Docker Volumes

npNOG 10

November 25 - 28, 2024

Docker Volumes

Docker volumes are a crucial part of the Docker ecosystem, providing a way to persist data generated by and used by Docker containers. Volumes are managed by Docker and can be used to share data between the host system and the containers, as well as between multiple containers.

Types of Docker Storage

- **Volumes**:
  - Volumes are stored in a part of the host filesystem which is managed by Docker (`/var/lib/docker/volumes/` on Linux).
  - Volumes are the preferred mechanism for persisting data in Docker.
- **Bind Mounts**:
  - Bind mounts can be used to mount a file or directory from the host filesystem into a container.
  - The file or directory is referenced by its full or relative path on the host machine.
  - This method provides more control over the exact mount point but is less portable than volumes.
- **tmpfs Mounts**:
  - A tmpfs mount is a temporary filesystem mount that is stored in memory and not persisted on disk.
  - This is useful for cases where you need fast, ephemeral storage.

Creating and Using Docker Volumes

- **Create a Volume**
  To create a volume, you use the `docker volume create` command:

```
docker volume create my_volume
```

- **Inspect a Volume**
  You can inspect a volume to see details about it:

```
docker volume inspect my_volume
```

- **Using a Volume in a Container**
  You can use a volume in a container by specifying the `-v` or `--mount` flag when you run the container:
  Using `-v` (short syntax):

```
docker run -d -v my_volume:/path/in/container my_image
```

  Using `--mount` (long syntax):

```
docker run -d --mount source=my_volume,target=/path/in/container my_image
```

Managing Docker Volumes

- **List Volumes**
  To list all the volumes on your system:

```
docker volume ls
```

- **Remove a Volume**
  To remove a volume that is not being used by any container:

```
docker volume rm my_volume
```

- **Remove All Unused Volumes**
  To remove all volumes that are not used by at least one container:

```
docker volume prune
```

Examples and Use Cases

- **Persistent Storage**:
  Use volumes to persist data that should remain even if the container is removed. For example, storing database data.

```
docker run -d --name my_db -v my_db_volume:/var/lib/mysql mysql
```

- **Sharing Data Between Containers**:
  Use volumes to share data between multiple containers. For example, sharing configuration files.

```
docker run -d --name container1 -v shared_volume:/config busybox
docker run -d --name container2 -v shared_volume:/config busybox
```

Examples and Use Cases (contd.)

- **Backup and Restore**:
  Use volumes to easily backup and restore container data.
  Backup:

```
docker run --rm -v my_volume:/data -v $(pwd):/backup busybox tar cvf /backup/backup.tar /data
```

  Restore:

```
docker run --rm -v my_volume:/data -v $(pwd):/backup busybox tar xvf /backup/backup.tar -C /
```

Best Practices

- **Use Volumes for Persistent Data**:
  Volumes are designed to persist data, making them the best choice for stateful applications like databases.
- **Use Named Volumes**:
  Named volumes are easier to manage and understand than anonymous volumes.
- **Avoid Using Host Paths**:
  Using host paths can lead to portability issues. Use volumes instead.
- **Clean Up Unused Volumes**:
  Regularly prune unused volumes to free up space.