04 – FROST docker deployment deda.next

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

deda.next

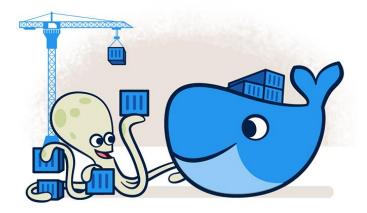
FDIA

4. Post it to the server:

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

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

deda<mark>, next</mark> EDIA®I

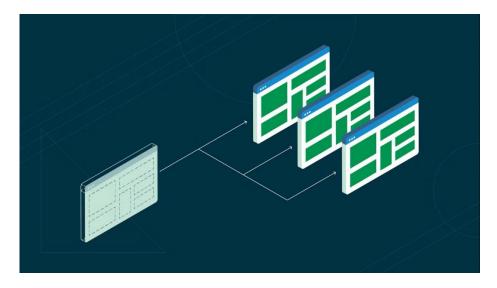
What is a Docker image?

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]*.



deda, next

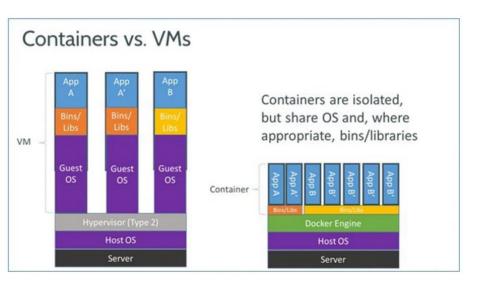
What is a Docker container?

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/

deda, next

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.

```
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

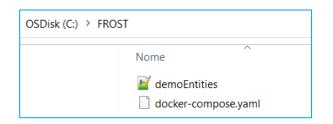
  database:
    image: postgis/postgis:14-3.2-alpine
    environment:
      - POSTGRES DB=sensorthings
      - POSTGRES USER=sensorthings
        POSTGRES PASSWORD=ChangeMe
    volumes:
      - postgis volume:/var/lib/postgresql/data
volumes:
    postgis volume:
```

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

1. Download FROST docker-compose file

Download docker-compose file:

https://raw.githubusercontent.com/FraunhoferIOSB/ FROST-Server/v2.x/scripts/docker-compose.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.postgresgl.Driver - persistence db url=jdbc:postgresgl://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			
2 web 10 layers [PPPPPPP]	0B/0B	Pulled	
9d19ee268e0d Pull complete			
32db0ad82863 Pull complete			
Iacd9f0b851b Pull complete			
2 715f805aa7a7 Pull complete			
I a5afda829b0d Pull complete			
2 a4ed0a1542d1 Pull complete			
269f42b70319 Pull complete			
2 e98948810141 Pull complete			frost-database-1 PostgreSQL init process complete; ready for start up. frost-database-1
2 48a011f3d287 Pull complete			frost-database-1
20bd3218a01f Pull complete			frost-database-1 2023-07-20 14:02:22.959 UTC [1] LOG: starting PostgreSQL 14.5 on x86_64-pc-linux-musl,
<pre>2 database 11 layers [PP2P2P2P2</pre>	0B	/0B Pulled	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
213ec9aee27d Pull complete	-		frost-database-1 2023-07-20 14:02:22.959 UTC [1] LOG: listening on IPv6 address "0.0.0.0", port 5432
85c3ef7cf9a6 Pull complete			frost-database-1 2023-07-20 14:02:22.964 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.P
🛛 ac29cc04759a Pull complete			
2a37e244d86b Pull complete			frost-database-1 2023-07-20 14:02:22.969 UTC [59] LOG: database system was shut down at 2023-07-20 14:02
36d7202aa1cf Pull complete			frost-database-1 2023-07-20 14:02:22.974 UTC [1] LOG: database system is ready to accept connections
2 3acdddb9790a Pull complete			frost-web-1 14:02:23.025 [main] INFO d.f.i.i.f.settings.Settings - Not set queueLogg
2 9a938759f2bf Pull complete			erval, using default value '0'. <pre>frost-web-1</pre> 20-Jul-2023 14:02:23.044 INFO [main] org.apache.catalina.startup.HostConfig.deployDirect
2 5d65a6241248 Pull complete			ployment of web application directory [/usr/local/tomcat/webapps/FROST-Server] has finished in [4,280] ms
2 dbb70fb41fb6 Pull complete			frost-web-1 20-Jul-2023 14:02:23.049 INFO [main] org.apache.coyote.AbstractProtocol.start Starting P
2 67d6b097c6c7 Pull complete			lHandler ["http-nio-8080"] frost-web-1 20-Jul-2023 14:02:23.057 INFO [main] org.apache.catalina.startup.Catalina.start Server s
2 bd335a8171c6 Pull complete			in [4355] milliseconds

deda, next

deda next

FROST multi container

Docker Desktop Upgrade plan	Q. Search for images, containers, volumes, extensions and more Ctrl+K 🕴 🏟 Sign in 😝 – 🗆						□ ×	After the docker-compose up you will have:								
Containers	Images Give feedback a • 2 images (frost-server e postgis)								С.							
Images Volumes	Local Hub Artifactory EARLY ACCESS						 1 multi container (frost) 									
Dev Environments BETA	1.01 GB / 1.55 G	B in use 3 images					Last refree	sh: 2 hours ago	C							
Docker Scout EARLY ACCESS Learning Center	Q Search		≂ Ш													
Extensions	Name		Тад		Status	Created	Size	Actions		Docker Desktop	Upgrade plan				Q Sea	arch foi
Add Extensions	41cf43ae	feriosb/frost-serv 085d 🗅	2.0		In use	13 hours ago	606.51 MB	▶ 1								
	912b66ct	welcome-to-dock	er latest		In use	30 days ago	13.39 MB	▶ :		Containe	ers	<		frost C:\FROST		
		/postgis 25b8f 10	14-3.2-alp	bine	<u>In use</u>	11 months age	394.7 MB	► ±		Images				0.1.1.001		
Docker Desktop	Upgrade plan	Q Search Containers		iners, volumes, extens	ions and more	Ctrl+K		8 🗘 🔇	Sign in	 Volumes Dev Envi 	ronments BETA			latabase-1 <u>s/postgis:14-3</u> ng	3.2-: ■	:
Volumes Volumes Dev Environr	ments BETA	Container CPU u 0.25% / 8009	usage (j) % (8 cores allocated)			Container memory usa			Show ch	A	SCOUT EARLY ACCESS		frost-v	web-1 oferiosb/frost	-50	
Docker Scou		Q Search			Only show running					🗢 Learning	Center		Runnin			:
			ime	Image	Status	CPU (%) Po	rt(s)	Last starte	d Action	1			<u>8080:8</u>	<u>3080</u> 🖸		
Extensions	:		0d5043b336ab	docker/welcome-to-c	Exited	0% 80	88:80 🖾	17 minutes	ago 🕨							
Add Extension	ons		≩ <u>frost</u>		Running (2/2)	0.25%		7 minutes a	ago 🔳							

deda<mark>, next</mark> EDIA©I

FROST server is running

$\leftarrow \rightarrow$	C localhost:8080/FROST-Server/v1.0
JSON Dati r	ion elaborati Header
Salva Copia C	comprimi tutto Espandi tutto 🛛 🗑 Filtra JSON
value:	
▼ 0:	
name:	"Datastreams"
▼ url:	" <pre>http://localhost:8080/FROST-Server/v1.0/Datastreams"</pre>
▼ 1:	
name:	"FeaturesOfInterest"
▼ url:	"http://localhost:8080/FROST-Server/v1.0/FeaturesOfInterest"
▼ 2:	
name:	"HistoricalLocations"
▼ url:	"http://localhost:8080/FROST-Server/v1.0/HistoricalLocations"
▼ 3:	
name:	"Locations"
url:	"http://localhost:8080/FROST-Server/v1.0/Locations"
▼ 4:	
name:	"Observations"
▼ url:	" <pre>http://localhost:8080/FROST-Server/v1.0/Observations"</pre>
▼ 5:	
name:	"ObservedProperties"
▼ url:	" <pre>http://localhost:8080/FROST-Server/v1.0/ObservedProperties"</pre>
▼ 6:	
name:	"Sensors"
url:	" <u>http://localhost:8080/FROST-Server/v1.0/Sensors</u> "
▼ 7:	
name:	"Things"
url:	" <u>http://localhost:8080/FROST-Server/v1.0/Things</u> "
▼ 8:	
name:	"MultiDatastreams"
▼ url:	" <u>http://localhost:8080/FROST-Server/v1.0/MultiDatastreams</u> "

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

deda<mark>, next</mark> EDIA©I

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 <u>http://localhost:8080/FROST-</u> Server/v1.1/Things

\leftarrow \rightarrow C \square localhos	t:8080/FROST-Server/v1.0/Locations
JSON Dati non elaborati Header	
Salva Copia Comprimi tutto Espandi tutto 🕅 Filtra JSON	N
<pre>@iot.count:</pre>	1
<pre>value:</pre>	
▼ 0:	
<pre> @iot.selfLink: </pre>	" <pre>http://localhost:8080/FROST-Server/v1.0/Locations(1)"</pre>
@iot.id:	1
name:	"My Living Room"
description:	"The living room of Fraunhoferstr. 1"
encodingType:	"application/vnd.geo+json"
<pre>v location:</pre>	
type:	"Point"
▼ coordinates:	
0:	8.4259727
1:	49.015308
HistoricalLocations@iot.navigationLink:	"http://localhost:8080/FROST-Server/v1.0/Locations(1)/HistoricalLocations"
Things@iot.navigationLink:	" <u>http://localhost:8080/FROST-Server/v1.0/Locations(1)/Things</u> "