aG docker $(whoami)
```

Você precisará sair do Droplet e voltar novamente com o mesmo usuário para habilitar essa alteração.

Se você precisar adicionar um usuário com o qual você não está logado ao grupo `docker`, declare aquele usuário explicitamente usando:

```
$ sudo usermod -aG docker username
```

O restante desse artigo assume que você está executando o comando `docker` como um usuário do grupo `docker`. Se você escolher não fazer dessa forma, por favor prefixe os comandos com `sudo`.

Passo 3 — Utilizando o Comando Docker

Com o Docker instalado e funcionando, agora é hora de começar a familiarizar-se com o utilitário de linha de comando. Utilizar o `docker` consiste em passar a ele uma cadeia de opções seguida de argumentos. A sintaxe assume essa forma:

```
$ docker [option] [command] [arguments]
```

Para ver todos os subcomandos disponíveis, digite:

```
$ docker
```

A partir do Docker 1.11.1, a lista completa de subcomandos disponíveis inclui:

Output

<code>attach</code>	Attach to a running container
<code>build</code>	Build an image from a Dockerfile
<code>commit</code>	Create a new image from a container's changes
<code>cp</code>	Copy files/folders between a container and the local filesystem
<code>create</code>	Create a new container
<code>diff</code>	Inspect changes on a container's filesystem
<code>events</code>	Get real time events from the server
<code>exec</code>	Run a command in a running container
<code>export</code>	Export a container's filesystem as a tar archive
<code>history</code>	Show the history of an image
<code>images</code>	List images
<code>import</code>	Import the contents from a tarball to create a filesystem image
<code>info</code>	Display system-wide information
<code>inspect</code>	Return low-level information on a container or image
<code>kill</code>	Kill a running container
<code>load</code>	Load an image from a tar archive or STDIN
<code>login</code>	Log in to a Docker registry
<code>logout</code>	Log out from a Docker registry
<code>logs</code>	Fetch the logs of a container
<code>network</code>	Manage Docker networks
<code>pause</code>	Pause all processes within a container
<code>port</code>	List port mappings or a specific mapping for the CONTAINER
<code>ps</code>	List containers
<code>pull</code>	Pull an image or a repository from a registry
<code>push</code>	Push an image or a repository to a registry

rename	Rename a container
restart	Restart a container
rm	Remove one or more containers
rmi	Remove one or more images
run	Run a command in a new container
save	Save one or more images to a tar archive
search	Search the Docker Hub for images
start	Start one or more stopped containers
stats	Display a live stream of container(s) resource usage statistics
stop	Stop a running container
tag	Tag an image into a repository
top	Display the running processes of a container
unpause	Unpause all processes within a container
update	Update configuration of one or more containers
version	Show the Docker version information
volume	Manage Docker volumes
wait	Block until a container stops, then print its exit code

Para ver as chaves disponíveis para um comando específico, digite:

```
$ docker subcomando-docker --help
```

Para ver informações globais de sistema sobre o Docker, utilize:

```
$ docker info
```

Passo 4 — Trabalhando com Imagens do Docker

Os contêineres Docker são executados a partir de imagens Docker. Por padrão, ele puxa essas imagens do Docker Hub, um cadastro de imagens gerenciado pela Docker, a empresa por trás do projeto Docker. Qualquer um pode construir e hospedar imagens Docker no Docker Hub, assim muitas aplicações e distribuições Linux que você vai precisar para rodar o Docker tem imagens que são mantidas no Docker hub.

Para verificar se você pode acessar e baixar imagens a partir do Docker hub, digite:

```
$ docker run hello-world
```

A saída, que deve incluir o que se segue, deve indicar que o Docker está executando corretamente:

```
[secondary-label Output]
```

```
Hello from Docker.
```

```
This message shows that your installation appears to be working correctly.
```

```
...
```

Você pode procurar por imagens disponíveis no Docker Hub utilizando o comando `docker` com o subcomando `search` . Por exemplo, para procurar por uma imagem Ubuntu, digite:

```
$ docker search ubuntu
```

O script vai rastrear o Docker Hub e retornar uma listagem de todas as imagens cujo nome corresponde à string de pesquisa. Nesse caso, a saída será semelhante à seguinte:

Output

NAME	DESCRIPTION	STARS	OFFICIAL
ubuntu	Ubuntu is a Debian-based Linux operating s...	3808	[OK]
ubuntu-upstart	Upstart is an event-based replacement for ...	61	[OK]
torusware/speedus-ubuntu	Always updated official Ubuntu docker imag...	25	
rastasheep/ubuntu-sshd	Dockerized SSH service, built on top of of...	24	
ubuntu-debootstrap	debootstrap --variant=minbase --components...	23	[OK]
nickistre/ubuntu-lamp	LAMP server on Ubuntu	6	
nickistre/ubuntu-lamp-wordpress	LAMP on Ubuntu with wp-cli installed	5	
nuagebec/ubuntu	Simple always updated Ubuntu docker images...	4	
nimmis/ubuntu	This is a docker images different LTS vers...	4	
maxexcloo/ubuntu	Docker base image built on Ubuntu with Sup...	2	
admiringworm/ubuntu	Base ubuntu images based on the official u...	1	
...			

Na coluna **OFICIAL**, **OK** indica uma imagem construída e suportada pela empresa por trás do projeto. Uma vez que você tenha identificado a imagem que você gostaria de usar, você pode baixá-la para seu computador utilizando o subcomando `pull` , dessa forma:

```
$ docker pull ubuntu
```

Depois de uma imagem ter sido baixada, você pode então executar um contêiner usando a imagem com o subcomando `run` . Se uma imagem não tiver sido baixada quando o `docker` é executado com o subcomando `run` , o cliente Docker irá primeiro baixar a imagem, depois executar um contêiner utilizando-a:

```
$ docker run ubuntu
```

Para ver as imagens que foram baixadas em seu computador, digite:

```
$ docker images
```

A saída deve se parecer com algo assim:

Output

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	c5f1cf30c96b	7 days ago	120.8 MB
hello-world	latest	94df4f0ce8a4	2 weeks ago	967 B

Como você verá adiante nesse tutorial, imagens que você utiliza para executar contêineres podem ser modificadas e utilizadas para gerar novas imagens, que podem então ser carregadas (pushed é o termo técnico) para o Docker Hub ou outro registro Docker.

Passo 5 — Executando um Contêiner Docker

O contêiner `hello-world` que você executou previamente é um exemplo de contêiner que executa e sai, depois de emitir uma mensagem de teste. Os contêineres, contudo, podem ser muito mais úteis do que isso, e eles podem ser interativos. Afinal, eles são semelhantes às máquinas virtuais, apenas com recursos mais amigáveis.

Como um exemplo, vamos executar um contêiner utilizando a última imagem do Ubuntu. A combinação das chaves `-i` e `-t` fornece a você acesso ao shell interativo dentro do contêiner:

```
$ docker run -it ubuntu
```

Seu prompt de comando deve mudar para refletir o fato de que você está agora trabalhando dentro do contêiner e deve assumir essa forma:

Output

```
root@d9b100f2f636:/#
```

Importante: Observe o id do contêiner no prompt de comando. Nesse exemplo, ele é `d9b100f2f636`.

Agora você pode executar qualquer comando dentro do contêiner. Por exemplo, vamos atualizar o banco de dados de pacotes dentro do contêiner. Não é necessário prefixar qualquer comando com `sudo`, porque você está operando dentro do contêiner com privilégios de root:

```
$ apt-get update
```

A seguir instale qualquer aplicação nele. Vamos instalar o NodeJS, por exemplo.

```
$ apt-get install -y nodejs
```

Passo 6 — Confirmando Alterações em um Contêiner para uma Imagem Docker

Quando você inicia uma imagem Docker, você pode criar, modificar, e deletar arquivos da mesma forma que você faz em uma máquina virtual. As alterações que você realiza somente serão aplicadas àquele contêiner. Você pode iniciá-lo e pará-lo, mas uma vez que você o destrói com o comando `docker rm`, as alterações serão perdidas para sempre.

Essa seção mostra como salvar o estado de um contêiner como uma nova imagem Docker.

Após a instalação do `nodejs` dentro do contêiner Ubuntu, você tem agora um contêiner executando uma imagem, mas o contêiner é diferente da imagem que você utilizou para criá-lo.

Para salvar o estado do contêiner como uma nova imagem, primeiro saia dele:

```
$ exit
```

Depois, confirme as alterações para uma nova instância de imagem Docker utilizando o seguinte comando. A chave `-m` é para a mensagem que ajuda você e outras pessoas saberem quais alterações você fez, enquanto que a chave `-a` é utilizada para especificar o autor. O ID do contêiner é o que você anotou anteriormente no tutorial quando você iniciou a sessão interativa de docker. A menos que você tenha criado repositórios adicionais no Docker Hub, o repositório é geralmente o seu nome de usuário do Docker Hub:

```
$ docker commit -m "What did you do to the image" -a "Author Name" id-do-contêiner repositório/nome_
```

Por exemplo:

```
$ docker commit -m "added node.js" -a "Sunday Ogwu-Chinuwa" d9b100f2f636 finid/ubuntu-nodejs
```

Nota: Quando você *confirma* ou faz *commit* de uma imagem, a nova imagem é salva localmente, isto é, no seu computador. Posteriormente nesse tutorial, você irá aprender como carregar uma imagem para um registro Docker como o Docker Hub de forma que ela possa ser acessada e utilizada por você e por outras pessoas.

Após a conclusão dessa operação, a listagem de imagens Docker no seu computador agora deve mostrar a nova imagem, bem como a antiga de onde ela se derivou:

```
$ docker images
```

A saída deve ser semelhante a essa:

Output

finid/ubuntu-nodejs	latest	62359544c9ba	50 seconds ago	206.6 MB
---------------------	--------	--------------	----------------	----------

ubuntu	latest	c5f1cf30c96b	7 days ago	120.8 MB
hello-world	latest	94df4f0ce8a4	2 weeks ago	967 B

No exemplo acima, **ubuntu-nodejs** é a nova imagem, que foi derivada da imagem ubuntu existente no Docker Hub. A diferença no tamanho reflete as alterações que foram realizadas. E nesse exemplo, a alteração foi que o NodeJS foi instalado. Assim, da próxima vez que você precisar executar um contêiner utilizando Ubuntu com NodeJS pré-instalado, você pode simplesmente utilizar a nova imagem. Imagens podem também ser construídas a partir do que é chamado Dockerfile. Mas esse é um processo mais avançado que está fora do escopo deste artigo.

Passo 7 — Listando Contêineres Docker

Após a utilização do Docker por um tempo, você terá muitos contêineres ativos (executando) e inativos em seu computador. Para visualizar **os ativos**, utilize:

```
$ docker ps
```

Você verá uma saída semelhante à seguinte:

Output

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	
f7c79cc556dd	ubuntu	"/bin/bash"	3 hours ago	Up 3 hours	P

Para ver todos os contêineres - ativos e inativos, passe ao comando a chave **-a**:

```
$ docker ps -a
```

Para ver o último contêiner que você criou, passe ao comando a chave **-l**:

```
$ docker ps -l
```

A paralisação de um contêiner executando ou ativo é tão simples como digitar:

```
$ docker stop id-do-contêiner
```

O `container-id` pode ser encontrado na saída do comando `docker ps`.

Step 8 — Carregando Imagens Docker para um Repositório Docker

O próximo passo lógico depois da criação de uma nova imagem a partir de uma imagem existente é compartilhá-la com alguns de seus amigos, com o mundo inteiro no Docker Hub, ou com outro registro Docker que você tenha acesso. Para carregar uma imagem para o Docker Hub ou qualquer outro registro Docker, você deve possuir uma conta lá.

Essa seção lhe mostra como carregar uma imagem para o Docker Hub. Para aprender como criar seu próprio registro Docker particular, confira [How To Set Up a Private Docker Registry on Ubuntu 14.04.](#)

Para criar uma conta no [Docker Hub](#), registre-se no Docker Hub. Posteriormente, para carregar a sua imagem, primeiro faça o login no Docker Hub. Você será solicitado a se autenticar:

```
$ docker login -u username-do-registro-docker
```

Se você especificou sua senha corretamente, a autenticação será realizada com sucesso. Então, você poderá carregar sua própria imagem utilizando:

```
$ docker push username-do-registro-docker/nome-da-imagem-docker
```


Levará algum tempo para completar, e quando estiver completo, a saída será semelhante à seguinte:

Output

```
The push refers to a repository [docker.io/finid/ubuntu-nodejs]
e3fbbfb44187: Pushed
5f70bf18a086: Pushed
a3b5c80a4eba: Pushed
7f18b442972b: Pushed
3ce512daaf78: Pushed
7aae4540b42d: Pushed

...
```

Após carregar uma imagem para um registro, ela deve ser listada no painel de sua conta, como aquele mostrado na imagem abaixo:


DashboardExploreOrganizations

finid

RepositoriesStarsContributed

Repositories

Type to filter repositories by name

 <div><div>finid/ubuntu-nodejs</div><div>public</div></div>	0 STARS	2 PULLS	> DETAILS
--	------------	------------	--------------

Se uma tentativa de envio resultar em um erro desse tipo, então você provavelmente não fez login:

```
Output
The push refers to a repository [docker.io/finid/ubuntu-nodejs]
e3fbbfb44187: Preparing
5f70bf18a086: Preparing
a3b5c80a4eba: Preparing
7f18b442972b: Preparing
3ce512daaf78: Preparing
7aae4540b42d: Waiting
unauthorized: authentication required
```

Faça login, depois tente o envio novamente.

Conclusão

Há muito mais sobre o Docker do que o que foi tratado nesse artigo, mas isso deve ser suficiente para você começar a trabalhar com ele no Ubuntu 16.04. Assim como a maioria dos projetos *open source*, o Docker é construído a partir de uma base de código de desenvolvimento rápido, assim tenha o hábito de visitar a [página do blog do projeto](#) para as últimas informações.

Adicionalmente confira [os outros tutoriais de Docker na comunidade DigitalOcean](#).



Editor:
Tammy Fox



Translation:
Fernando Pimenta



We just made it easier for you to deploy faster.

[TRY FREE](#)

Related Tutorials

[How To Install Docker Compose on Ubuntu 18.04](#)

[How To Remove Docker Images, Containers, and Volumes](#)

[Naming Docker Containers: 3 Tips for Beginners](#)

[Экосистема Docker: распределение задач \(Scheduling\) и оркестровка \(Orchestration\)](#)

[Экосистема Docker: обзор контейнеризации](#)

2 Comments

Leave a comment...

[Log In to Comment](#)

^ [daniloturquet](#) *October 27, 2017*

0 Sensacional!

^ [moneytime](#) *June 12, 2018*

0 Olá, este tutorial parece não estar mais funcional, atualizaram a forma de inalação, e no site do Docker é bem confuso. Teriam um novo tutorial simplificado nos moldes deste?



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2019 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)