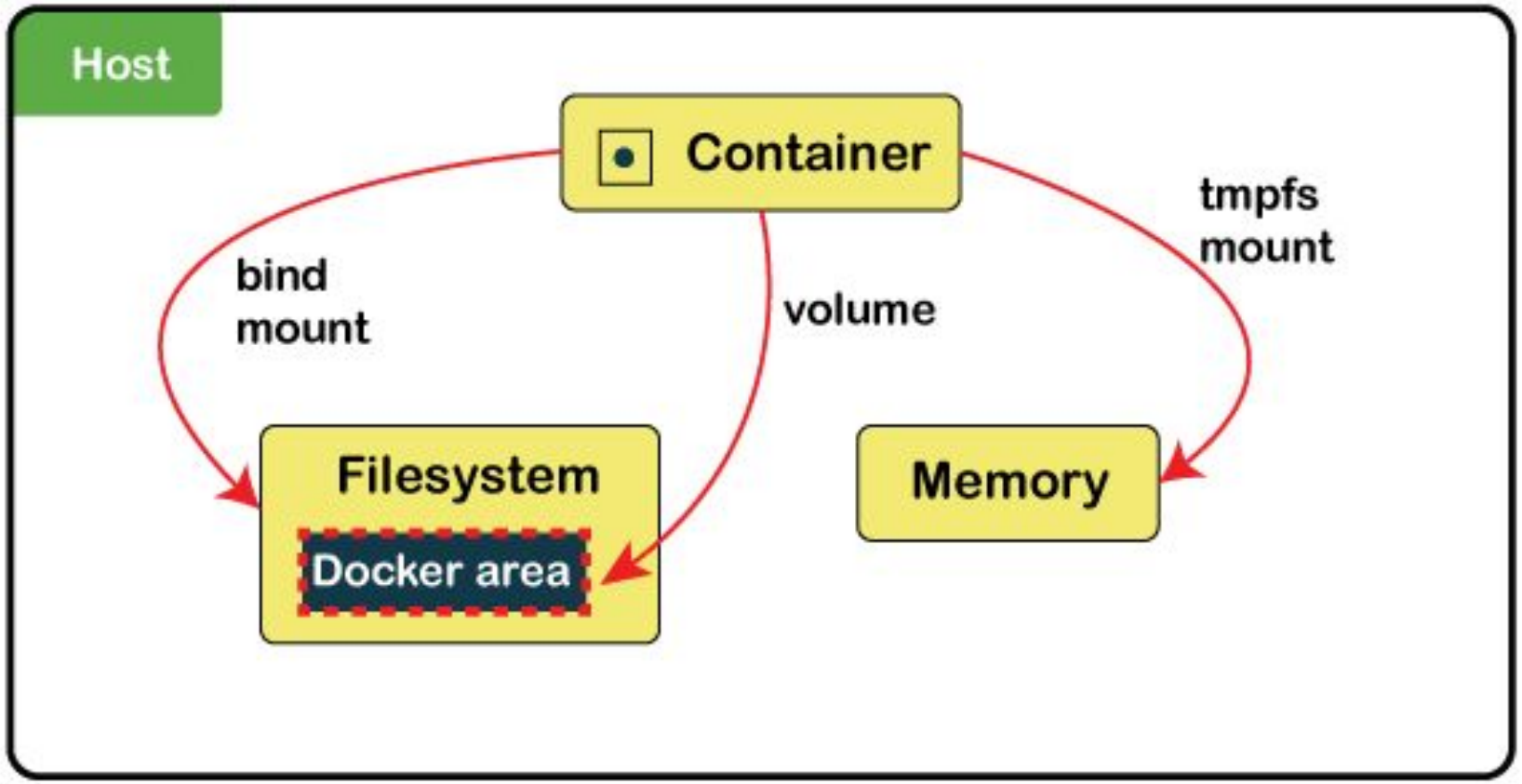
DevOps

Docker Volume

What is Docker Volume?



What is Docker Volume?

- Docker volumes are a widely used and useful tool for **ensuring data persistence** while working in containers. Docker volumes are file systems mounted on Docker containers to preserve data generated by the running container.
- The **data doesn't persist** when that container no longer exists, and it can be difficult to get the data out of the container if another process needs it.
- Docker has two options for containers to store files in the host machine so that the files are persisted even after the container stops:

    1. Volumes:Volumes are stored in a part of the host filesystem, managed by Docker (/var/lib/docker/volumes/ on Linux).

    2. Bind mounts: Bind mounts may be stored anywhere on the host system.

Volume

- $ docker volume create my-vol
- $ docker volume ls
- $ docker run -d -p 5432:5432 -e POSTGRES_PASSWORD=123 -v my-vol:/var/lib/postgresql/data postgres

If volume not exist, it will be created automatically.

- $ docker run -d -p 5433:5432 -e POSTGRES_PASSWORD=123 --mount source=my-vol,target=/app postgres

target = destination; /app will be created if not exist

- $ docker volume rm volume-name # Remove volume by name first
- $ docker volume prune # Remove all unused volume

Create Different types of volumes (docker volume)

- Create docker volume

```
docker run -dp 5432:5432 -e POSTGRES_PASSWORD=password \
--name postgres-cont \
--mount type=volume,source=my-new-vol,target=/var/lib/postgresql/data \
postgres
```

- it will create volume for you automatically to store your data

Create Different types of volumes (tmpfs)

- Using tmpfs

```
21  docker run -d \
22    --name pg-tmpfs \
23    -e POSTGRES_PASSWORD=password \
24    --tmpfs /var/lib/postgresql/data \
25    postgres
```

Create Different types of volumes (bind Mount)

- Create bind mount volume

```
docker run -dp 5432:5432 -e POSTGRES_PASSWORD=password \
--name postgres-cont \
--mount type=bind,source="$(pwd)",target=/var/lib/postgresql/data \
postgres
```

```
docker run -dp 5432:5432 -e POSTGRES_PASSWORD=password \
--name postgres-cont \
--mount
type=bind,source="/Users/myuser/Documents/basic/database/myfolder",target=/var
/lib/postgresql/data \
postgres
```

Volume

- `$ sudo vim Dockerfile`

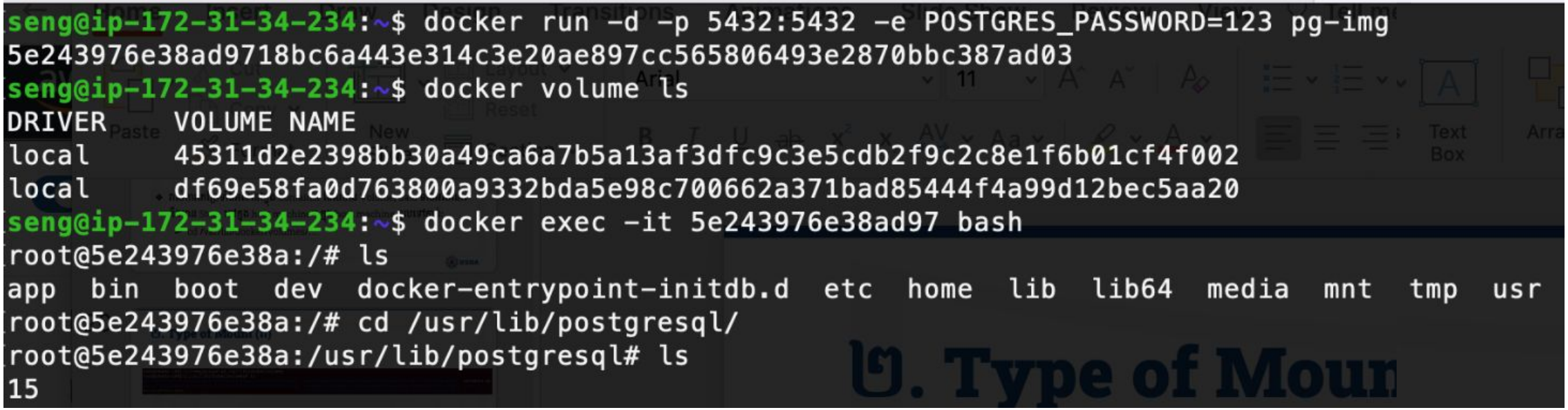
```
FROM postgres:13.8

VOLUME /var/lib/postgresql/data
VOLUME /app

ENV POSTGRES_PASSWORD=123
ENV POSTGRES_DB=test
ENV POSTGRES_USER=dara
```

- 1st VOLUME will be created automatically and mount to that path “/var/lib/postgresql/data”
- 2nd VOLUME will be created automatically and directory “/app” will be created if not exist and mount to it.

Volume




```
$ docker run -d -p 2348:5432 --name my-postgres8 -e POSTGRES_PASSWORD=123
-v /app postgres:14 # add and mount /app to the generated volume

$ docker run -d -p 2349:5432 --name my-postgres9 -e POSTGRES_PASSWORD=123
--volumes-from=my-postgres8 postgres:14 # mount volume from the existing
container
```

Bind mounts

- $ docker run -d -p 5435:5432 -e POSTGRES_PASSWORD=123 -v /home/seng/pg-db:/var/lib/postgresql/data postgres

If directory “pg-db” not exist, it will be created automatically.

- $ sudo mkdir pg-db2 # pg-db2 must be existing before use
- $ docker run -d -p 5436:5432 -e POSTGRES_PASSWORD=123 --mount type=bind,source=/home/seng/pg-db2,target=/app postgres

target = destination; /app will be created if not exist

Spring boot Dockerfile

```
FROM openjdk:11
ADD register-v2.jar ROOT.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "ROOT.jar"]
```

- `docker build -t register-img .`
- `docker run -d -p 8081:8080 -v /home/$USER/my-images:/src/main/resources/images register-img`

Spring boot Dockerfile

```
ARG GRADLE_VERSION=7.6
FROM gradle:${GRADLE_VERSION} AS builder
WORKDIR /app
# Copy necessary directory
COPY build.gradle ./build.gradle
COPY settings.gradle ./settings.gradle
COPY src ./src
# COPY . .
RUN gradle build -x test
# -x test : means skip the test
# serve
FROM openjdk:17
ARG PORT=8080
ENV PORT=${PORT}
WORKDIR /app
COPY --from=builder /app/build/libs/*.jar  app.jar
VOLUME [ "/app/filestorage/images" ]
EXPOSE 8080
ENTRYPOINT ["java", "-jar","app.jar","--server.port=${PORT}"]

```

If you want to copy files inside your volumes

```
docker volume create image-storage-vol
# now that you have the images store inside the volumes
docker run --rm \
    -v image-storage-vol:/data \
    -v $(pwd)/backup-images:/backup \
    alpine \
    sh -c "cp -r /data/. /backup"
```

- You can also use cp commands at the ***mountPoints*** to copy the files and content inside

Bonus with the database postgres backup #01

```
33  docker volume create new-postgres-vol
34  docker run -d -p 5432:5432 \
35  --name new-postgres-cont \
36  -e POSTGRES_PASSWORD=password \
37  -v new-postgres-vol:/var/lib/postgresql/data \
38  postgres
```

Bonus with the database postgres backup #01

```
4  docker run --rm \
5    -v pgdata:/volume \
6    -v $(pwd):/backup \
7    alpine \
8    sh -c "tar czf /backup/pgdata-backup.tar.gz -C /volume ."
9
10 # Option 2
11 docker exec -t d027490f4a1a  pg_dump -U dara -d test > backup.sql
12
13
14 # to populate the database back to the container
15 docker cp backup.sql postgres-cont:/backup.sql
16 docker exec -it postgres-cont psql -U dara -d test -f /backup.sql
```

Resources for the projects test

- https://gitlab.com/devops-trainings3/special-trainning/sample-projects/sample-restful-jpa.git
- https://gitlab.com/devops-trainings3/special-trainning/sample-projects/simple-fileupload-gradle.git
- https://devhints.io/yaml
- https://quickref.me/yaml.html

Thank you

Begin with the theory, Start with the practical