Docker Anonymous Volumes

To manage my stacks, containers, images, networks, and volumes, I use Portainer Community Edition.

I was surprised when I found three volumes with randomly generated names mixed in with my carefully organized ones. Initially, I thought it was a bug, but I soon realized they were **Anonymous Volumes**.

What Are Anonymous Volumes?

A **Docker Anonymous Volume** is a type of storage volume that Docker automatically creates when a container starts.

These volumes are not explicitly named by the user, which is why they're called *anonymous*. Docker uses them to persist data generated or required by the container, especially when a volume mount point is specified in the container's `Dockerfile` or `docker run` command without a named volume being provided.

Key Characteristics

1. **Automatic Creation**: Docker automatically creates anonymous volumes when a container specifies a mount point for data storage (via the `VOLUME` directive in the `Dockerfile` or the `-v` option in `docker run`) without assigning a specific name to the volume.
2. **Naming Convention**: These volumes are given unique, random names by Docker, making them harder to identify and manage.
3. **Purpose**: Anonymous volumes are typically used to persist data that a container needs to retain between runs, such as database files or logs, without requiring user management.
4. **Temporary Nature**: If a container is removed, the anonymous volume remains on the system unless manually deleted. Over time, these volumes can accumulate and consume disk space if not properly managed.

Organizing Anonymous Volumes

While anonymous volumes can be useful in certain situations, I find them a bit untidy for production environments, so I decided to trace their origins.

Example Scenario

Let’s consider the following scenario:

- You run a Redis container using the `redis:latest` image.

If you run the container without specifying a volume name:

```
docker run redis:latest
```

Docker will create an anonymous volume with a random name.

To uncover more details about the volume configuration, you can use the `docker image inspect` command to query the image’s metadata. For example, with the `redis:latest` image, the command would be:

```
docker image inspect redis:latest --format {{.Config.Volumes}}
```

This command will show the path inside the container where the volume is mounted:

```
map[/data:{}]
```

In this case, the path is `/data`.

This occurs because the Redis `Dockerfile` includes a `VOLUME /data` directive, instructing Docker to store Redis data in `/data` inside the container.

To keep things organized, I added a named volume to my `compose.yaml` file:

```
volumes:
    - volume_name:/data
```

With this adjustment, the problem is solved, and everything is neatly organized once again.

Happy me! □□

Revision #2
Created 17 August 2024 19:33:17 by Tiffanie BOREUX
Updated 16 October 2024 01:56:05 by Tiffanie BOREUX