



Comandos básicos do Docker

Este documento explora os comandos básicos do Docker, uma plataforma de containers essencial para o desenvolvimento moderno. Abordamos conceitos fundamentais como imagens e containers, além de comandos práticos para gerenciar ambientes Docker. O guia inclui exemplos, explicações detalhadas e um exercício prático para consolidar o aprendizado.

editado por Everton Coimbra de Araújo

O que é Docker e sua importância

Docker é uma plataforma de containers que permite empacotar uma aplicação e suas dependências em um ambiente isolado e portátil. Ele ajuda a garantir que a aplicação funcione da mesma forma em qualquer sistema, resolvendo o famoso problema do "funciona na minha máquina, mas não no servidor". Docker é amplamente utilizado para criar ambientes consistentes de desenvolvimento e produção, simplificar o gerenciamento de dependências e tornar o processo de deploy mais eficiente.

Cada comando que veremos a seguir será explicado em detalhes, com exemplos práticos e forneceremos também as razões pelas quais cada comando é importante, além dos pontos positivos e negativos de seu uso. Assim, nosso objetivo é ensinar de maneira prática e didática como você pode utilizar o Docker para otimizar seu trabalho e melhorar suas habilidades no desenvolvimento de software.

O que é uma Imagem Docker e um Container?

Imagem Docker

Uma imagem Docker é um pacote que contém tudo o que é necessário para executar uma aplicação: código, dependências, bibliotecas, variáveis de ambiente e instruções para a execução. É como um "modelo" a partir do qual os containers são criados. As imagens são imutáveis, o que significa que não podem ser modificadas depois de criadas. Isso garante consistência e confiabilidade durante o desenvolvimento e o deploy.

Por exemplo, uma imagem pode ser de um servidor web como **Nginx**, um banco de dados como **MySQL** ou até mesmo uma aplicação específica que você desenvolveu. As imagens são armazenadas em **registries**, como o **Docker Hub**, onde você pode buscar imagens públicas ou armazenar as suas próprias.

Container

Um container é uma instância de uma imagem em execução. Ele pode ser visto como um ambiente de execução isolado, criado a partir de uma imagem. Imagine que a imagem seja a receita e o container seja o prato preparado; um container é, portanto, a materialização da imagem em um ambiente que pode ser executado.

Os containers são leves e compartilham o kernel do sistema operacional do host, o que os torna mais eficientes do que máquinas virtuais. Cada container criado a partir de uma imagem é independente e pode ser manipulado individualmente, proporcionando uma maneira ágil e escalável de rodar aplicações.



Comandos Docker: docker run e docker ps

1

docker run hello-world

O comando `docker run hello-world` é utilizado para baixar e executar uma imagem de teste chamada `hello-world`. Ele é uma excelente maneira de verificar se a instalação do Docker foi bem-sucedida e se está funcionando corretamente. Quando você executa esse comando, ele baixa a imagem do Docker Hub (caso ainda não esteja armazenada localmente) e cria um container a partir dessa imagem. O Docker Hub é um serviço de registro de imagens onde os desenvolvedores podem compartilhar e armazenar imagens Docker. Ele é essencial para facilitar o acesso a imagens pré-configuradas e para compartilhar suas próprias com a comunidade ou em equipes.

2

docker ps

Outro comando muito útil é o `docker ps`. Esse comando permite listar todos os containers em execução no momento. Com ele, você pode monitorar quais containers estão ativos, além de obter informações como o ID do container, o nome e o status. Esse é um comando fundamental para gerenciar o seu ambiente Docker, permitindo visualizar o que está acontecendo no sistema e monitorar containers ativos. No entanto, ele não mostra containers que já foram parados, limitando um pouco a visão geral do que foi executado.

3

docker ps -a

Para obter uma visão mais completa, o comando `docker ps -a` é essencial. Ele lista todos os containers, incluindo aqueles que não estão em execução. Isso significa que você pode ver o histórico completo de containers que foram criados, seja para diagnosticar problemas, seja para saber quais containers ainda estão armazenados no sistema, mas não estão ativos. Isso pode ser extremamente útil ao depurar falhas ou ao tentar entender a sequência de operações que foi realizada. Por outro lado, quando há muitos containers, a listagem pode ficar bastante extensa e difícil de analisar, exigindo o uso de filtros para melhorar a organização.

Comandos Docker: Executando e gerenciando o Nginx

Levante um servidor web com um único comando

Quando você precisa de um servidor web rápido e fácil de configurar, o comando **docker run nginx** é a solução perfeita. Com apenas uma linha de código, você baixa e executa a imagem do servidor web Nginx em um container Docker. Seu servidor web já está no ar, pronto para receber tráfego na porta 80 padrão. É uma abordagem incrível para testes e desenvolvimento, permitindo que você se concentre no que realmente importa.

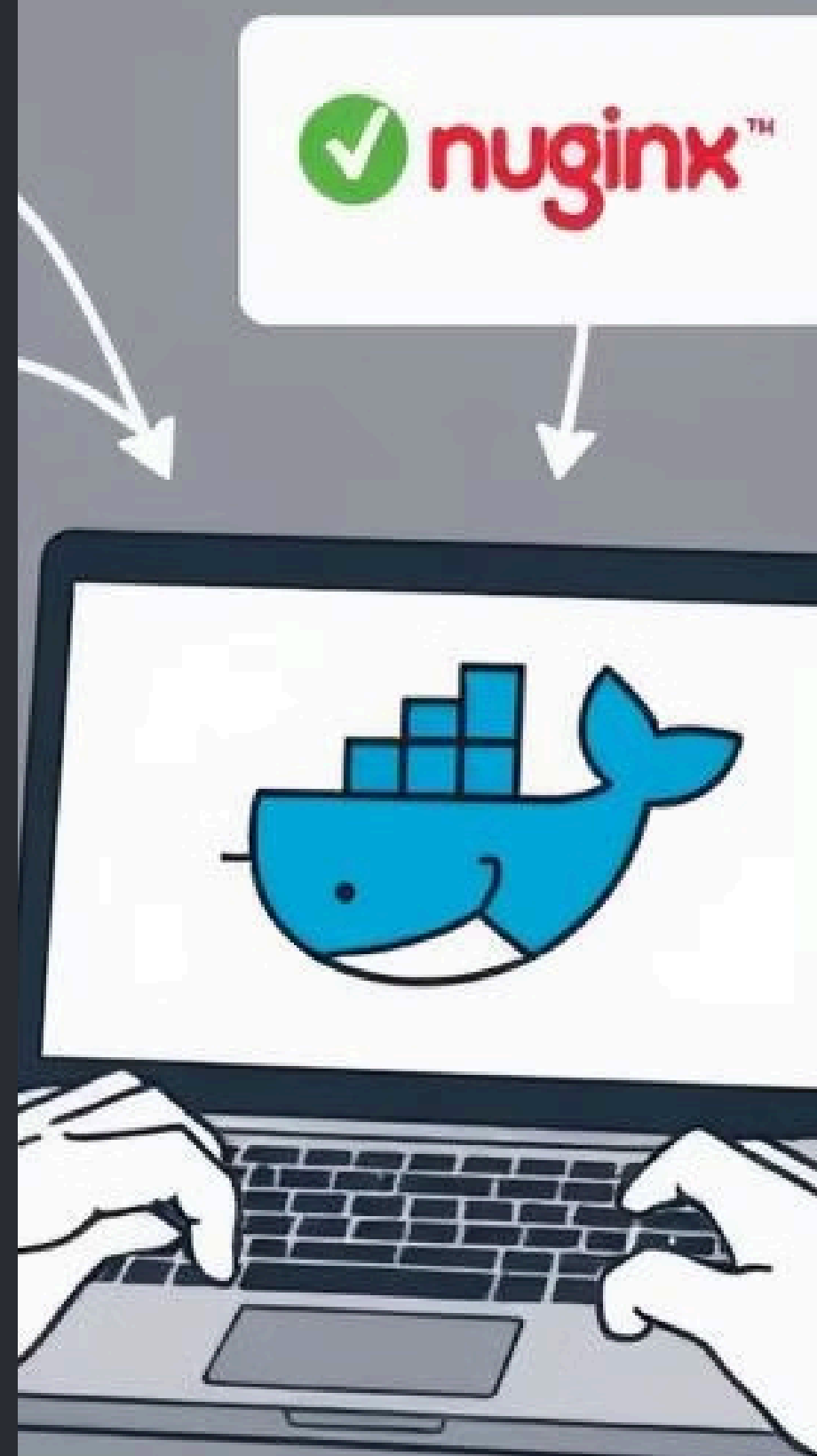
Evite conflitos de porta com mapeamento personalizado

Às vezes, pode haver outros serviços rodando em sua máquina na porta 80. Nesse caso, o comando **docker run -p 8082:80 nginx** é sua solução. Ele mapeia a porta 8082 do seu host para a porta 80 do container Nginx. Agora você pode acessar seu servidor web em **http://localhost:8082** sem nenhum conflito. Essa é uma técnica poderosa para executar múltiplos serviços em um único host, mantendo tudo organizado e isolado.

Execute o servidor web em segundo plano

Quando você precisa manter o servidor web Nginx em execução, mas também liberar o terminal para outras tarefas, o comando **docker run -d -p 8082:80 nginx** é a resposta. Ele roda o container em modo desanexado (-d), o que significa que o processo continua rodando em segundo plano. Você pode continuar usando o terminal para outras atividades, enquanto seu servidor web Nginx continua atendendo requisições na porta 8082. Embora você não veja os logs diretamente no terminal, é possível acessá-los a qualquer momento com o comando **docker logs**.

inial ce proger
tee mesalof te



Comandos Docker: Gerenciando containers

`docker stop <container_id>`

Depois de ter containers em execução, pode chegar um momento em que você precise parar um container. O comando `docker stop <container_id>` é a forma mais segura de fazer isso, pois envia um sinal para que o container seja encerrado de maneira adequada. O container é parado, mas continua existindo no sistema, o que significa que pode ser reiniciado a qualquer momento. Essa é uma abordagem útil quando você precisa liberar recursos, mas não quer perder o estado do container. Contudo, é importante lembrar que ele ainda ocupa espaço no disco.

`docker rm <container_id>`

Finalmente, para limpar containers que não são mais necessários, o comando `docker rm <container_id>` é utilizado para removê-los do sistema permanentemente. Isso é muito importante para liberar espaço e manter seu ambiente Docker organizado. É uma boa prática remover containers que não serão mais utilizados, para evitar acúmulo desnecessário. No entanto, uma vez removido, o container não pode ser recuperado, e todos os dados armazenados nele serão perdidos.



`docker start <container_id>`

Caso precise reiniciar um container que foi previamente parado, o comando `docker start <container_id>` permite fazer isso sem precisar recriar o container. Isso economiza tempo e mantém todas as configurações e dados intactos. A principal vantagem aqui é a rapidez com que você pode colocar o container de volta em operação. No entanto, o estado do container no momento em que foi parado pode não ser completamente restaurado, especialmente se ele estiver no meio de uma tarefa.

Comandos Docker: Operações avançadas

Limpeza Radical

Imagine ter um container teimoso, que simplesmente se recusa a parar. Nesse caso, o comando **docker rm -f <container_id>** é sua solução definitiva. Esse comando força a remoção do container, encerrando qualquer processo em execução e garantindo que ele seja removido do sistema. Embora essa seja uma medida drástica, às vezes é necessário adotar uma abordagem mais firme para lidar com containers problemáticos ou travados. No entanto, é importante lembrar que, uma vez removido, o container e todos os seus dados serão perdidos para sempre.

Reinício Rápido


Quando um container precisa de uma "reinicialização", o comando **docker restart <container_id>** é a solução perfeita. Diferente de ter que parar e iniciar o container manualmente, o restart faz tudo isso em uma única etapa. Isso é especialmente útil quando um serviço parou de responder ou você precisa aplicar pequenas alterações que não exigem a recriação completa do container. Com apenas um comando, você pode ter seu container de volta em ação, pronto para atender suas necessidades.

Monitorando Atividades

Para acompanhar o que está acontecendo dentro de um container, especialmente quando ele está rodando em segundo plano, o comando **docker logs <container_id>** é sua melhor ferramenta. Esse comando exibe os logs de saída do container, permitindo que você acompanhe o comportamento da sua aplicação em tempo real. Você pode adicionar a flag **-f** para seguir os logs continuamente, como se estivesse monitorando a atividade do container de forma ininterrupta. Essa é uma funcionalidade essencial para diagnosticar problemas e entender o que está acontecendo dentro de seus ambientes Docker.

Conclusão e FAQ

Nesta seção, vimos os principais comandos básicos do Docker, desde a execução de um container com `docker run` até o gerenciamento de containers com `docker ps`, `docker stop`, `docker rm` e outros. Esses comandos são fundamentais para o uso eficiente do Docker e ajudam a entender como criar, gerenciar e remover containers. A prática desses comandos permitirá a criação de ambientes de desenvolvimento consistentes e o gerenciamento eficaz de aplicações containerizadas.



O que é Docker?

Docker é uma plataforma de containers que permite empacotar uma aplicação e suas dependências em um ambiente isolado.





Imagem vs Container

Uma imagem é um pacote que contém tudo o necessário para executar uma aplicação, enquanto um container é uma instância em execução dessa imagem.



Liberando espaço

Use `docker rm <container_id>` para remover containers e `docker rmi <image_id>` para remover imagens não utilizadas.

Exercício Prático

Para consolidar o conhecimento adquirido, siga as instruções abaixo para realizar alguns exercícios práticos com Docker:

1. Execute o comando `docker run hello-world` para verificar se a instalação do Docker está funcionando corretamente.
2. Crie um container Nginx usando `docker run -d -p 8080:80 nginx` e acesse-o no navegador através de `http://localhost:8080`.
3. Liste todos os containers em execução usando `docker ps` e, em seguida, pare o container Nginx com `docker stop <container_id>`.
4. Reinicie o container Nginx utilizando `docker start <container_id>` e verifique se ele está novamente acessível no navegador.
5. Remova o container Nginx usando `docker rm <container_id>` e confirme que ele não está mais presente utilizando `docker ps -a`.

Resolução Comentada dos Exercícios:

1. **Verificação do Docker:** O comando `docker run hello-world` baixará e executará uma imagem de teste que exibirá uma mensagem no terminal. Isso confirma que a instalação do Docker está funcionando corretamente.
2. **Criando um container Nginx:** O comando `docker run -d -p 8080:80 nginx` criará um container em segundo plano, que executará o servidor web Nginx, mapeando a porta 8080 do host para a porta 80 do container. Ao acessar `http://localhost:8080`, você verá a página padrão do Nginx, indicando que o container está funcionando.
3. **Listando e parando o container:** Com o comando `docker ps`, você poderá ver o ID do container Nginx. Em seguida, use `docker stop <container_id>` para parar o container, interrompendo o servidor web.
4. **Reiniciando o container:** O comando `docker start <container_id>` reiniciará o container parado, e você poderá acessar novamente o servidor Nginx pelo navegador para verificar se ele está funcionando corretamente.
5. **Removendo o container:** Para liberar recursos, use `docker rm <container_id>` e, em seguida, `docker ps -a` para confirmar que o container foi removido do sistema.

