CONEX NETWORKS

Conhecendo Docker e Docker Compose: Visão Geral e Principais Recursos

By Aluizio Developer

O que é Docker?

Docker é uma plataforma de software que permite criar, testar e implantar aplicativos rapidamente. Ele utiliza containers, que são pacotes leves, portáteis e autossuficientes, que incluem tudo o que é necessário para executar um pedaço de software, incluindo código, runtime, bibliotecas e configurações.

Aqui estão alguns pontos-chave sobre o Docker:

Containers: Ao contrário das máquinas virtuais, os containers compartilham o kernel do sistema operacional, mas são isolados uns dos outros e do sistema host, o que os torna mais leves e rápidos.

Portabilidade: Como os containers incluem todas as dependências necessárias para o aplicativo, eles podem ser executados em qualquer ambiente que tenha o Docker instalado, seja no seu laptop, em servidores de desenvolvimento, ou em ambientes de produção na nuvem.

Imagens Docker: Uma imagem Docker é um modelo estático que define um contêiner Docker. As imagens podem ser criadas a partir de um Dockerfile, que é um script com uma série de instruções para construir a imagem.

Docker Hub: É um registro público de imagens Docker onde os desenvolvedores podem compartilhar e obter imagens prontas para uso.

Orquestração: Ferramentas como Kubernetes podem ser usadas junto com Docker para gerenciar, escalar e implantar containers em ambientes de produção.

Docker simplifica o processo de desenvolvimento, permitindo que os desenvolvedores se concentrem em escrever código sem se preocupar com o ambiente onde ele será executado.

Docker Engine

Docker Engine é a tecnologia subjacente que cria e executa containers. É um runtime de containers que inclui o Docker daemon, uma API REST e a interface de linha de comando (CLI). Componentes:

Docker Daemon: Gerencia os containers Docker em um host específico.

API REST: Permite a interação programática com o daemon.

CLI: Interface de linha de comando usada para interagir com o Docker.

O Docker Engine pode ser instalado diretamente em servidores Linux e em algumas versões de Windows e macOS (com limitações), sendo tipicamente usado em ambientes de servidor ou produção, onde um ambiente de desktop completo não é necessário.

Os processos de configuração e gerenciamento mais diretos geralmente são feitos por administradores de sistemas ou DevOps.

Docker Desktop

Docker Desktop é uma aplicação que fornece uma interface de usuário amigável e um conjunto de ferramentas adicionais para facilitar o uso do Docker em ambientes de desktop, como Windows e macOS. Inclui Docker Engine para criar e executar containers. Componentes:

Docker Compose: Ferramenta para definir e gerenciar aplicativos multi-contêiner.

Kubernetes: Orquestrador de containers que pode ser habilitado opcionalmente.

Dashboard: Interface gráfica para gerenciar containers, imagens e volumes.

Volume Management: Ferramentas para gerenciar volumes Docker.

Automatic Updates: Mantém o Docker atualizado automaticamente.

O Docker Desktop foi projetado para desenvolvedores que trabalham em ambientes de desktop e precisam de uma maneira fácil para criar, testar e gerenciar containers. Facilita a configuração e uso do Docker em um ambiente de desenvolvimento, oferecendo uma experiência mais integrada e simplificada.

Diferenças entre Docker Engine e Docker Desktop

Docker Engine é a tecnologia principal que executa containers, normalmente usada em servidores e ambientes de produção.

Docker Desktop é uma aplicação completa para desktops que inclui Docker Engine e outras ferramentas para facilitar o desenvolvimento e a gestão de containers em ambientes de desktop.

Docker Desktop é ideal para desenvolvedores que desejam uma solução fácil e integrada para trabalhar com containers, enquanto Docker Engine é mais

adequado para uso em ambientes de produção e servidores, onde a interface gráfica não é necessária.

CLI do Docker

Para criar um container de um servidor PostgreSQL utilizando a CLI do Docker a partir de uma imagem da Bitnami, você pode seguir os seguintes passos:

Pull da Imagem: baixar a imagem do PostgreSQL da Bitnami a partir do Docker Hub.

Criar e Iniciar o container: criar e iniciar um container com a imagem baixada, especificando as configurações necessárias, como as variáveis de ambiente para o usuário e a senha do PostgreSQL.

Baixar a imagem do PostgreSQL da Bitnami:

```
docker pull bitnami/postgresql:latest
```

Criar e iniciar o container:

```
docker run -d \
  --name meu_postgres \
  -e POSTGRESQL_USERNAME=usuario \
  -e POSTGRESQL_PASSWORD=senha \
  -e POSTGRESQL_DATABASE=meubanco \
  -p 5432:5432 \
  bitnami/postgresql:latest
```

Neste comando:

-d: executa o container em segundo plano (modo desanexado).

--name meu_postgres: dá um nome ao container.

-e POSTGRESQL_USERNAME=usuario: define o nome de usuário do PostgreSQL.

-e POSTGRESQL_PASSWORD=senha: define a senha do PostgreSQL.

-e POSTGRESQL_DATABASE=meubanco: define o nome do banco de dados a ser criado.

-p 5432:5432: mapeia a porta 5432 do container para a porta 5432 do host, permitindo acesso ao PostgreSQL do container a partir do host.

bitnami/postgresql:latest: especifica a imagem a ser usada.

Com esses passos, você terá um container PostgreSQL em execução utilizando a imagem da Bitnami.

Docker Compose

O Docker Compose é uma ferramenta que permite definir e gerenciar aplicativos Docker de vários containers. Ele usa um arquivo YAML para configurar os serviços de um aplicativo, o que facilita a definição, execução e orquestração de aplicativos complexos que requerem múltiplos containers. Aqui estão alguns pontos importantes sobre o Docker Compose:

Arquivo YAML (docker-compose.yml): esse arquivo define todos os serviços que compõem o aplicativo, incluindo containers, redes e volumes. Ele especifica como cada container deve ser construído, quais portas devem ser expostas, quais volumes devem ser montados e quais variáveis de ambiente devem ser definidas.

Facilidade de Uso: com um único comando (docker-compose up), você pode iniciar todos os serviços definidos no arquivo YAML. Da mesma forma, com o comando docker-compose down, você pode parar e remover todos os containers, redes e volumes associados ao aplicativo.

Serviços Multi-Container: docker compose facilita a execução de aplicativos que consistem em vários serviços interdependentes, como um banco de dados, um servidor web e um serviço de cache.

Escalabilidade: é possível escalar um serviço para um número específico de containers com o comando docker-compose up --scale, o que é útil para testar a escalabilidade e o balanceamento de carga.

Ambientes de Desenvolvimento: docker compose é muito útil para criar ambientes de desenvolvimento replicáveis e consistentes. Todos os membros da equipe de desenvolvimento podem usar o mesmo arquivo docker-compose.yml para garantir que estão executando o mesmo ambiente em suas máquinas locais.

Exemplo básico de um arquivo docker-compose.yml:

```
version: '3'
services:
  web:
    image: nginx
    ports:
      - "80:80"
  db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: example
```

Neste exemplo, o arquivo define dois serviços: um serviço web que usa a imagem do Nginx e expõe a porta 80, e um serviço db que usa a imagem do PostgreSQL e define a senha do banco de dados através de uma variável de ambiente.

Container PostgreSQL

Para criar um container de um servidor PostgreSQL usando o Docker Compose com a imagem da Bitnami, você precisa criar um arquivo docker-compose.yml com as configurações necessárias.

Crie um arquivo chamado docker-compose.yml com o seguinte conteúdo:

```
version: '3.8'

services:
  postgres:
    image: bitnami/postgresql:latest
    container_name: meu_postgres
    environment:
      - POSTGRESQL_USERNAME=usuario
      - POSTGRESQL_PASSWORD=senha
      - POSTGRESQL_DATABASE=meubanco
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/bitnami/postgresql

volumes:
  postgres_data:
    driver: local
```

Salve o arquivo docker-compose.yml e execute o comando a seguir no terminal para iniciar o container:

```
docker-compose up -d
```

Neste arquivo docker-compose.yml, temos:

version: '3.8': especifica a versão do Docker Compose.

services: define os serviços que serão executados.

postgres: é o nome do serviço.

image: especifica a imagem a ser usada.

container_name: dá um nome ao container.

environment: define as variáveis de ambiente para configurar o PostgreSQL.

ports: mapeia a porta 5432 do container para a porta 5432 do host.

volumes: monta o volume postgres_data no diretório /bitnami/postgresql dentro do container.

Com esse arquivo, você pode facilmente iniciar e gerenciar o container PostgreSQL usando o Docker Compose.