Aula 20 – Gerenciando Contêineres Docker

Imagine que você está construindo uma casa. Você tem todos os materiais, as ferramentas e o projeto. Mas, para que a casa funcione, você precisa gerenciar cada etapa da construção, desde o levantamento das paredes até a instalação elétrica e hidráulica. No mundo do desenvolvimento de software, os contêineres Docker são como essas casas pré-fabricadas: eles vêm com tudo o que uma aplicação precisa para rodar, isolada e pronta para uso. No entanto, assim como uma casa, um contêiner precisa ser gerenciado em seu ciclo de vida.

Nesta aula, nosso foco é desmistificar o gerenciamento de contêineres Docker, transformando você em um arquiteto capaz de orquestrar essas "casas" digitais com maestria. Você aprenderá a iniciar, parar, reiniciar e remover contêineres, além de entender como eles interagem com o mundo exterior e como podemos "espiar" o que acontece dentro deles. Ao final, você não apenas compreenderá os comandos essenciais, mas também a lógica por trás de cada ação, preparando-o para construir e manter ambientes de desenvolvimento e produção robustos e eficientes.

Nosso percurso começará com o ciclo de vida básico de um contêiner, explorando comandos fundamentais como docker run, docker stop, docker start e docker rm. Em seguida, mergulharemos nas nuances de execução, diferenciando os modos interativo e destacado. Abordaremos a crucial tarefa de mapear portas para expor seus serviços e, por fim, aprenderemos a inspecionar e visualizar os logs dos contêineres, ferramentas indispensáveis para a depuração e monitoramento. Prepare-se para uma jornada prática que solidificará seu conhecimento em Docker.

O Ciclo de Vida de um Contêiner: Do Nascimento à Despedida

Quando pensamos em software, muitas vezes imaginamos algo estático, um programa que simplesmente "existe". Contudo, no universo dos contêineres, cada aplicação é uma entidade viva, com um ciclo de vida bem definido. Assim como um ser vivo nasce, cresce, interage e, eventualmente, cessa suas atividades, um contêiner Docker passa por estágios que precisam ser compreendidos e controlados. Dominar esses estágios é o primeiro passo para se tornar um verdadeiro especialista em orquestração de aplicações.

01

Criação e Inicialização

O contêiner nasce a partir de uma imagem e começa sua execução

02

Execução de Tarefas

O contêiner realiza suas funções, processando dados e servindo aplicações

03

Pausa ou Reinício

O contêiner pode ser temporariamente interrompido e retomado conforme necessário

04

Remoção

O contêiner é desligado e removido do sistema, liberando recursos

A jornada de um contêiner começa com sua criação e inicialização, passa pela execução de suas tarefas, pode ser pausada ou reiniciada, e culmina em sua remoção. Essa analogia com a vida nos ajuda a entender que o gerenciamento não é apenas sobre digitar comandos, mas sobre entender o estado atual do seu ambiente e o que você deseja que ele faça. É como ser o maestro de uma orquestra, onde cada instrumento (contêiner) precisa ser iniciado, tocado e silenciado no momento certo para que a sinfonia (sua aplicação) seja perfeita.

Comando Fundamental: docker run

Este comando não apenas cria uma nova instância de um contêiner a partir de uma imagem, mas também a inicia. É o equivalente a "ligar" uma máquina virtual, mas de forma muito mais leve e rápida.

Vamos começar com o comando que dá vida a um contêiner: docker run. Quando você executa docker run, o Docker verifica se a imagem existe localmente; se não, ele a baixa do Docker Hub (ou de outro registro) e, em seguida, cria e inicia o contêiner.

```
# Exemplo básico de como iniciar um contêiner Ubuntu e executar um comando
docker run ubuntu echo "Olá do contêiner Ubuntu!"
```

Neste exemplo, o Docker baixa a imagem ubuntu (se ainda não a tiver), cria um contêiner a partir dela, executa o comando echo "Olá do contêiner Ubuntu!" dentro dele e, uma vez que o comando termina, o contêiner para e é removido automaticamente (a menos que especifiquemos o contrário). É uma operação rápida e eficiente, ideal para tarefas pontuais ou para testar imagens.

Dando Vida e Interagindo: docker run em Detalhes

O comando docker run é o coração do gerenciamento de contêineres, mas sua simplicidade inicial esconde uma vasta gama de possibilidades. Ele não apenas inicia um contêiner, mas define como ele se comportará, como interagirá com você e com o sistema operacional hospedeiro. Compreender suas opções é fundamental para criar ambientes flexíveis e adequados às suas necessidades, seja para desenvolvimento interativo ou para serviços em segundo plano.

Pense no docker run como o controle remoto universal para seus contêineres. Com ele, você pode não só ligar a TV, mas também ajustar o volume, mudar de canal e até programar gravações. No contexto do Docker, isso significa escolher se você quer interagir diretamente com o contêiner (como se estivesse dentro dele) ou se prefere que ele execute suas tarefas silenciosamente em segundo plano, sem a necessidade de sua atenção constante.

Modo Interativo

Para quando você precisa de uma sessão de terminal dentro do contêiner, ideal para depuração e exploração

Modo Destacado

Para serviços que rodam em segundo plano, liberando seu terminal para outras tarefas

Executando Contêineres em Modo Interativo (-it)

O modo interativo é ideal para quando você precisa de uma sessão de terminal dentro do contêiner, como se estivesse acessando uma máquina remota via SSH. É perfeito para depurar, instalar pacotes, ou simplesmente explorar o ambiente do contêiner.

Opção -i (interactive)

Mantém o STDIN aberto, permitindo que você envie comandos para o contêiner.

Opção -t (tty)

Aloca um pseudo-TTY, essencial para interagir com o shell do contêiner de forma eficaz.

```
# Exemplo de execução interativa de um contêiner Ubuntu
docker run -it ubuntu bash
```

Dica de Navegação

Para sair do contêiner sem pará-lo, use **Ctrl+P, Ctrl+Q**. Para sair e pará-lo, digite **exit**.

Ao executar este comando, você será levado para o prompt de comando dentro do contêiner Ubuntu. Você pode então executar comandos como ls, apt update, ping google.com, etc.

Contêineres em Segundo Plano e o Gerenciamento Essencial

Continuando nossa exploração do docker run, agora vamos focar no modo que a maioria dos serviços em produção utiliza: o modo destacado. Este modo é crucial para aplicações que precisam rodar continuamente sem a necessidade de interação humana direta, liberando seu terminal para outras tarefas.

Executando Contêineres em Modo Destacado (-d)

Quando você executa um contêiner em modo destacado, ele roda em segundo plano, liberando seu terminal imediatamente. É como iniciar um serviço em seu computador que não precisa de uma janela aberta para funcionar. Isso é ideal para servidores web, bancos de dados e outras aplicações de longa duração.

Opção -d (detached)

Inicia o contêiner em segundo plano e imprime o ID do contêiner

```
# Exemplo de execução destacada de um contêiner Nginx
docker run -d --name meu-servidor-web nginx
```

Após executar este comando, o Docker iniciará um contêiner Nginx em segundo plano e você verá uma longa sequência de caracteres, que é o ID do contêiner. Seu terminal estará livre para você continuar trabalhando. Para verificar se o contêiner está rodando, você pode usar docker ps.

Parando e Iniciando Contêineres: docker stop e docker start

Contêineres não são imutáveis; eles precisam ser controlados. Assim como você pode pausar e retomar um filme, você pode parar e iniciar seus contêineres.

docker stop

Envia um sinal SIGTERM para o processo principal, aguarda 10 segundos e então envia SIGKILL se necessário

docker start

Inicia um contêiner que foi previamente parado, retomando do estado em que foi parado

```
# Parar o contêiner Nginx que criamos
docker stop meu-servidor-web

# Iniciar o contêiner Nginx novamente
docker start meu-servidor-web
```

Esses comandos são essenciais para gerenciar a disponibilidade de seus serviços. Se um contêiner precisa ser atualizado ou configurado, você pode pará-lo, fazer as alterações (se for o caso de volumes, que veremos na próxima aula), e depois iniciá-lo novamente.

Removendo Contêineres: docker rm e o Ciclo Completo

A vida útil de um contêiner, por mais útil que seja, eventualmente chega ao fim. Seja porque ele cumpriu sua função, foi substituído por uma versão mais nova, ou simplesmente porque você precisa liberar recursos, a remoção é uma etapa crucial no gerenciamento. Manter o ambiente limpo e organizado é uma boa prática, evitando o acúmulo de contêineres inativos que consomem espaço e recursos desnecessariamente.

Pense na remoção de um contêiner como a reciclagem de um objeto. Você não joga fora algo que ainda pode ser útil, mas também não guarda lixo. Contêineres parados são como objetos guardados no sótão: não estão sendo usados, mas ocupam espaço. O comando docker rm é a sua ferramenta para essa "limpeza" digital, garantindo que apenas o que é necessário permaneça ativo ou disponível para reinício.

Removendo Contêineres: docker rm

Importante

Você só pode remover contêineres que estejam parados. Se um contêiner estiver em execução, você precisará pará-lo primeiro.

docker rm [ID_ou_Nome]

Remove o contêiner especificado de forma segura

docker rm -f [ID_ou_Nome]

Força a remoção de um contêiner em execução (use com cautela!)

```
# Parar e remover o contêiner Nginx
docker stop meu-servidor-web
docker rm meu-servidor-web

# Ou, para remover um contêiner em execução (não recomendado em produção)
# docker rm -f meu-servidor-web
```

Para remover todos os contêineres parados de uma vez, você pode usar um comando combinado:

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

Onde docker ps -aq lista todos os IDs de contêineres (incluindo os parados) de forma silenciosa.

Resumo do Ciclo de Vida do Contêiner

docker run	Cria e inicia um novo contêiner a partir de uma imagem	Criado, Rodando
docker stop	Envia sinal de parada, desliga o contêiner	Parado
docker start	Inicia um contêiner parado	Rodando
docker rm	Remove um contêiner (deve estar parado)	Removido

Com esses comandos, você tem controle total sobre o ciclo de vida de seus contêineres, desde a sua criação até a sua remoção definitiva. Isso é fundamental para manter a agilidade e a eficiência em seus projetos.

Mapeamento de Portas: Conectando o Contêiner ao Mundo Exterior

Um contêiner, por sua natureza, é isolado. Ele tem seu próprio sistema de arquivos, seus próprios processos e, crucialmente, suas próprias portas de rede. Se você tem uma aplicação web rodando dentro de um contêiner na porta 80, como o mundo exterior (seu navegador, por exemplo) consegue acessá-la? A resposta está no mapeamento de portas, uma funcionalidade essencial que atua como uma ponte entre o contêiner e a máquina hospedeira.

Máquina Hospedeira

O "prédio" onde os contêineres residem, com portas públicas acessíveis

Contêiner

O "apartamento" isolado com suas próprias portas internas

Mapeamento

A "ponte" que conecta a porta pública à porta interna do contêiner

Imagine seu contêiner como um apartamento em um grande prédio (a máquina hospedeira). Seu apartamento tem um número interno (a porta do contêiner, digamos, 80). Para que alguém de fora do prédio possa te visitar, você precisa de um número de interfone que esteja listado na portaria (a porta da máquina hospedeira, digamos, 8080) e que direcione para o seu apartamento. O mapeamento de portas faz exatamente isso: ele "publica" uma porta do contêiner para uma porta específica na máquina hospedeira.

Sem o mapeamento de portas, sua aplicação dentro do contêiner seria como uma ilha isolada, incapaz de se comunicar com o restante da rede ou com os usuários. Isso é útil para segurança e isolamento, mas inviabiliza a maioria das aplicações que precisam interagir com o mundo. O mapeamento de portas é a chave para transformar essa ilha em um ponto de acesso vital.

Como Funciona o Mapeamento de Portas (-p)

O mapeamento de portas é configurado usando a opção -p (ou --publish) com o comando docker run. A sintaxe básica é **[porta_hospedeira]:[porta_contêiner]**.

porta_hospedeira

A porta na máquina onde o Docker está rodando. É por esta porta que você acessará o serviço.

porta_contêiner

A porta interna do contêiner onde a aplicação está escutando.

```
# Exemplo: Mapear a porta 8080 da máquina hospedeira para a porta 80 do contêiner Nginx
docker run -d -p 8080:80 --name meu-servidor-web-publico nginx
```

Como Acessar

Após executar este comando, o servidor Nginx dentro do contêiner estará escutando na porta 80. Para acessá-lo do seu navegador, digite **http://localhost:8080** (se estiver rodando localmente).

O Docker se encarrega de encaminhar o tráfego da porta 8080 do seu host para a porta 80 do contêiner.

Mapeamento de Portas: Flexibilidade e Boas Práticas

A capacidade de mapear portas oferece grande flexibilidade, permitindo que você execute múltiplos contêineres de uma mesma imagem, cada um expondo seu serviço em uma porta diferente da máquina hospedeira. Isso é extremamente útil em ambientes de desenvolvimento, onde você pode precisar testar diferentes versões de uma aplicação ou rodar serviços que usam a mesma porta interna, mas precisam ser acessíveis externamente.

Imagine que você tem dois servidores web Nginx, mas cada um serve um site diferente. Ambos os contêineres, por padrão, escutariam na porta 80 internamente. Sem o mapeamento de portas flexível, você não conseguiria rodá-los simultaneamente e acessá-los externamente. O Docker resolve isso permitindo que você escolha portas hospedeiras distintas para cada um, como se cada apartamento no prédio tivesse um número de interfone único, mesmo que internamente todos os apartamentos tenham a mesma "porta da frente" dentro de suas unidades.

Opções Avançadas de Mapeamento de Portas

Mapeamento de porta aleatória

Se você não especificar a porta hospedeira, o Docker escolherá uma porta aleatória e disponível.

```
docker run -d -p 80 --name meu-servidor-aleatorio nginx
# Para ver qual porta foi mapeada:
docker port meu-servidor-aleatorio 80
```

Mapeamento para um IP específico

Você pode especificar um endereço IP da máquina hospedeira para o qual a porta será mapeada.

```
docker run -d -p 192.168.1.100:8080:80 --name meu-servidor-ip nginx
```

Mapeamento de portas UDP

Para serviços que usam UDP (como DNS ou VPNs).

```
docker run -d -p 53:53/udp --name meu-dns-server bind9
```

Conexão com GitOps

O mapeamento de portas é uma funcionalidade poderosa que garante que seus serviços contêinerizados sejam acessíveis quando necessário, mantendo o isolamento quando não. É um pilar para a construção de arquiteturas de microserviços e para a implantação de aplicações em ambientes de produção. A prática de mapear portas de forma consistente e documentada é uma das bases para a implementação de **GitOps**, onde a configuração de infraestrutura é versionada e aplicada automaticamente.

Inspecionando Contêineres: Entendendo o Que Acontece Por Trás das Cenas

Contêineres são caixas pretas por design, mas isso não significa que não podemos saber o que está acontecendo dentro delas ou como estão configuradas. Quando um contêiner não se comporta como esperado, ou quando precisamos de informações detalhadas sobre seu estado, rede, volumes ou processos, as ferramentas de inspeção do Docker se tornam indispensáveis. Elas nos permitem "olhar por dentro" sem a necessidade de entrar no contêiner.

Mecânico de Sistemas

Assim como um mecânico usa um scanner de diagnóstico antes de mexer no carro, você deve inspecionar contêineres antes de depurar

Auditoria e Segurança

A inspeção é vital para verificar se as configurações e políticas de segurança estão sendo aplicadas corretamente

DevSecOps

Em cenários de DevSecOps, a inspeção ajuda a verificar políticas de segurança de rede e acesso a recursos

Imagine que você é um mecânico e um carro chega à sua oficina. Você não vai simplesmente começar a mexer sem antes verificar o painel, o manual e talvez conectar um scanner de diagnóstico. Da mesma forma, antes de tentar depurar um problema em um contêiner, é crucial coletar informações sobre ele. O Docker oferece comandos que agem como esse "scanner de diagnóstico", fornecendo uma riqueza de detalhes sobre cada contêiner em execução ou parado.

Listando Contêineres: docker ps

O comando docker ps é o seu ponto de partida para ver quais contêineres estão ativos no seu sistema.

- **docker ps**

  Lista apenas os contêineres em execução

- **docker ps -a (ou --all)**

  Lista todos os contêineres, incluindo os parados

- **docker ps -s (ou --size)**

  Exibe o tamanho do contêiner

- **docker ps -f "status=exited"**

  Filtra contêineres por status (ex: exited, running, paused)

```
# Listar todos os contêineres (ativos e inativos)
docker ps -a
```

A saída de docker ps mostra informações como ID do contêiner, imagem usada, comando executado, tempo de criação, status, portas mapeadas e o nome do contêiner. É um resumo rápido e útil para ter uma visão geral do seu ambiente.

Inspecionando Detalhes: docker inspect

Enquanto docker ps oferece uma visão panorâmica, docker inspect mergulha fundo, revelando cada detalhe da configuração e do estado de um contêiner. É como ler o manual técnico completo de um equipamento, com todas as especificações e configurações.

docker inspect [ID_ou_Nome_do_Contêiner]

Este comando retorna um objeto JSON com uma vasta quantidade de informações sobre o contêiner, incluindo:

ID e Nome Identificadores únicos do contêiner	**Estado** Se está rodando, parado, pausado, etc., com detalhes como PID e tempos
Configuração Imagem usada, variáveis de ambiente, comandos de entrada	**Rede** Endereços IP, configurações de rede, portas mapeadas
Volumes Quais volumes estão montados e onde	**Recursos** Limites de CPU e memória

```
# Inspecionar o contêiner Nginx que criamos
docker inspect meu-servidor-web-publico
```

A saída JSON pode ser bastante longa, mas é extremamente útil para scripts de automação, depuração avançada ou para entender exatamente como um contêiner está configurado. Você pode usar ferramentas como jq para filtrar e formatar a saída JSON, facilitando a leitura e a extração de informações específicas.

> **Extração de Dados Específicos**
>
> O -f (ou --format) permite que você use templates Go para extrair informações específicas, tornando docker inspect uma ferramenta poderosa para automação e scripts.

```
# Exemplo: Obter o endereço IP do contêiner
docker inspect -f '{{.NetworkSettings.IPAddress}}' meu-servidor-web-publico
```

A capacidade de extrair dados programaticamente é fundamental em pipelines de CI/CD e para a implementação de **GitOps**, onde a infraestrutura é definida como código e as verificações automatizadas dependem dessas informações.

Visualizando Logs: A Voz do Seu Contêiner

Contêineres são ambientes isolados, e as aplicações que rodam dentro deles geram informações constantemente. Essas informações, os logs, são a "voz" do seu contêiner, narrando o que está acontecendo internamente: erros, avisos, informações de depuração, requisições recebidas, e muito mais. Acessar e analisar esses logs é fundamental para entender o comportamento da sua aplicação, diagnosticar problemas e monitorar sua saúde.

Sinais Vitais

Os logs são como os sinais vitais de um paciente: fornecem um histórico completo e indicam tendências ou problemas

Operação Informada

Sem acesso aos logs, você estaria operando às cegas, incapaz de reagir a falhas ou otimizar o desempenho

Imagine que você está monitorando um paciente em um hospital. Os logs são como os sinais vitais e os registros médicos: eles fornecem um histórico completo do que aconteceu, o que está acontecendo agora e podem indicar tendências ou problemas futuros. Sem acesso a essas informações, você estaria operando às cegas, incapaz de reagir a falhas ou otimizar o desempenho.

Conexão com AIOps

Em um ambiente de produção moderno, especialmente com a adoção de **AIOps**, a coleta e análise de logs de contêineres são automatizadas e alimentam sistemas de monitoramento e detecção de anomalias.

Visualizando Logs de Contêineres: docker logs

O comando docker logs recupera os logs de um contêiner. Por padrão, ele mostra a saída padrão (stdout) e o erro padrão (stderr) do processo principal do contêiner.

docker logs [ID_ou_Nome]

Exibe todos os logs do contêiner desde o início

docker logs -f [ID_ou_Nome]

Segue a saída de log em tempo real, como o comando tail -f

docker logs --tail [número] [ID_ou_Nome]

Exibe apenas as últimas [número] linhas do log

docker logs --since [timestamp] [ID_ou_Nome]

Exibe logs a partir de um determinado momento

```
# Exemplo: Ver os logs do nosso servidor Nginx
docker logs meu-servidor-web-publico

# Exemplo: Acompanhar os logs em tempo real
docker logs -f meu-servidor-web-publico

# Exemplo: Ver as últimas 10 linhas de log
docker logs --tail 10 meu-servidor-web-publico
```

Logs Avançados e a Importância no Cenário Atual

A capacidade de visualizar logs em tempo real (-f) é inestimável durante o desenvolvimento e a depuração. Você pode iniciar sua aplicação em um contêiner, abrir outro terminal para seguir os logs e ver imediatamente o impacto de suas interações ou as mensagens de erro geradas. Isso acelera drasticamente o ciclo de feedback e a resolução de problemas.

Além da depuração manual, os logs são a espinha dorsal de qualquer estratégia de monitoramento robusta. Em um cenário de **AIOps**, por exemplo, esses logs são coletados por agentes, centralizados em plataformas como ELK Stack (Elasticsearch, Logstash, Kibana) ou Splunk, e então analisados por algoritmos de Machine Learning. Esses algoritmos podem detectar padrões incomuns, prever falhas ou identificar a causa raiz de problemas de forma autônoma, muito antes que um humano perceba.

Boas Práticas para Logs em Contêineres

Log para stdout/stderr

As aplicações dentro do contêiner devem enviar seus logs para a saída padrão e erro padrão, pois o Docker os captura automaticamente. Evite escrever logs diretamente em arquivos dentro do contêiner, pois eles se perdem quando o contêiner é removido.

Formato estruturado

Se possível, configure suas aplicações para gerar logs em formatos estruturados como JSON. Isso facilita a análise e o parsing por ferramentas de monitoramento.

Níveis de log

Utilize níveis de log (DEBUG, INFO, WARN, ERROR, FATAL) para controlar a verbosidade e filtrar informações importantes.

Centralização de logs

Em ambientes de produção, sempre use uma solução de centralização de logs. O docker logs é ótimo para depuração local, mas não escala para múltiplos contêineres e serviços.

Observabilidade

A gestão eficaz de logs é um componente crítico da observabilidade de sistemas distribuídos. Dominar docker logs é o primeiro passo para garantir que você sempre tenha visibilidade sobre o que suas aplicações contêinerizadas estão fazendo, um requisito fundamental para a resiliência e a performance em qualquer ambiente moderno.

Colocando em Prática: Um Fluxo Completo de Gerenciamento

Até agora, exploramos os comandos de gerenciamento de contêineres de forma individual. No entanto, o verdadeiro poder do Docker reside na capacidade de orquestrar esses comandos em um fluxo de trabalho coeso. Para solidificar seu aprendizado, vamos simular um cenário comum: implantar uma aplicação web simples, gerenciá-la e, em seguida, desmontar o ambiente.

Imagine que você está desenvolvendo uma nova funcionalidade para um site. Você precisa de um ambiente de teste rápido e isolado para rodar sua aplicação e um banco de dados. O Docker permite que você crie esse ambiente em segundos, sem poluir sua máquina local com instalações e dependências. Este é o cenário ideal para aplicar tudo o que aprendemos.

Cenário: Aplicação Web com Nginx

Vamos criar um contêiner Nginx que servirá uma página HTML estática.

Iniciar um contêiner Nginx

Em modo destacado e mapear a porta

```
docker run -d -p 8080:80 --name meu-site-docker nginx
```

- **-d:** Roda em segundo plano
- **-p 8080:80:** Mapeia a porta 8080 do host para a porta 80 do contêiner
- **--name meu-site-docker:** Dá um nome fácil de lembrar ao contêiner
- **nginx:** A imagem a ser usada

Verificar se o contêiner está rodando

```
docker ps
```

Você deve ver **meu-site-docker** na lista, com status Up.

Acessar a aplicação

Abra seu navegador e digite **http://localhost:8080**. Você deve ver a página de boas-vindas padrão do Nginx.

Inspecionar o contêiner

Para obter detalhes de rede

```
docker inspect meu-site-docker | grep "IPAddress"
```

Isso mostrará o endereço IP interno do contêiner.

Gerenciamento e Limpeza do Ambiente

Continuando nosso fluxo de trabalho, agora que a aplicação está rodando e você sabe como inspecioná-la, vamos praticar a visualização de logs e, finalmente, a remoção do ambiente.

01

Visualizar os logs do contêiner

```
docker logs meu-site-docker
```

Você verá os logs de acesso do Nginx. Se você atualizar a página no navegador, novos logs aparecerão.

02

Acompanhar os logs em tempo real

(em outro terminal)

```
docker logs -f meu-site-docker
```

Agora, cada vez que você atualizar a página no navegador, verá as novas entradas de log instantaneamente.

03

Parar o contêiner

```
docker stop meu-site-docker
```

Verifique com **docker ps** que ele não está mais na lista de contêineres em execução. Com **docker ps -a**, ele aparecerá como Exited.

04

Remover o contêiner

```
docker rm meu-site-docker
```

Agora, o contêiner foi completamente removido do seu sistema.

Agilidade e Eficiência

Este fluxo de trabalho demonstra a agilidade e a eficiência do Docker. Você pode rapidamente subir um ambiente, testar sua aplicação, monitorar seu comportamento e, em seguida, desmontá-lo sem deixar rastros em sua máquina. Essa capacidade de criar e destruir ambientes efêmeros é um dos pilares da cultura **DevOps** e facilita a implementação de práticas como **GitOps**, onde a infraestrutura é tratada como código e pode ser provisionada e desprovisionada sob demanda.

Boas Práticas e Conexões com Tendências Atuais

Dominar os comandos básicos de gerenciamento de contêineres é o alicerce, mas a verdadeira maestria vem com a aplicação de boas práticas e a compreensão de como esses conceitos se encaixam nas tendências mais amplas do DevOps. A gestão de contêineres não é um fim em si mesma, mas uma parte integrante de um ecossistema maior que busca automação, segurança e resiliência.

Pense em um piloto de avião. Ele não apenas sabe como ligar o motor, mas entende todo o sistema de navegação, as condições climáticas e os protocolos de segurança. Da mesma forma, um especialista em Docker não apenas executa comandos, mas compreende o contexto de GitOps, AIOps e DevSecOps, aplicando os princípios de gerenciamento de contêineres para otimizar todo o ciclo de vida do software.

Boas Práticas no Gerenciamento de Contêineres

1

Nomeie seus contêineres

Use a opção --name para dar nomes significativos aos seus contêineres. Isso facilita a identificação e o gerenciamento, especialmente quando você tem muitos contêineres.

2

Limpeza regular

Contêineres parados e imagens não utilizadas consomem espaço em disco. Use docker system prune (com cautela!) para limpar recursos não utilizados.

3

Use docker-compose

Para aplicações com múltiplos contêineres (ex: web app + banco de dados), docker-compose simplifica o gerenciamento, permitindo definir e orquestrar todos os serviços em um único arquivo.

4

Monitore seus contêineres

Além de docker logs, use ferramentas de monitoramento para acompanhar o uso de recursos (CPU, memória) e a saúde dos seus contêineres.

Conectando com as Tendências

GitOps

O gerenciamento de contêineres é fundamental para GitOps. A definição de como os contêineres são executados (suas imagens, portas, variáveis de ambiente) é versionada no Git. Ferramentas de GitOps, como Argo CD ou Flux, monitoram o repositório Git e garantem que o estado real dos contêineres no cluster corresponda ao estado desejado definido no Git.

AIOps

AIOps se baseia fortemente na coleta e análise de logs e métricas de contêineres. Os logs que você aprendeu a visualizar com docker logs são a matéria-prima para algoritmos de IA que detectam anomalias, preveem falhas e automatizam a resposta a incidentes.

DevSecOps (Shift-Left)

Gerenciar contêineres de forma segura é um pilar do DevSecOps. Isso inclui usar imagens base seguras, escanear imagens em busca de vulnerabilidades antes de executá-las (docker scan), e garantir que os contêineres rodem com os privilégios mínimos necessários. A inspeção de contêineres (docker inspect) pode ajudar a verificar configurações de segurança.

Ao adotar essas práticas e entender o contexto mais amplo, você não está apenas gerenciando contêineres, mas construindo sistemas mais robustos, seguros e automatizados.

Consolidação e Próximos Passos

Nesta aula, mergulhamos no universo do gerenciamento de contêineres Docker, desvendando o ciclo de vida completo, desde a sua criação até a sua remoção. Exploramos os comandos essenciais como docker run, docker stop, docker start e docker rm, que são a base para qualquer interação com contêineres. Aprendemos a diferenciar os modos de execução interativo e destacado, e a importância vital do mapeamento de portas para expor serviços. Por fim, dominamos as ferramentas de inspeção (docker ps, docker inspect) e visualização de logs (docker logs), que são seus olhos e ouvidos dentro do ambiente contêinerizado.

4	2	100%
Comandos Essenciais	**Modos de Execução**	**Controle Total**
run, stop, start, rm - a base do gerenciamento	Interativo (-it) e Destacado (-d)	Sobre o ciclo de vida dos contêineres

Você agora possui as habilidades fundamentais para orquestrar contêineres Docker, um conhecimento indispensável no cenário atual de desenvolvimento e operações. A capacidade de gerenciar esses componentes isolados e efêmeros é o que impulsiona a agilidade e a escalabilidade das aplicações modernas, sendo um diferencial competitivo para qualquer profissional de tecnologia.

Em Prática

Para solidificar seu aprendizado, experimente criar diferentes tipos de contêineres (um servidor web, um banco de dados simples, uma ferramenta de linha de comando), pratique iniciar e parar, mapear portas variadas e inspecionar seus detalhes. Simule cenários de erro e use os logs para depurar. A prática constante é a chave para a maestria em Docker.

Autoavaliação

1. Qual comando Docker é utilizado para criar e iniciar um novo contêiner a partir de uma imagem?
   a) docker create
   b) docker start
   c) docker run
   d) docker init
2. Para executar um contêiner em segundo plano, liberando o terminal, qual opção deve ser utilizada com o comando docker run?
   a) -i
   b) -t
   c) -d
   d) -p
3. Você tem um servidor web rodando dentro de um contêiner na porta 80. Para acessá-lo do seu navegador na porta 8000 da máquina hospedeira, qual seria a sintaxe correta para o mapeamento de portas?
   a) -p 80:8000
   b) -p 8000:80
   c) -p 8000
   d) -p 80
4. Qual comando é mais adequado para visualizar as últimas 20 linhas de log de um contêiner chamado minha-app em tempo real?
   a) docker logs minha-app --tail 20
   b) docker logs -f --tail 20 minha-app
   c) docker logs minha-app -f
   d) docker logs minha-app --last 20
5. Explique a importância do comando docker inspect para a depuração e monitoramento de contêineres, e cite dois tipos de informações cruciais que ele pode fornecer.

Gabarito

1. c) | 2. c) | 3. b) | 4. b)

Próxima Aula

Na **Aula 21 – Volumes e Redes em Docker**, aprofundaremos ainda mais no ecossistema Docker, explorando como gerenciar dados persistentes com volumes e como configurar redes complexas para permitir a comunicação entre contêineres e com o mundo exterior de forma mais sofisticada.

Recursos Adicionais

- **Documentação Oficial do Docker:** Fonte primária e mais completa para todos os comandos e conceitos.
- **Docker Labs:** Tutoriais práticos e interativos para aprofundar o conhecimento.
- **Artigos sobre GitOps, AIOps e DevSecOps:** Para entender o contexto mais amplo da gestão de contêineres em pipelines modernos.

NOTA IMPORTANTE: As informações técnicas desta aula estão atualizadas até 2025. Consulte sempre fontes oficiais para verificar alterações e as melhores práticas de segurança.