**DevOps and Cloud**

**March 2025**



# Advanced Containerization Concepts

Homework (M2)

Vasil Atanasov

@VasAtanasov

## Summary

To repeat the steps from the practice, I will use a PostgreSQL Docker image. One key point, which will become evident in the steps involving volumes, is that each PostgreSQL container instance sharing a common data volume with another instance will cause the PostgreSQL server to detect this and create a copy of the volume or fail. If the database is modified by one server, the other server will not see the changes. To see the changes a custom docker image where only PostgreSQL client is installed whit which we will connect to the remote server will be queried.

Postgres Docker Image: <https://github.com/docker-library/postgres>

## Network and Volumes

### Networks

#### Default Network

Containers on the default bridge network cannot resolve each other by container name, unlike user-defined bridge networks.

docker network ls

docker network inspect bridge

docker run -d \

--name postgres1 \

-e POSTGRES\_PASSWORD=Password1 \

-e POSTGRES\_DB=bulgaria \

-p 5432:5432 \

postgres

docker run -d \

--name postgres2 \

-e POSTGRES\_PASSWORD=Password1 \

-e POSTGRES\_DB=bulgaria \

-p 5433:5432 \

postgres

docker network inspect bridge -f '{{range .Containers}}{{.Name}}, {{println .IPv4Address}}{{end}}'

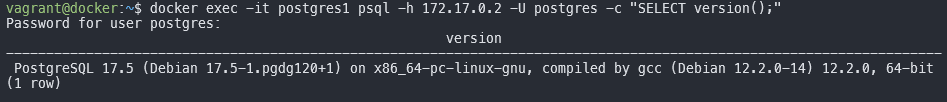
A screenshot of a computer

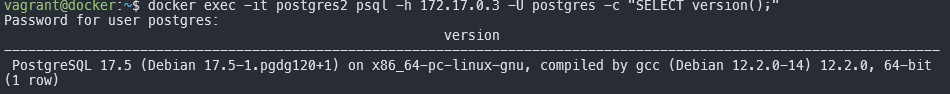
AI-generated content may be incorrect.

We can query for the version of the PostgreSQL server running on postgres1 from postgres2 container using only IP address of the container because default bridge network does not support name resolution.

docker exec -it postgres1 psql -h 172.17.0.2 -U postgres -c "SELECT version();"

docker exec -it postgres2 psql -h 172.17.0.3 -U postgres -c "SELECT version();"





If we try to use the container name as host, we will get an error.



#### Custom Network

Now we will create a custom bridge network **pg-net.**

docker network create -d bridge --subnet 10.0.0.0/24 pg-net

The two PostgreSQL containers are running so we can connect them to the new network.

docker network connect pg-net postgres1

docker network connect pg-net postgres2

Or we can stop them and start them again with the –network option passing the name of the network.

docker run -d \

--name postgres1 \

--network pg-net \

-e POSTGRES\_DB=bulgaria \

-e POSTGRES\_PASSWORD=Password1 \

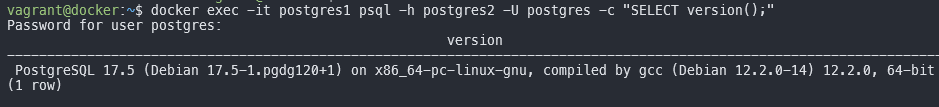
-p 5432:5432 \

postgres

docker network inspect pg-net -f '{{range .Containers}}{{.Name}}, {{println .IPv4Address}}{{end}}'

A computer screen with white text

AI-generated content may be incorrect.

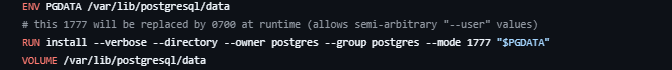


### Volumes

#### On The Fly

Note: Postgres Dockerfile has an instruction to create a volume that is why for the two containers there are volumes already created on the fly when run.

<https://github.com/docker-library/postgres/blob/38b3c10a487945e08b7f63dee25dc4f7b86a79d1/Dockerfile-debian.template#L200>



That is why when the image is run it will create a volume on the fly.

A screenshot of a computer program

AI-generated content may be incorrect.

We will populate the database and start a second container using the volume from the first container.

docker exec -i postgres1 psql -U postgres -d bulgaria <<-EOSQL

CREATE TABLE cities (

id SERIAL PRIMARY KEY,

city\_name VARCHAR(50),

population INT

);

INSERT INTO cities (city\_name, population) VALUES ('София', 1248452);

INSERT INTO cities (city\_name, population) VALUES ('Пловдив', 343070);

INSERT INTO cities (city\_name, population) VALUES ('Варна', 332686);

INSERT INTO cities (city\_name, population) VALUES ('Бургас', 199571);

INSERT INTO cities (city\_name, population) VALUES ('Русе', 137533);

INSERT INTO cities (city\_name, population) VALUES ('Стара Загора', 124599);

INSERT INTO cities (city\_name, population) VALUES ('Плевен', 93214);

INSERT INTO cities (city\_name, population) VALUES ('Сливен', 83740);

INSERT INTO cities (city\_name, population) VALUES ('Добрич', 79269);

INSERT INTO cities (city\_name, population) VALUES ('Шумен', 72342);

EOSQL

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

We can see still only one volume exists.

If we query the database, we will get:

docker exec -it postgres2 psql \

-h postgres1 \

-U postgres \

-d bulgaria \

-c "SELECT \* FROM cities;"

A screenshot of a computer

AI-generated content may be incorrect.

#### Attach a Prepopulated (Existing) Folder

Due to the nature of Postgres, shared data directory is not allowed, and Postgres makes a copy of the data. If a row is deleted in the first container database, it will not affect the database of the second container. It is recommended to connect to the database container as a client. This way only one server manages the db. That is why we will run the first container, populate it with data and then run the second container. When we query the database in the second container we will see there is data there.

docker run -d \

--name postgres1 \

--network pg-net \

-e POSTGRES\_DB=bulgaria \

-e POSTGRES\_PASSWORD=Password1 \

-p 5432:5432 \

-v $(pwd)/postgres-data:/var/lib/postgresql/data \

postgres

docker exec -i postgres1 psql -U postgres -d bulgaria <<-EOSQL

CREATE TABLE cities (

id SERIAL PRIMARY KEY,

city\_name VARCHAR(50),

population INT

);

INSERT INTO cities (city\_name, population) VALUES ('София', 1248452);

INSERT INTO cities (city\_name, population) VALUES ('Пловдив', 343070);

INSERT INTO cities (city\_name, population) VALUES ('Варна', 332686);

INSERT INTO cities (city\_name, population) VALUES ('Бургас', 199571);

INSERT INTO cities (city\_name, population) VALUES ('Русе', 137533);

INSERT INTO cities (city\_name, population) VALUES ('Стара Загора', 124599);

INSERT INTO cities (city\_name, population) VALUES ('Плевен', 93214);

INSERT INTO cities (city\_name, population) VALUES ('Сливен', 83740);

INSERT INTO cities (city\_name, population) VALUES ('Добрич', 79269);

INSERT INTO cities (city\_name, population) VALUES ('Шумен', 72342);

EOSQL

docker run -d \

--name postgres2 \

--network pg-net \

-e POSTGRES\_DB=bulgaria \

-e POSTGRES\_PASSWORD=Password1 \

-p 5433:5432 \

-v $(pwd)/postgres-data:/var/lib/postgresql/data \

postgres

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a computer code

AI-generated content may be incorrect.

#### Dedicated Volume

docker volume create postgres-data

A screen shot of a computer

AI-generated content may be incorrect.

As be before we will start the first PostgreSQL instance populate the data and then start the second instance.

docker run -d \

--name postgres1 \

--network pg-net \

-e POSTGRES\_DB=bulgaria \

-e POSTGRES\_PASSWORD=Password1 \

-p 5432:5432 \

-v postgres-data:/var/lib/postgresql/data \

postgres

Population of data omitted for brevity.

docker run -d \

--name postgres2 \

--network pg-net \

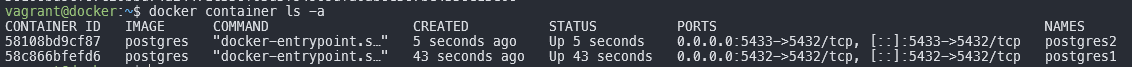
-e POSTGRES\_DB=bulgaria \

-e POSTGRES\_PASSWORD=Password1 \

-p 5433:5432 \

-v postgres-data:/var/lib/postgresql/data \

postgres



A screenshot of a computer

AI-generated content may be incorrect.

#### Volume Containers

To be useful the database volume we will start the PostgreSQL server, populate it with data and then stop it and use its volume to start other instances.

docker run -d \

--name bulgaria-db-base \

--network pg-net \

-e POSTGRES\_DB=bulgaria \

-e POSTGRES\_PASSWORD=Password1 \

-p 5431:5432 \

-v postgres-data:/var/lib/postgresql/data \

postgres

docker container stop bulgaria-db-base

Population of data omitted for brevity.

docker run -d \

--name postgres1 \

--network pg-net \

-e POSTGRES\_DB=bulgaria \

-e POSTGRES\_PASSWORD=Password1 \

-p 5432:5432 \

--volumes-from bulgaria-db-base \

postgres

## Custom Container Images

### Create image from Dockerfile and add content

For this step we will use the files in the *bulgaria-db* folder. Inside there is a Dockerfile which will extend postgres docker image and using custom population script to populate the database with data on run.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### Publishing an image

docker login