# Running Spring Boot with Docker Compose

Ömer Naci Soydemir

Senior Software Engineer

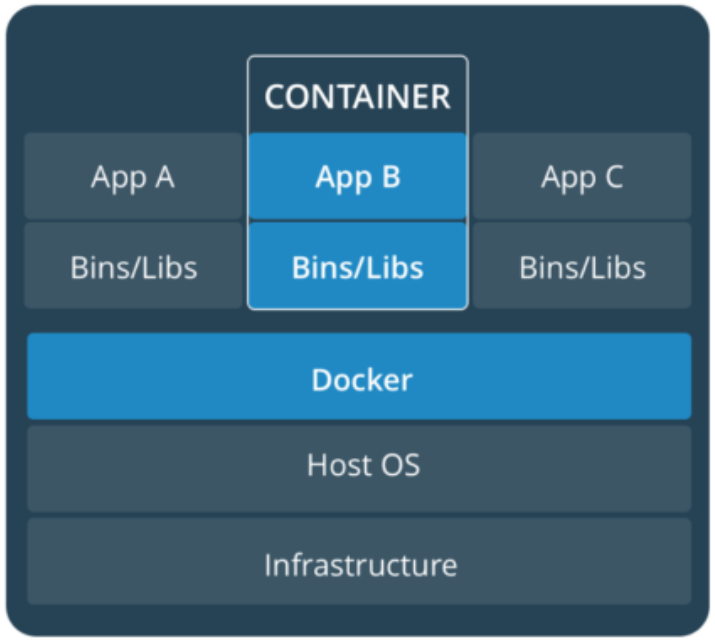
## Roadmap

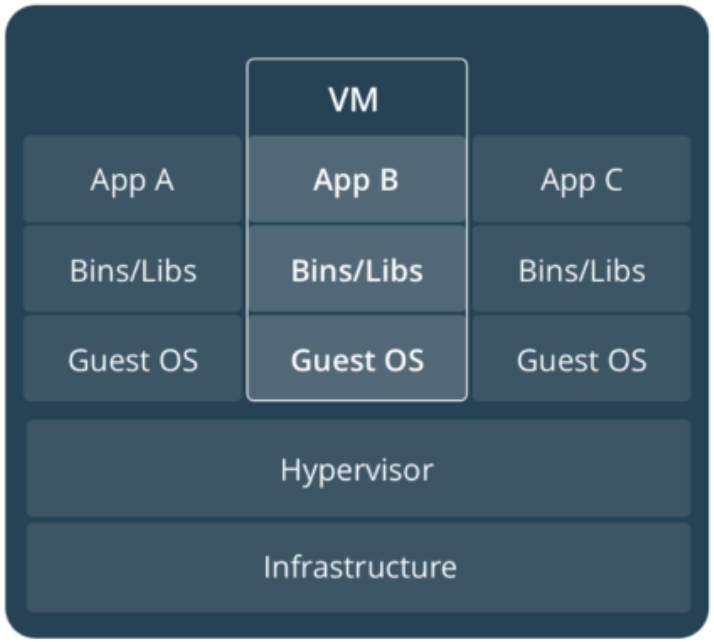
- Introduction to Docker
- Basic Docker Commands
- Docker Compose
- Running Spring Boot with PostgreSQL in Docker Compose
- Conclusion

## Introduction to Docker

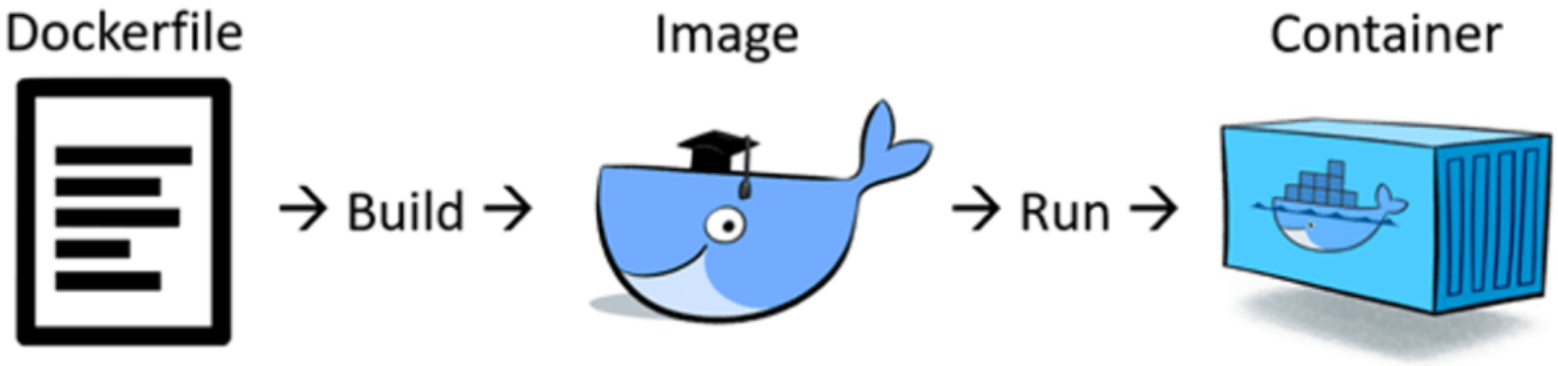
- Docker is basically a container engine that uses the Linux kernel features.
- Docker's virtualization structure does not have a Hypervisor layer.
- It accesses the host operating system through the Docker Engine and uses shared system tools.
- It consumes less system resources than conventional VMs.

# Introduction to Docker



- *Dockerfile* : A script that defines how an application packages to be run as a container.



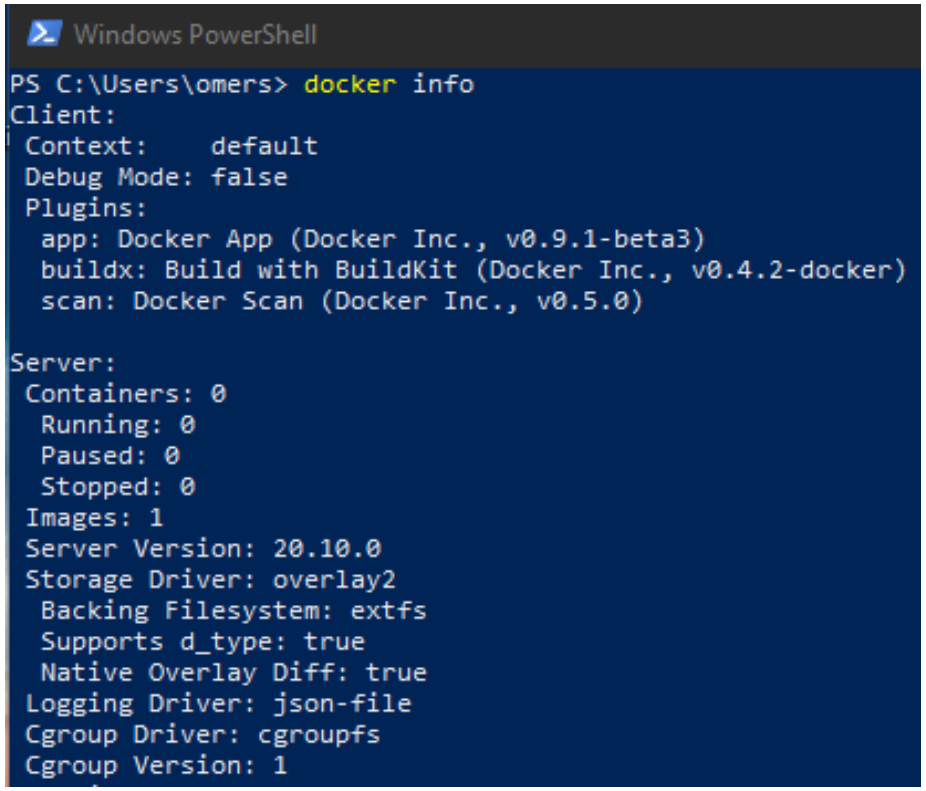

```
FROM adoptopenjdk/openjdk11:jre-11.0.6_10-alpine
COPY build/libs/ /opt/apps/
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/opt/apps/docker-demo-0.0.1-SNAPSHOT.jar"]
```

## Introduction to Docker

- *Docker registry* : It is an environment in which images are kept and distributed.
- *Docker deamon* : Allows containers to work in isolation from each other, makes the use and distribution of resources.
- *Docker image* : An executable application package that the docker service creates that will use dockerfile and base image.
- *Docker container* : Containers are working examples of docker images. Running an image creates a docker container.

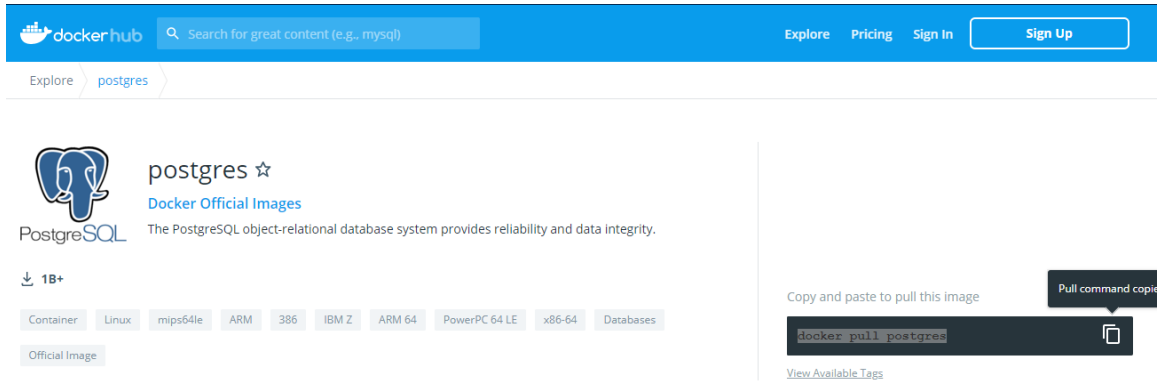
## Basic Docker Commands

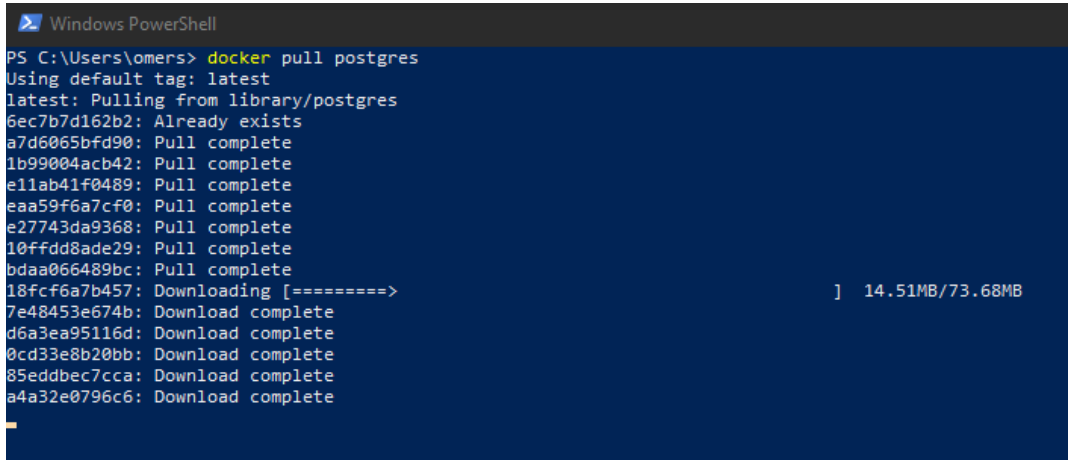
- *docker info* : This command displays system wide information regarding the Docker installation.



# Basic Docker Commands

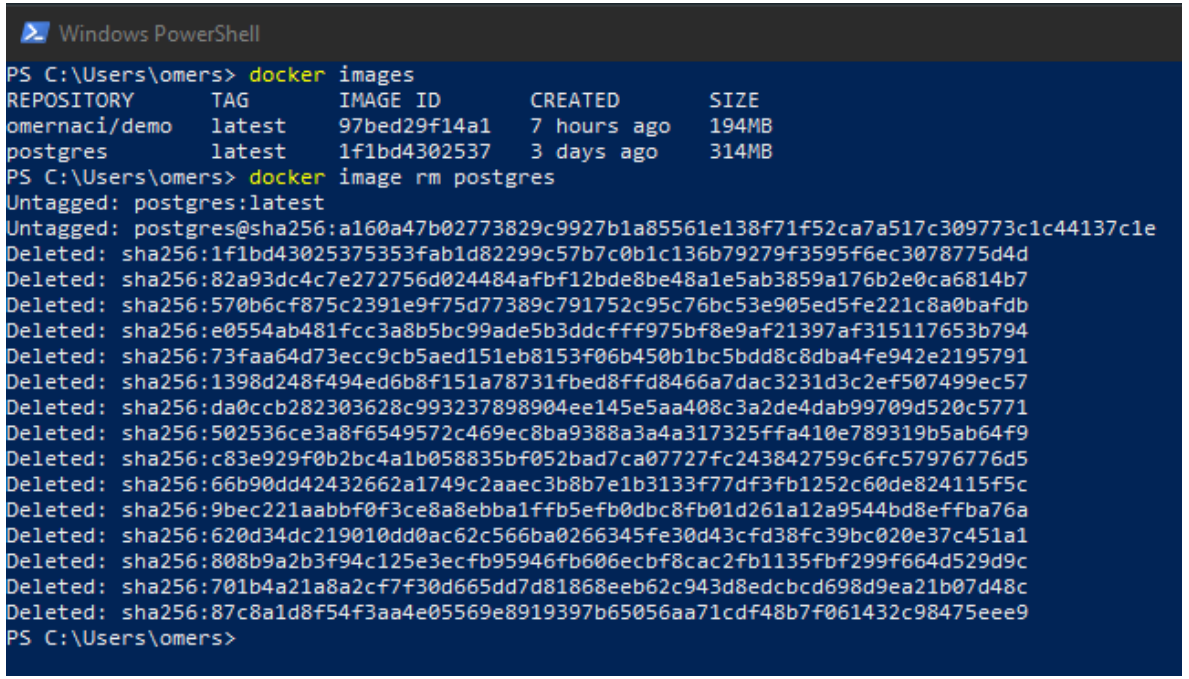
- ***docker pull image_name*** : Download an image, and all its parents, from the registry.



## Basic Docker Commands

- *docker images* : Lists all local docker images.
- *docker image rm image_name (docker rmi image_name)* : Delete an image from the local image store.



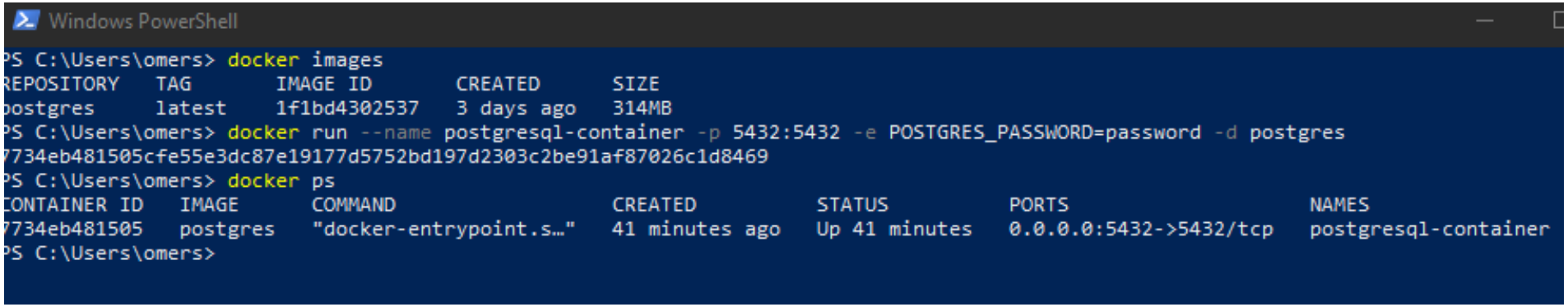
## Basic Docker Commands

- *docker build –t image_tag* : Performs container build according to the definitions in dockerfile.

```
C:\Users\omers\Documents\workspace\workshops\docker\docker-demo>docker build -t omernaci/demo:latest .
[+] Building 1.4s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 32B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/adoptopenjdk/openjdk11:jre-11.0.6_10-alpine
 => [internal] load build context
 => => transferring context: 120B
 => [1/2] FROM docker.io/adoptopenjdk/openjdk11:jre-11.0.6_10-alpine@sha256:a9295aa69d9dbbeb123c7d8b72a8bef5f3523f89523b77802a4d7f74f33df6a9
 => CACHED [2/2] COPY build/libs/ /opt/apps/
 => exporting to image
 => => exporting layers
 => => writing image sha256:97bed29f14a112eec162ce1b63b181e1f2d48c5fcc1b8680cee5f247a4dc0e6d
 => => naming to docker.io/omernaci/demo:latest
```

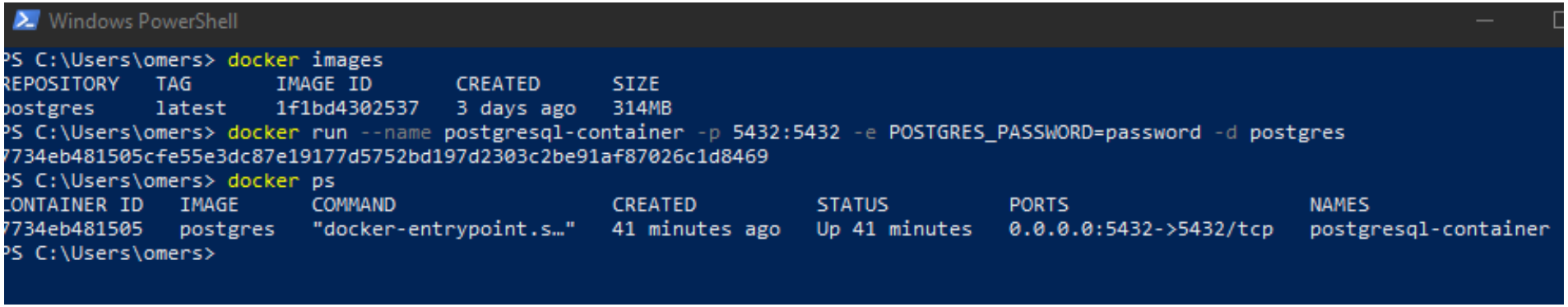
## Basic Docker Commands

- *docker run* : Creates a container from a given image and starts the container using a given command.



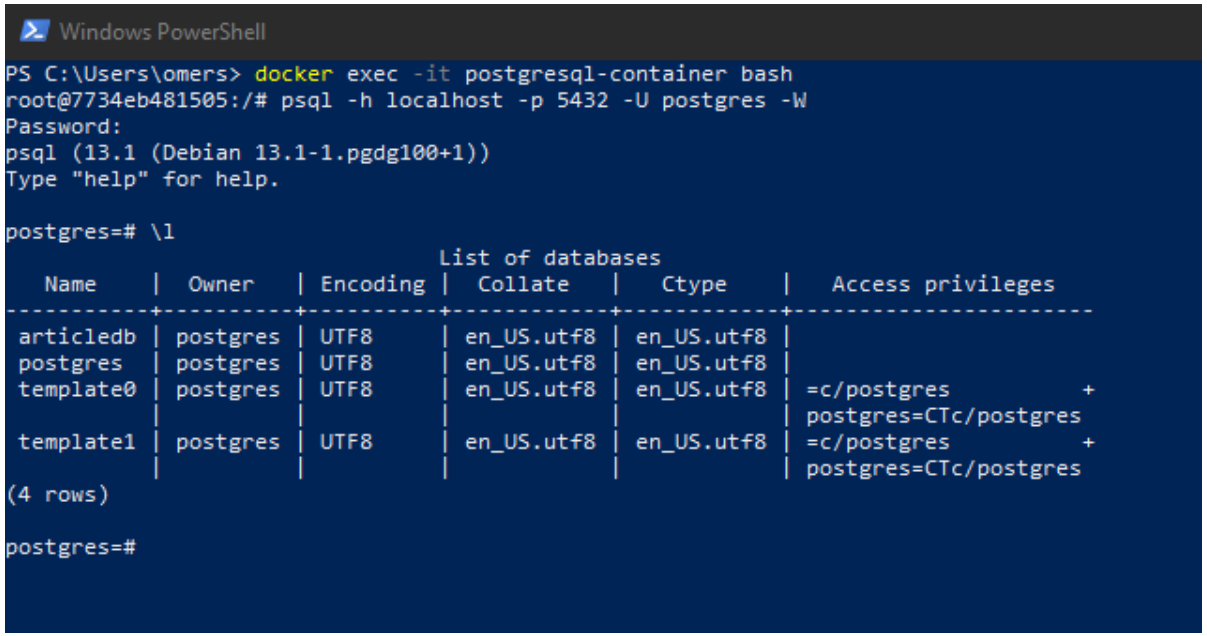
## Basic Docker Commands

- *docker ps (docker container ls)* : Creates a container from a given image and starts the container using a given command.



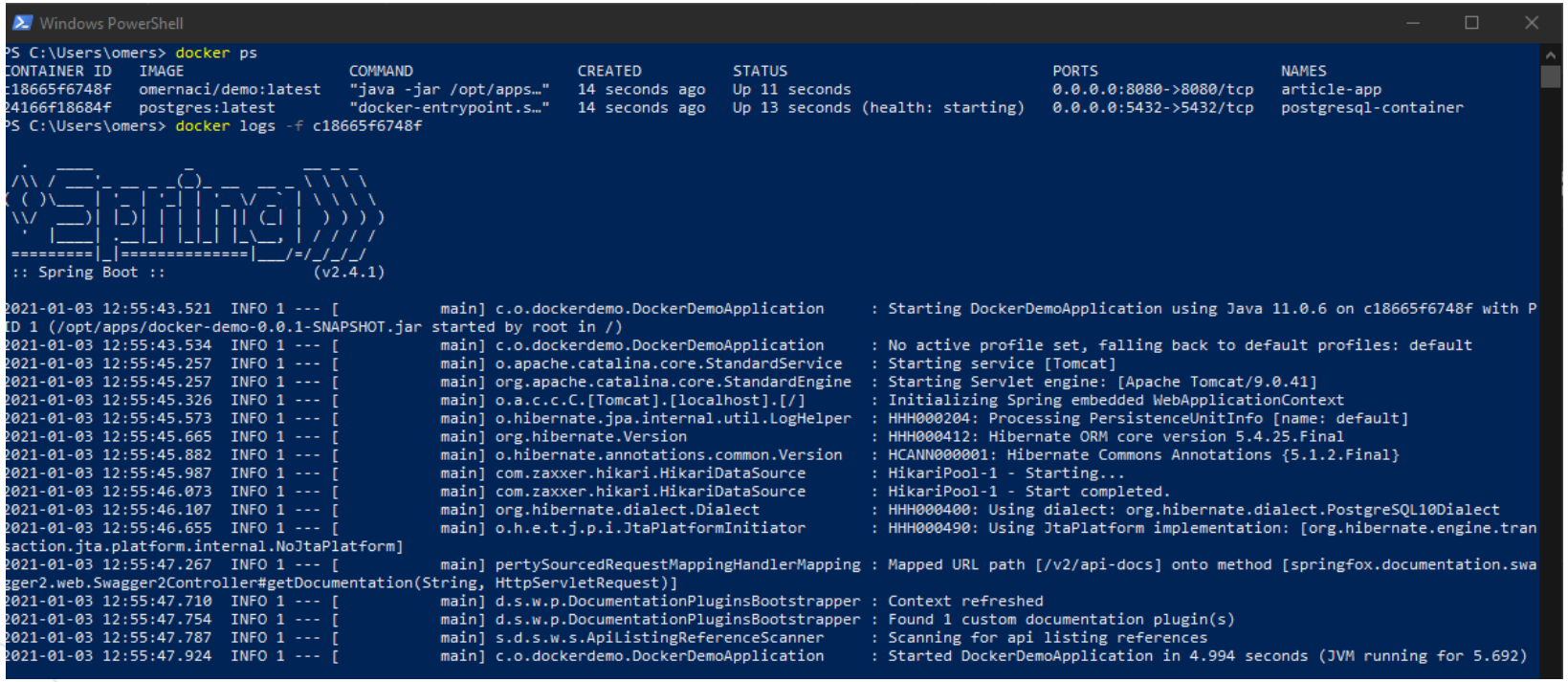
## Basic Docker Commands

- ***docker exec*** : When dealing with the interactive processes like bash, use the -i and -t options to start the container.



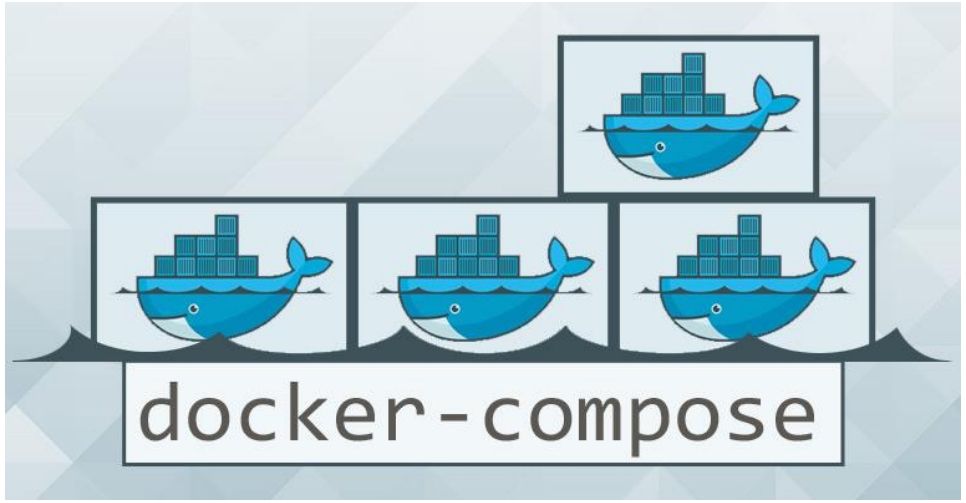
## Basic Docker Commands

- *docker logs* : Displays the logs that have occurred up to that moment in the corresponding container's terminal.
- *-f* continues to show logs formed after that moment with the follow parameter.



## Docker Compose

- Docker compose is a “yml” - based file that allows applications that can run in container to be run and stopped with a single command.
- The *docker-compose* tool is currently offered by Docker Inc, which also provides the Docker CLI and Docker daemon

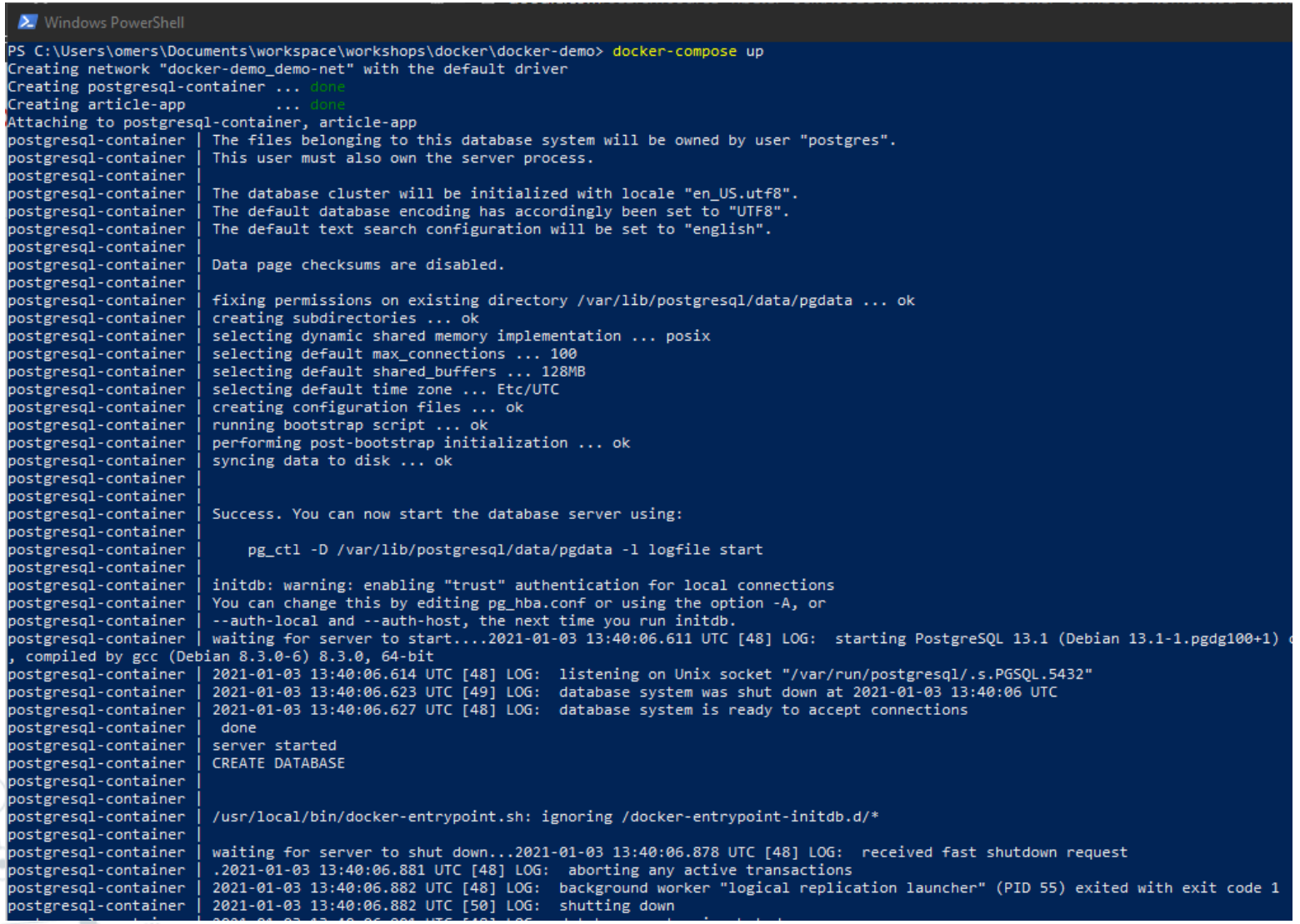


```yaml
version: "3"

networks:
  demo-net:

services:
  # Database Service (Postgres)
  db:
    image: postgres:latest
    container_name: postgresql-container
    restart: always
    #volumes:
    #  - postgres-data:/var/lib/postgresql/data
    networks:
      - demo-net
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: password
      POSTGRES_DB: articledb
      PGDATA: /var/lib/postgresql/data/pgdata
    healthcheck:
      test: pg_isready -U postgres
      interval: 1m
      timeout: 10s
      retries: 2

  # Article Service
  app:
    image: omernaci/demo:latest
    container_name: article-app
    ports:
      - "8080:8080"
    networks:
      - demo-net
    environment:
      - SPRING_DATASOURCE_URL=jdbc:postgresql://db/articledb
      - SPRING_DATASOURCE_USERNAME=postgres
      - SPRING_DATASOURCE_PASSWORD=password
    depends_on:
      - db
```

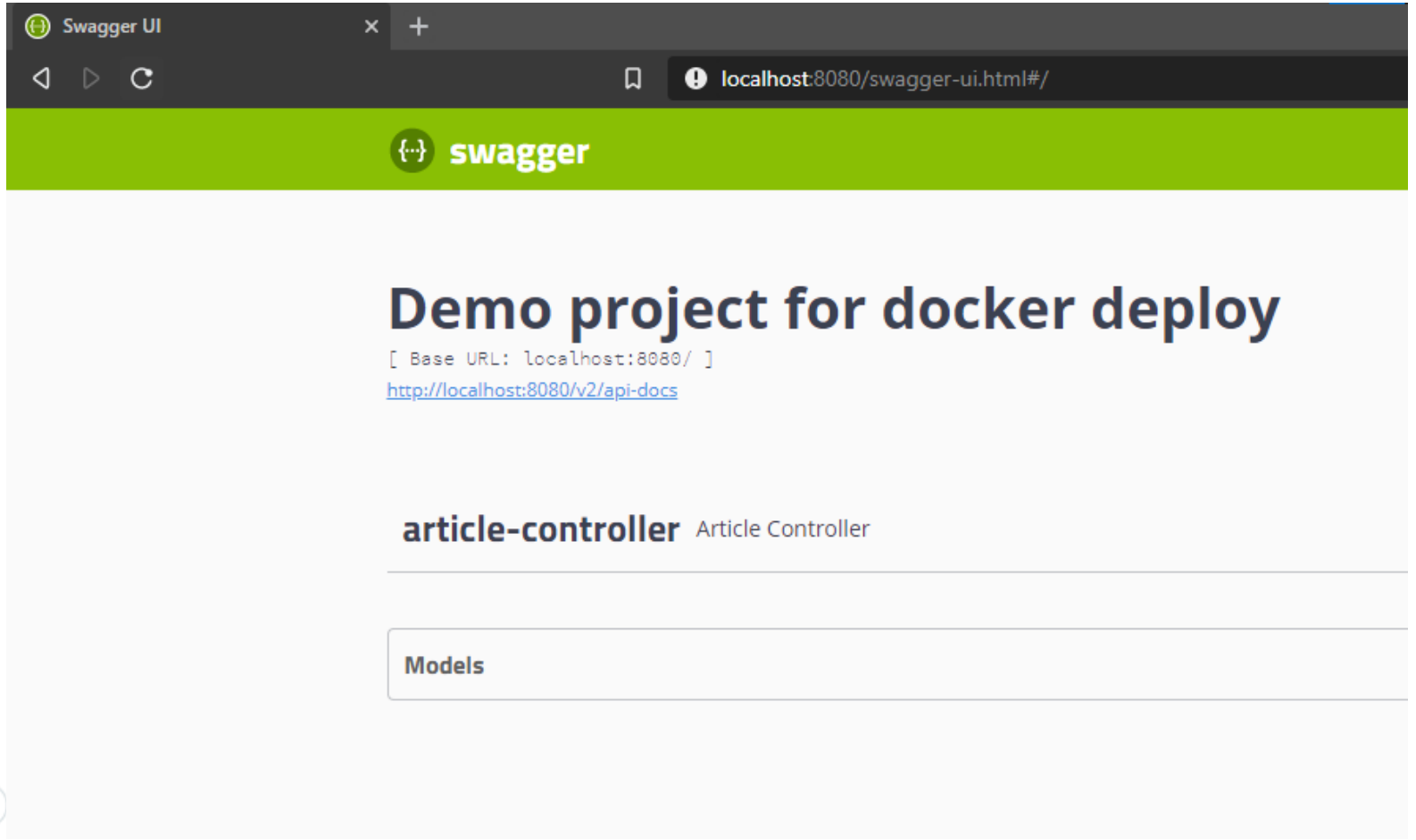
## Running Spring Boot with PostgreSQL in Docker Compose

- *docker-compose up* : It sets up the network for the broadcast of services and stands up all the services contained in the yml file.
- **-d** parameter, docker compose runs in detached mode.
- *docker-compose down :* It completely deletes services.
- *docker-compose stop:* Docker environment does not work, but the network is available.

# Running Spring Boot with PostgreSQL in Docker Compose



# Running Spring Boot with PostgreSQL in Docker Compose



## Conclusion

- It starts in seconds.
- The containers it contains are just a process its lightweight structure makes Docker fast.
- Enables faster deployment process.
- Programmers who want to work on different servers just need to run Docker images.

Time for Questions!