AI-03973 Docker Volumes vs Named Volumes

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	June 24, 2026 12:37 PM

In Portainer, there are volumes which look like this - 08dd1b92e6d568ed8e01245e7931120fbb96, how can change my compose file so that all the volumes are named accordingly and easily identifiable?

You can absolutely make your Docker volumes human-readable in Portainer — the key is to **explicitly name your volumes in your** `docker-compose.yml` instead of letting Docker auto-generate those long hash-style names.

The short answer:

Define **named volumes** under the `volumes:` section at the bottom of your compose file, then reference them in your services.

🧩 Why you're seeing hash-like volume names

When a compose file uses **anonymous volumes**, Docker creates volumes with names like:

```
08dd1b92e6d568ed8e01245e7931120fbb96
```

These are auto-generated because Docker had no name to assign.

✅ How to fix it: explicitly name your volumes

Here's the clean, predictable pattern you want:

Example: Before (anonymous volumes)

```yaml
services:
  db:
    image: postgres
    volumes:
      - /var/lib/postgresql/data
```

This creates a random volume name.

After (named volumes)

```yaml
services:
  db:
    image: postgres
    volumes:
      - db_data:/var/lib/postgresql/data

volumes:
  db_data:
```

Now Portainer will show a volume named **db_data**, not a hash.

🧱 A more complete example with multiple services

```yaml
services:
  db:
    image: postgres
    volumes:
      - db_data:/var/lib/postgresql/data

  app:
    image: myapp
    volumes:
      - app_logs:/var/log/myapp
```

```
volumes:
  db_data:
  app_logs:
```

Every volume is now clearly labeled and easy to identify in Portainer.

🖼️ Visual reference: Docker volumes in Portainer

🧠 Extra tip: prefix volumes for large stacks

If you run multiple stacks, prefixing helps avoid collisions:

```
volumes:
  myproject_db:
  myproject_redis:
  myproject_uploads:
```

Portainer will show:

- myproject_db
- myproject_redis
- myproject_uploads

Much easier to manage.

Want me to help rewrite your actual compose file?

If you paste your current `docker-compose.yml`, I can rewrite it with clean, readable volume names.

So I make the change, will the old volume be orphaned?

Short version: **Yes — the old volume becomes orphaned, but it is NOT deleted.**

Docker will simply stop using it once your compose file points to a new, named volume.

Here's the breakdown so you know exactly what to expect.

🧭 What happens when you rename volumes in Compose

When you change this:

```
- /var/lib/postgresql/data
```

to this:

```
- db_data:/var/lib/postgresql/data
```

Docker Compose treats it as a **different volume**. The old anonymous one remains on disk, unused.

In Portainer, you'll see:

- **db_data** → the new named volume
- **08dd1b92e6d568ed8e01245e7931120fbb96** → the old orphaned volume

---

🧹 Should you delete the old volume?

Only after you confirm your service is running correctly with the new named volume.

If you delete the old one too early, you could lose data you meant to keep.

🛠️ How to safely migrate data (if needed)

If the old anonymous volume contains important data, you can copy it into the new named volume before deleting anything.

You can do this with a temporary container:

```
docker run --rm \
  -v 08dd1b92e6d568ed8e01245e7931120fbb96:/from \
```

```
  -v db_data:/to \
  alpine sh -c "cp -a /from/. /to/"
```

Now the new named volume contains everything the old one did.

🧼 Cleaning up orphaned volumes

Once everything works:

```
docker volume rm 08dd1b92e6d568ed8e01245e7931120fbb96
```

Or remove all unused volumes:

```
docker volume prune
```

If you want, I can check your compose file

If you paste your `docker-compose.yml`, I'll rewrite it with clean, readable volume names and tell you exactly which old volumes will become orphaned.