04 – FROST docker deployment

Martina Forconi

Deploying FROST-Server using Docker

https://fraunhoferiosb.github.io/FROST-Server/deployment/docker.html

You can run FROST-Server and the needed database inside one or multiple Docker containers.

You need to install **docker** and **docker-compose**

Steps:

1. Download docker-compose file:

   ```
   wget https://raw.githubusercontent.com/FraunhoferIOSB/FROST-Server/v2.x/scripts/docker-compose.yaml
   ```

2. Start the server with docker: ***docker-compose up***
3. Fetch a json file with some demo entities:

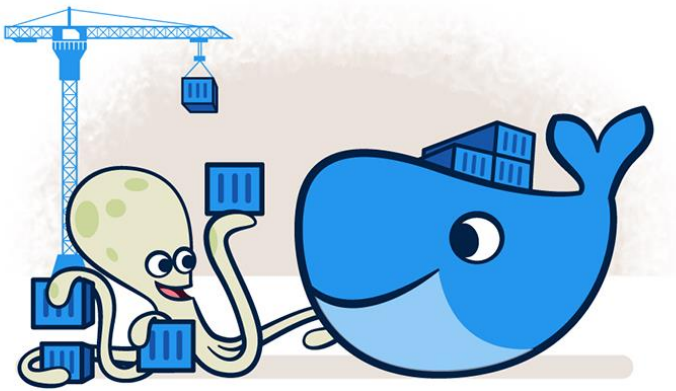
   ```
   wget https://gist.githubusercontent.com/hylkevds/4ffba774fe0128305047b7bcbcd2672e/raw/demoEntities.json
   ```

4. Post it to the server:

   ```
   curl -X POST -H "Content-Type: application/json" -d @demoEntities.json http://localhost:8080/FROST-Server/v1.1/Things
   ```

5. Browse to http://localhost:8080/FROST-Server/v1.0

Docker



Develop faster. Run anywhere.

Docker is an open platform for developing, shipping, and running applications.

Docker provides the ability to package and run an application in a **loosely isolated environment** called a **container**.

The isolation and security allows you to run many containers simultaneously on a given host.

Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host

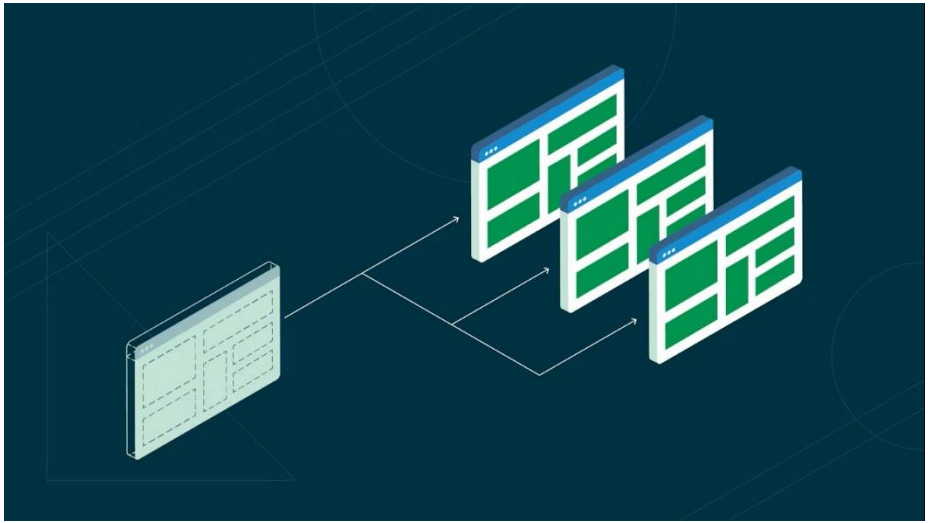
https://www.docker.com/

What is a Docker image?

Images are read-only templates containing instructions for creating a container. A Docker image creates containers to run on the Docker platform.

Think of an image like a blueprint or snapshot of what will be in a container when it runs.

You can manually build images using a **Dockerfile**, a text document containing all the commands to create a Docker image.
You can also pull images from a central repository called a registry, or from repositories like Docker Hub using the command ***docker pull [name]***.



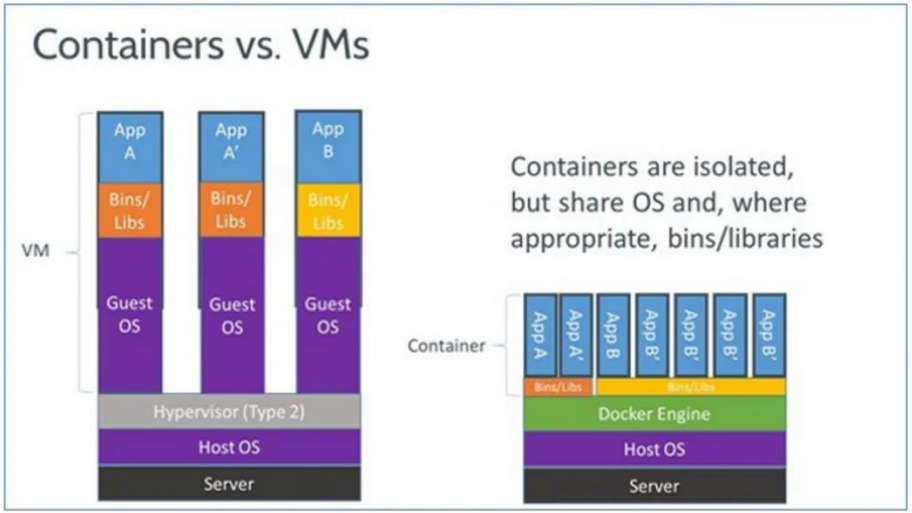
What is a Docker container?

A container is a runnable instance of an image.

You can create, start, stop, move, or delete a container. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host.

By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine.

A container is defined by its image as well as any configuration options you provide to it when you create or start it.



https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/

Docker-compose

Compose is a tool for defining and **running multi-container** Docker applications.

With Compose, you use a **YAML** file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

With Docker compose, **you can configure and start multiple containers with a single yaml file**.

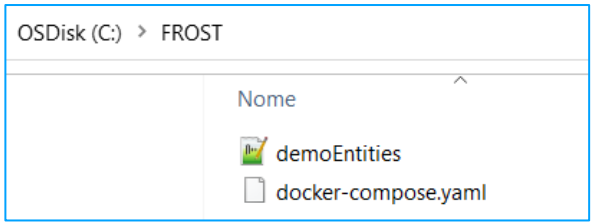
Under the services section we will list all the types of applications to be configured.

```yaml
version: '3'

services:
  web:
    image: fraunhoferiosb/frost-server:2.0
    environment:
      - serviceRootUrl=http://localhost:8080/FROST-Server
      - plugins.multiDatastream.enable=true
      - http_cors_enable=true
      - http_cors_allowed_origins=*
      - persistence_db_driver=org.postgresql.Driver
      - persistence_db_url=jdbc:postgresql://database:5432/sensorthings
      - persistence_db_username=sensorthings
      - persistence_db_password=ChangeMe
      - persistence_autoUpdateDatabase=true
    ports:
      - 8080:8080
      - 1883:1883
    depends_on:
      - database

  database:
    image: postgis/postgis:14-3.2-alpine
    environment:
      - POSTGRES_DB=sensorthings
      - POSTGRES_USER=sensorthings
      - POSTGRES_PASSWORD=ChangeMe
    volumes:
      - postgis_volume:/var/lib/postgresql/data
volumes:
    postgis_volume:
```

https://raw.githubusercontent.com/FraunhoferIOSB/FROST-Server/v2.x/scripts/docker-compose.yaml

1. Download FROST docker-compose file

Download docker-compose file:
https://raw.githubusercontent.com/FraunhoferIOSB/FROST-Server/v2.x/scripts/docker-compose.yaml




```yaml
version: '3'

services:
  web:
    image: fraunhoferiosb/frost-server:2.0
    environment:
      - serviceRootUrl=http://localhost:8080/FROST-Server
      - plugins.multiDatastream.enable=true
      - http_cors_enable=true
      - http_cors_allowed_origins=*
      - persistence_db_driver=org.postgresql.Driver
      - persistence_db_url=jdbc:postgresql://database:5432/sensorthings
      - persistence_db_username=sensorthings
      - persistence_db_password=ChangeMe
      - persistence_autoUpdateDatabase=true
    ports:
      - 8080:8080
      - 1883:1883
    depends_on:
      - database

  database:
    image: postgis/postgis:14-3.2-alpine
    environment:
      - POSTGRES_DB=sensorthings
      - POSTGRES_USER=sensorthings
      - POSTGRES_PASSWORD=ChangeMe
    volumes:
      - postgis_volume:/var/lib/postgresql/data
volumes:
    postgis_volume:
```

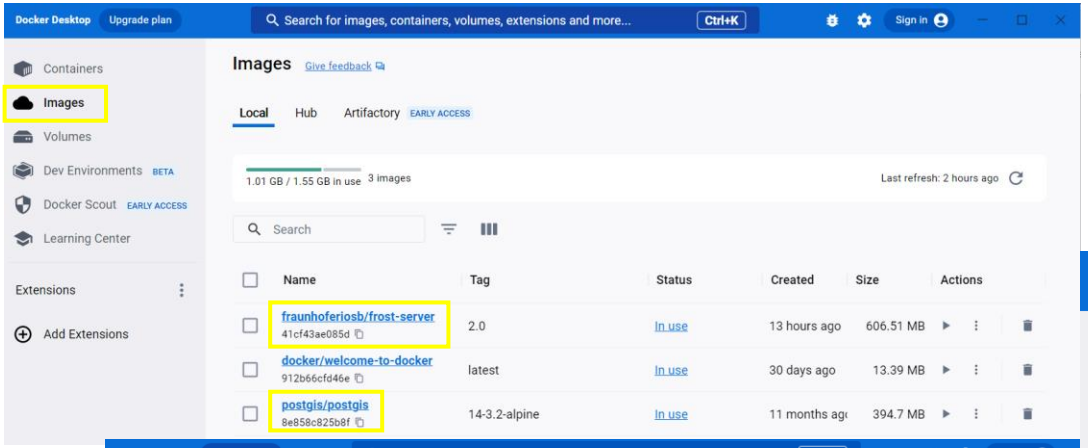
https://raw.githubusercontent.com/FraunhoferIOSB/FROST-Server/v2.x/scripts/docker-compose.yaml

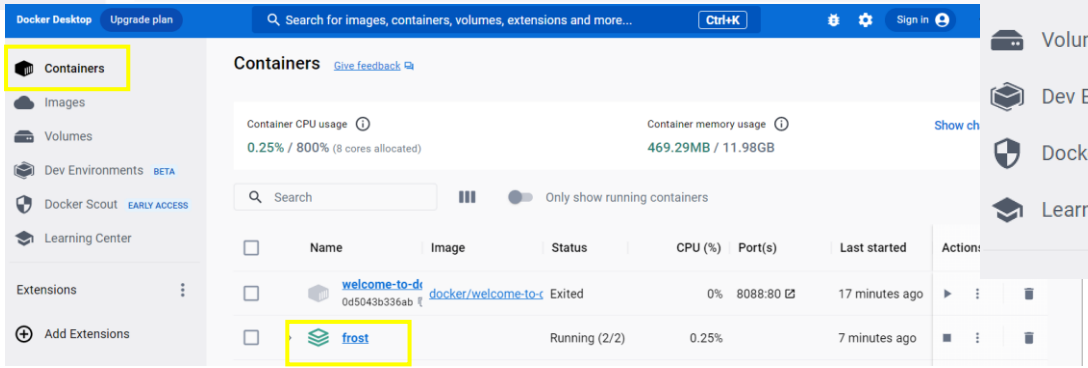
Start the server with docker: ***docker-compose up***

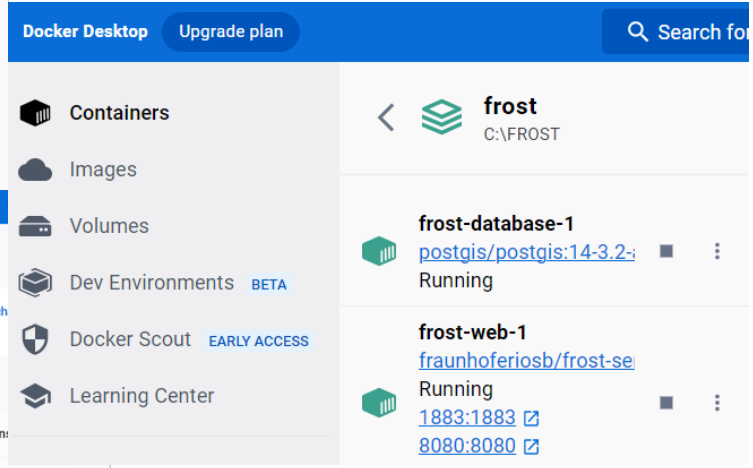
```
C:\FROST>docker-compose up
[+] Running 23/23
 ⠿ web 10 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled
   ⠿ 9d19ee268e0d Pull complete
   ⠿ 32db0ad82863 Pull complete
   ⠿ 1acd9f0b851b Pull complete
   ⠿ 715f805aa7a7 Pull complete
   ⠿ a5afda829b0d Pull complete
   ⠿ a4ed0a1542d1 Pull complete
   ⠿ 269f42b70319 Pull complete
   ⠿ e98948810141 Pull complete
   ⠿ 48a011f3d287 Pull complete
   ⠿ 20bd3218a01f Pull complete
 ⠿ database 11 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled
   ⠿ 213ec9aee27d Pull complete
   ⠿ 85c3ef7cf9a6 Pull complete
   ⠿ ac29cc04759a Pull complete
   ⠿ 2a37e244d86b Pull complete
   ⠿ 36d7202aa1cf Pull complete
   ⠿ 3acdddb9790a Pull complete
   ⠿ 9a938759f2bf Pull complete
   ⠿ 5d65a6241248 Pull complete
   ⠿ dbb70fb41fb6 Pull complete
   ⠿ 67d6b097c6c7 Pull complete
   ⠿ bd335a8171c6 Pull complete
```

```
frost-database-1  | PostgreSQL init process complete; ready for start up.
frost-database-1  |
frost-database-1  |
frost-database-1  | 2023-07-20 14:02:22.959 UTC [1] LOG:  starting PostgreSQL 14.5 on x86_64-pc-linux-musl,
ed by gcc (Alpine 11.2.1_git20220219) 11.2.1 20220219, 64-bit
frost-database-1  | 2023-07-20 14:02:22.959 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
frost-database-1  | 2023-07-20 14:02:22.959 UTC [1] LOG:  listening on IPv6 address "::", port 5432
frost-database-1  | 2023-07-20 14:02:22.964 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.P
432"
frost-database-1  | 2023-07-20 14:02:22.969 UTC [59] LOG:  database system was shut down at 2023-07-20 14:02
C
frost-database-1  | 2023-07-20 14:02:22.974 UTC [1] LOG:  database system is ready to accept connections
frost-web-1       | 14:02:23.025 [            main] INFO     d.f.i.i.f.settings.Settings - Not set queueLogg
erval, using default value '0'.
frost-web-1       | 20-Jul-2023 14:02:23.044 INFO [main] org.apache.catalina.startup.HostConfig.deployDirect
ployment of web application directory [/usr/local/tomcat/webapps/FROST-Server] has finished in [4,280] ms
frost-web-1       | 20-Jul-2023 14:02:23.049 INFO [main] org.apache.coyote.AbstractProtocol.start Starting P
lHandler ["http-nio-8080"]
frost-web-1       | 20-Jul-2023 14:02:23.057 INFO [main] org.apache.catalina.startup.Catalina.start Server s
 in [4355] milliseconds
```

FROST multi container



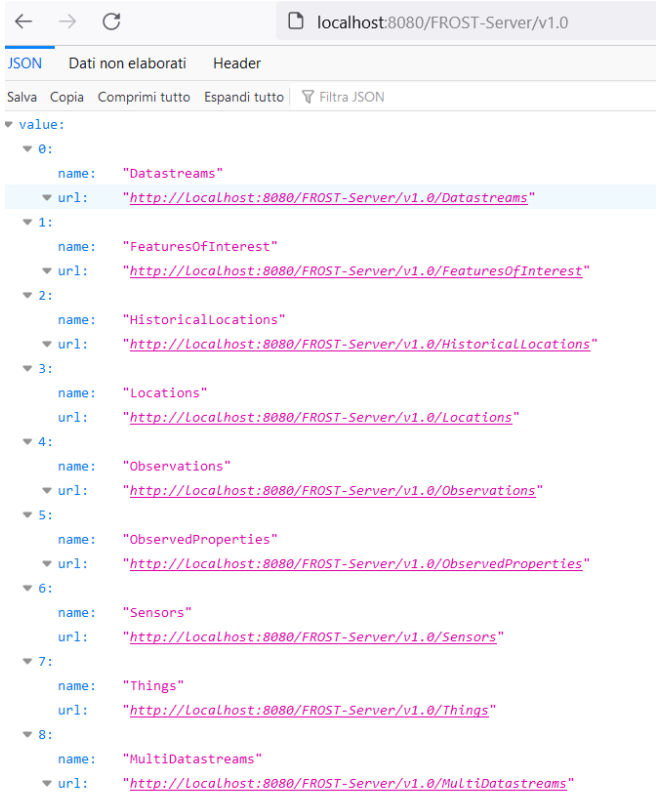




After the docker-compose up you will have:

- 2 images (frost-server e postgis)
- 1 multi container (frost)

FROST server is running



http://localhost:8080/FROST-Server/v1.0

Insert demo entities

1. Fetch a json file with some demo entities:

   wget https://gist.githubusercontent.com/hylkevds/4ffba774fe0128305047b7bcbcd2672e/raw/demoEntities.json

2. Post it to the server:

   curl -X POST -H "Content-Type: application/json" -d @demoEntities.json http://localhost:8080/FROST-Server/v1.1/Things