VOLUMES DOCKER

BACKUP

Palestrante: Brendo Freitas

Desenvolvedor Full Stack

O que é um Volume Docker ?

Em Docker, um volume é um mecanismo para persistir e compartilhar dados entre contêineres e o host do sistema. Eles são usados para armazenar dados que devem sobreviver à vida efêmera de um contêiner. Um volume Docker é uma pasta especializada dentro do sistema de arquivos do host ou uma localização no sistema de arquivos gerenciado pelo Docker. É um recurso essencial para lidar com dados persistentes em contêineres.

Para que Serve um Volume Docker?

1- Persistência de Dados:

Os contêineres são efêmeros, o que significa que eles podem ser facilmente parados e removidos. Volumes permitem que você persista dados importantes, como bancos de dados, arquivos de configuração e uploads de usuário, mesmo quando os contêineres são excluídos.

2- Compartilhamento de Dados

Volumes podem ser compartilhados entre vários contêineres. Isso é útil para compartilhar informações e estados entre diferentes partes de um aplicativo, como um aplicativo web e um banco de dados.

3- Backup e Restauração

Você pode criar backups de volumes para proteger seus dados e restaurá-los quando necessário. Isso é crucial para a recuperação de desastres e migração de aplicativos.

Para que Serve um Volume Docker?

4- Integração com Armazenamento Externo

Volumes podem ser usados para integrar-se com sistemas de armazenamento externo, como NFS ou sistemas de armazenamento em nuvem. Isso permite que você armazene seus dados em locais centralizados e escaláveis.

5- Isolamento de Dados

1. Volumes isolam os dados dos contêineres, garantindo que os dados persistentes não afetem a imagem do contêiner. Isso é importante para manter a integridade e portabilidade das imagens Docker.

6- Desenvolvimento Local

Volumes podem ser usados para montar o código-fonte e os arquivos de desenvolvimento localmente no contêiner. Isso facilita o desenvolvimento e teste de aplicativos dentro de um ambiente de contêiner.

Resumindo,

os volumes Docker são uma parte fundamental da orquestração de contêineres e são usados para garantir que os dados persistentes sejam gerenciados de maneira eficiente e confiável em ambientes de contêineres. Eles desempenham um papel crucial na persistência de dados e na portabilidade de aplicativos Docker.

Segurança nos volumes dos contêineres:

1- Evite Dados Sensíveis

:Evite armazenar dados altamente sensíveis em volumes Docker sempre que possível. Considere alternativas, como o uso de serviços de armazenamento gerenciados por provedores de nuvem.

2 -Backup e Recuperação de Dados

Implemente uma estratégia de backup e recuperação de dados para garantir que você possa restaurar volumes em caso de perda de dados ou corrupção.

3 - Use Volumes Gerenciados pelo Docker

Evite montar diretamente pastas do sistema host no contêiner, a menos que seja estritamente necessário. Em vez disso, use volumes gerenciados pelo Docker, que oferecem mais controle e segurança sobre os dados.

Segurança nos volumes dos contêineres:

4 - Use Volumes Criptografados

Se você estiver lidando com dados sensíveis, considere usar volumes criptografados para proteger os dados em repouso.

5- Controle de Acesso a Volumes

Use controle de acesso baseado em funções (RBAC) ou outras soluções de autenticação para controlar quem tem acesso de leitura e gravação aos volumes. Evite dar permissões de escrita a contêineres que não precisam delas.

6- Auditoria de Acesso

Implemente logs e auditorias para registrar atividades de acesso aos volumes, permitindo a detecção de comportamento suspeito ou não autorizado.

Fazendo Backup de um Volume Docker:

Antes de criar um backup, é necessário ter um volume Docker configurado para o seu contêiner.

Comando para criar um volume:
docker volume create meu_volume

Vamos precisar de um container temporário para criar o backup!

O próximo comando, cria um contêiner temporário, monta o volume e a pasta do sistema host:

Fazendo Backup de um Volume Docker:

Antes, precisamos criar uma pasta em nosso host:

```
mkdir backup
```

Depois executamos o comando:

```
docker run -it --name meu_container -v meu_volume:/volume_data -v $(pwd):/backup alpine:latest sh
```

Irei explicar cada passo do comando,

Pode ficar tranquilo!

Fazendo Backup de um Volume Docker:

docker run -it: *Executa um contêiner em modo interativo.*

--name meu_container: *Define o nome do contêiner como meu_container.*

-v meu_volume:/volume_data: *Monta o volume Docker meu_volume dentro do contêiner em /volume_data.*

-v $(pwd):/backup: *Monta a pasta atual do sistema host (onde você executou o comando) em /backup dentro do contêiner.*

alpine:latest: *Usa a imagem Alpine Linux como base para o contêiner temporário.*

sh: *Inicia o shell dentro do contêiner.*

Fazendo Backup de um Volume Docker:

Dentro do contêiner temporário, vamos usar o comando tar para criar o backup:

tar -czvf /backup/meu_backup.tar.gz -C /volume_data .

tar -czvf: *Comando para criar um arquivo tar e compactá-lo.*

/backup/meu_backup.tar.gz: *O caminho onde o arquivo de backup será salvo dentro do contêiner.*

-C /volume_data: *Mude o diretório para /volume_data, pois é onde os dados do volume estão.*

. (ponto): *Inclui todos os arquivos e pastas no diretório atual (que é /volume_data).*

Fazendo Backup de um Volume Docker:

Copiando o Backup para Fora do Contêiner:

Depois de criar o backup, nós vamos copiar para fora do nosso sistema host:

docker cp meu_container:/backup/meu_backup.tar.gz ./meu_backup.tar.gz

docker cp: *Comando para copiar arquivos entre o contêiner e o sistema host.*

./meu_backup.tar.gz: *O destino no sistema host onde o arquivo de backup será copiado.*

Depois, removemos o contêiner temporário:

docker rm meu_container

Restaurando um backup no Volume

Criar um contêiner temporário:

```
docker run -it --name meu_container -v meu_volume:/volume_data -v $(pwd):/backup alpine:latest sh
```

Descompactar o arquivo:

```
tar -xzvf /backup/meu_backup.tar.gz -C /volume_data
```

Depois, iremos remover o contêiner temporário:

```
docker rm meu_container
```

Usando Plugins

O que são Docker Plugins?

Docker plugins são extensões que adicionam funcionalidades personalizadas ao Docker, permitindo que você estenda e personalize a funcionalidade do Docker Engine. Eles permitem que você integre soluções de terceiros, ferramentas de armazenamento, redes personalizadas, drivers de volume e muito mais diretamente ao Docker, estendendo suas capacidades.

Principais Componentes de um Docker Plugin:

1- Plugin Framework:

O Docker Engine fornece um framework para a criação, gerenciamento e execução de plugins. Isso inclui a API Plugin, que permite a comunicação entre o Docker e os plugins.

2- Plugin API:

Os plugins Docker seguem uma API específica que define como o Docker interage com eles. Isso inclui endpoints de API e um conjunto de comandos para gerenciar os plugins.

3- Plugin Manager:

O Docker Engine possui um componente chamado Plugin Manager, que é responsável por gerenciar os plugins, incluindo a instalação, ativação, desativação e remoção de plugins.

4- Plugins de Tipo:

Existem vários tipos de plugins Docker, incluindo plugins de rede, plugins de volume, plugins de log, etc. Cada tipo de plugin estende uma área específica de funcionalidade do Docker.

Principais Tipos de Docker Plugins:

1- Network Plugins:

Esses plugins estendem a funcionalidade de rede do Docker, permitindo que você crie redes personalizadas para contêineres, integre com soluções de SDN (Software Defined Networking) e muito mais. Exemplos incluem o plugin "bridge" padrão e plugins de rede de terceiros, como o Calico.

2- Volume Plugins

Os plugins de volume permitem que você integre soluções de armazenamento externo ao Docker. Eles permitem que você use sistemas de arquivos distribuídos, armazenamento em nuvem e outras soluções de armazenamento como volumes para contêineres.

3- Logging Plugins:

Esses plugins permitem que você estenda a funcionalidade de registro do Docker. Eles podem direcionar logs para locais específicos, fazer análises ou integrar com soluções de monitoramento.

Benefícios dos Docker Plugins:

1- Extensibilidade:

Os plugins permitem estender o Docker para atender às necessidades específicas de sua infraestrutura e aplicativos.

2- Reutilização:

Você pode reutilizar e compartilhar plugins com a comunidade, economizando tempo na criação de soluções personalizadas.

3-Integração:

Os plugins permitem que você integre facilmente soluções de terceiros ao Docker, aproveitando as inovações externas.

Automatizando:

Code Github:
https://github.com/brendofreitas/BackupContainers.git

*Código de automatização de backup de **volumes docker**.*

docker

OBRIGADO!

linkedin