Comandos mais utilizados no Docker

Comandos mais utilizados durante o Workflow com Docker

11/Abr/2016 — 7 minutos de leitura

Editar este artigo

Se este conteúdo te ajudar, considere apoiar meu trabalho através do Apoia.se

---

Um post rapidinho com os comandos mais utilizados do Docker para não esquecer e consultar facilmente. :D

Se você não viu a primeira parte dessa sequência, clique aqui .

E a segunda parte, aqui

Sumário

Caso você queira pular para algum comando específico.

- Como eu sei quais as imagens disponíveis no meu repositório local?
- Como adicionar imagens locais?
- Como remover imagens locais?
- Criar um container
- Criar um container e entrar no Terminal
- Criar um container com um apelido
- Verificar o estado ou encontrar o ID de um container
- Remover um container
- Outras informações úteis que o Docker pode nos passar sobre o container
- Commitar alterações em uma imagem
- Mapeando uma porta para o container
- Montar containers auto destrutivos
- Executando containers em segundo plano
- Removendo todos os containers e imagens de uma só vez
- Como executar comandos, sem precisar estar dentro do container?

Como eu sei quais as imagens disponíveis no meu repositório local?

Utilize o comando **`images`** para listar:

```
docker images
```

Será apresentado uma tabela no seu Terminal com:

- **REPOSITORY** - repositório
- **TAG** - a tag utilizada no repositório (é setado pelo mantenedor)
- **IMAGE ID** - o ID da Imagem
- **CREATED** - quando essa imagem foi criada
- **SIZE** - tamanho dessa imagem

Como adicionar imagens locais?

Utilize o comando **`search`** para procurar a imagem e o comando **`pull`** para baixar:

```
docker search ubuntu
```

Encontrado a imagem correta, utilize **`pull`** com o nome dessa imagem:

```
docker pull ubuntu
```

Como remover imagens locais?

Localize o ID ou nome do repositório com o comando **`docker images`**.

Com o id ou o nome do repositório em mãos, utilize o comando **`rmi`** para

excluir as imagens.

```
docker rmi ID_ou_nome_da_imagem
```

Criar um container

Para executar um container utilize o comando **`run`** com o nome da imagem que vá utilizar para a criação:

```
docker run nome_da_imagem
```

Criar um container e entrar no Terminal

Conseguimos criar um container e já entrar nesse container com o comando **`-it`**:

```
docker run -it ubuntu /bin/bash
```

Vai subir um container com o Ubuntu e entrar no Bash.

O **`-i`** significa interatividade e o **`-t`** que queremos um link com o Terminal do container.

Criar um container com um apelido

Você pode colocar apelidos nos containers para facilitar sua organização passando por parâmetro o **`--name`** para o comando **`docker run`**:

```
docker run --name ubuntinho ubuntu
```

Perceba que logo depois do parâmetro **`--name`** vem o nome que deseja e o nome da imagem que vai ser utilizada para gerar o container.

Nesse caso a imagem **`ubuntu`** e o alias **`ubuntinho`**.

Verificar o estado ou encontrar o ID de um container

Você consegue uma lista dos containers ativos com o comando **`ps`**:

```
docker ps
```

Vai aparecer uma tabela com

- **CONTAINER ID** - ID do container
- **IMAGE** - a imagem que foi utilizada para gerar esse container
- **COMMAND** - o comando passado como parâmetro para esse container (exemplo o /bin/bash)
- **CREATED** - a data da criação do container

- **STATUS** - o estado do container (parado ou em funcionamento)
- **PORTS** - as portas compartilhadas entre host e container
- **NAMES** - e o nome que você deu ao container, se o fez

O `ps` só vai mostrar os containers que estão em atividade, para verificar todos os containers criados, incluindo os que estiverem parados, utilize o `ps -a`:

```
docker ps -a
```

Para pegar apenas o ID do container do topo da tabela, utilize o comando `ps -qa`

Remover um container

Remover um container seria o mesmo que desligar a máquina virtual.

Utilize o comando `rm` para remover o container com o ID que você pode pegar com o `docker ps` ou o apelido que você escolheu:

```
docker rm id_ou_apelido
```

Outras informações úteis que o Docker pode nos passar sobre o container

Informações de uso de Hardware do container:

```
docker stats id_ou_apelido
```

Veremos informações como:

- **CONTAINER** - ID do Container
- **CPU %** - uso de CPU em porcentagem
- **MEM USAGE / LIMIT** - Memória usada/Limite que você pode ter setado
- **MEM** - uso de memória em porcentagem
- **NET I/O** - I/O de Internet
- **BLOCK IO** - Outros processos de I/O.

```
docker inspect id_ou_apelido
```

Esse comando trás muita informação útil, então é bom dar uma olhada na documentação oficial para não se perder pelas linhas!

Commitar alterações em uma imagem

As alterações que você faz em um container, durante sua execução, não são salvas, a menos que você gere uma nova imagem com base nesse container.

Para commitar o que você fez em uma imagem, utilize o comando `commit`

`commit`:

```
docker commit ID/apelido nome_da_nova_imagem
```

Ele vai gerar uma nova imagem a partir desse commit.

Mapeando uma porta para o container

Usamos o comando `-p`:

```
docker run -it -p 8080:80 ubuntu
```

Bem útil para listar uma porta para um servidor web:

```
docker run -it -p 8080:80 nginx
```

Estamos informando que a porta 8080 no Host é aberta e deve ser mapeada na porta 80 do container.

Montar containers auto destrutivos

Usando o comando `--rm`, podemos montar containers que se destroem ao sairmos da sessão.

Exemplo utilizando o NGINX .

```
docker run -it --rm -p 8080:80 nginx /bin/bash
```

Ao usar um **`exit`** para sair do Terminal do SO rodando no container, o mesmo será removido.

Executando containers em segundo plano

Podemos executar o container e deixar ele em segundo plano, sem precisar ficar conectado pelo Shell, com o comando **`-d`**.

Exemplo utilizando o NGINX.

```
docker run -d -p 8080:80 nginx /usr/sbin/nginx -g
```

Para controlar esse container usamos os comandos **`stop`** e **`start`**:

```
docker stop id_ou_apelido
```

Para parar e:

```
docker start id_ou_apelido
```

Para ativar o container.

Removendo todos os containers e imagens de uma só vez

Usamos um pouquinho de Shell Script e conseguimos automatizar o processo de remoção de todos os containers ativos com:

```
docker rm $(docker ps -qa)
```

Para entender o que o `$(docker ps -qa)` está fazendo, execute somente esse comando no Terminal e veja o retorno.

```
$(docker ps -qa)
```

Para imagens

```
docker rmi $(docker images -q)
```

Como executar comandos, sem precisar estar dentro do container?

Para não precisar acessar um container para executar comandos basicos,

podemos usar o **exec**:

```
docker exec -it id_ou_apelido comando
```

Ex.:

```
docker exec -it ubuntinho ifconfig eth0
```

Vai retornar o endereço de IP do container.

Esses são os comandos mais básicos para sobreviver os primeiros dias com o Docker, depois vai ficar fixado na cabeça e é só alegria.

Nos próximos artigos vou falar sobre como utilizar o Dockerfile para automatizar a criação de containers e como criar sua própria imagem com as suas configurações.

Não perca.

Gostou? Comenta aqui em baixo.

Acha que da para incrementar com mais comandos legais?

Abre uma issue , comenta aqui em baixo ou adiciona direto no Blog pra mim!

Espalhe a palavra.

Este conteúdo te ajudou?

Se eu consegui te ajudar, considere contribuir com o meu trabalho através dos links abaixo.

Qualquer valor é muito bem vindo e os apoios começam a partir de 1 real.

Apoiar via Apoia.se

Apoiar via PicPay

Apoiar via PayPal

Espalhe a palavra!

Compartilhe este artigo nas redes sociais clicando nos ícones.

Tags: docker infraestrutura linux

William Oliveira

Inclusão social através do ensino de programação e front-end

Desenvolvido com Eleventy e Hylia Eleventy Starter Kit v0.7.0.