How to Run/Deploy Spring Boot & Postgres SQL in multi-container Docker applications.

In this article, I am going to explain how to deploy a RESTful Spring Boot application and Postgres sql data base on docker container.

## You Will Learn

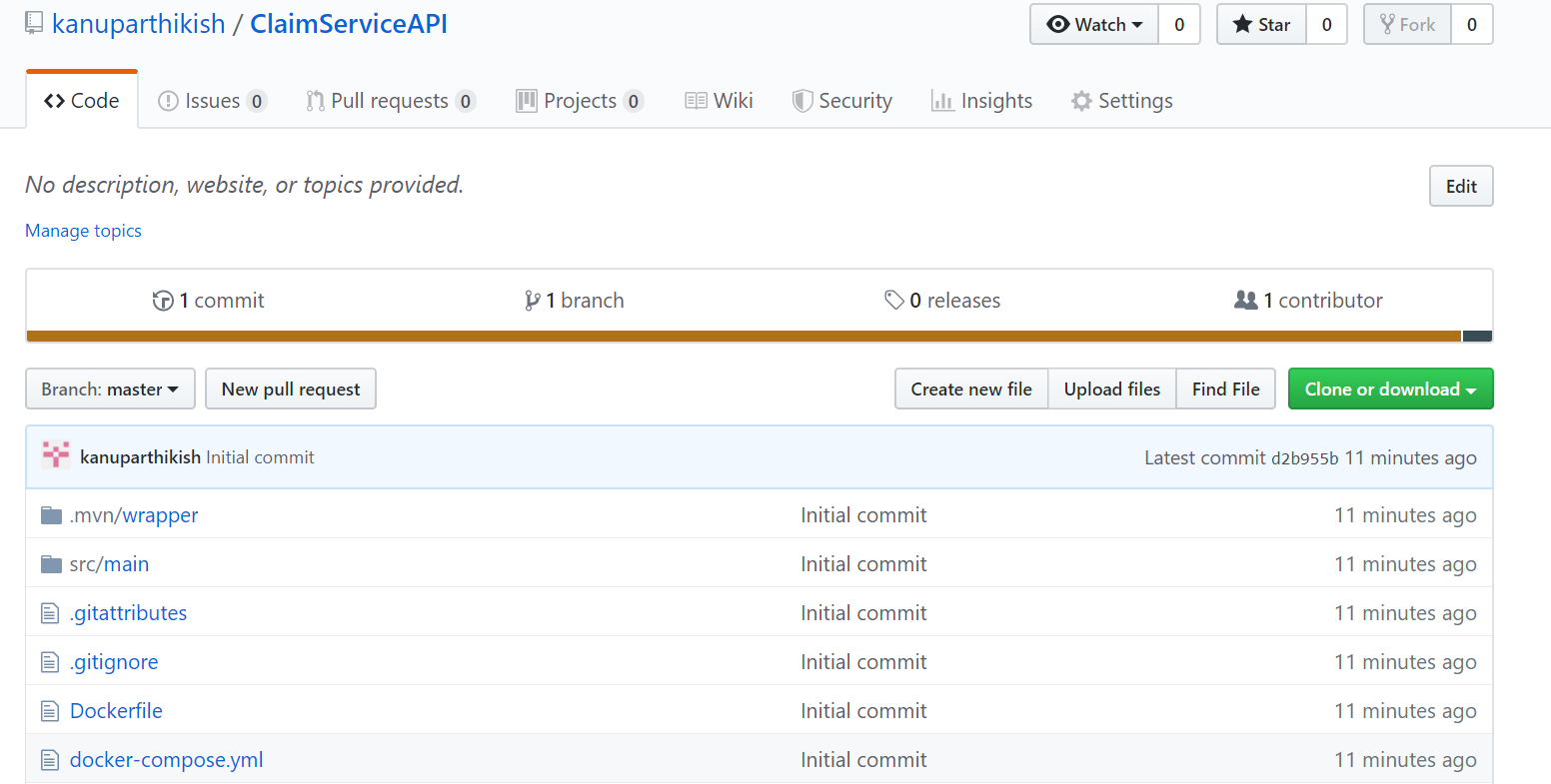
how can we deploy a spring boot REST API and Postgres Sql in docker container by docker-compose.yml

## Tools Required

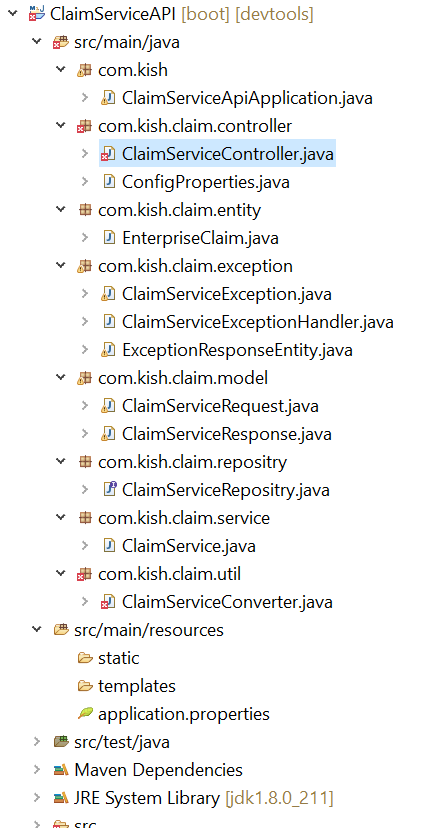
* Maven 3.0+ build tool
* STS or Eclipse as IDE.
* Docker
* Postman

## Complete Maven Project With Code Example is available in Github

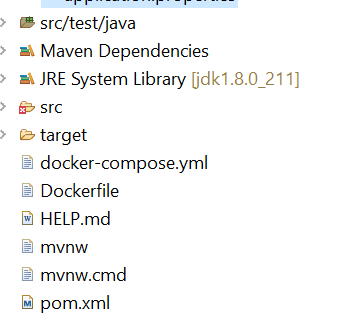
<https://github.com/kanuparthikish/ClaimServiceAPI.git>



2. Create the spring boot project structure as per below by using Spring Starter Project in eclipse IDE



3. Create the Dockerfile and docker-compose.yml file in project root directory for Spring Boot and Postgres application so make up your app in docker-compose.yml so they can be run together in an isolated environment.



4. Dockerfile for spring Boot application

FROM openjdk:8

ADD target/ClaimServiceAPI-0.0.1-SNAPSHOT.jar ClaimServiceAPI-0.0.1-SNAPSHOT.jar

EXPOSE 8080

ENTRYPOINT ["java","-jar","ClaimServiceAPI-0.0.1-SNAPSHOT.jar"]

5. Rest Controller

@RestController

@RequestMapping(value="/ClaimService")

**public** **class** ClaimServiceController {

@Autowired

ClaimService claimService;

@GetMapping

**public** ClaimServiceResponse getClaimInfo(@RequestParam("service-type")String serviceType,@RequestBody ClaimServiceRequest serviceRequest)

{

**if**(serviceType!=**null** && (serviceType.equalsIgnoreCase("DupCheck")||serviceType.equalsIgnoreCase("DrgPricing")))

{

serviceRequest.setServiceType(serviceType);

System.***out***.println("serviceType"+serviceType);

**return** claimService.getClaimDuplicate(serviceRequest);

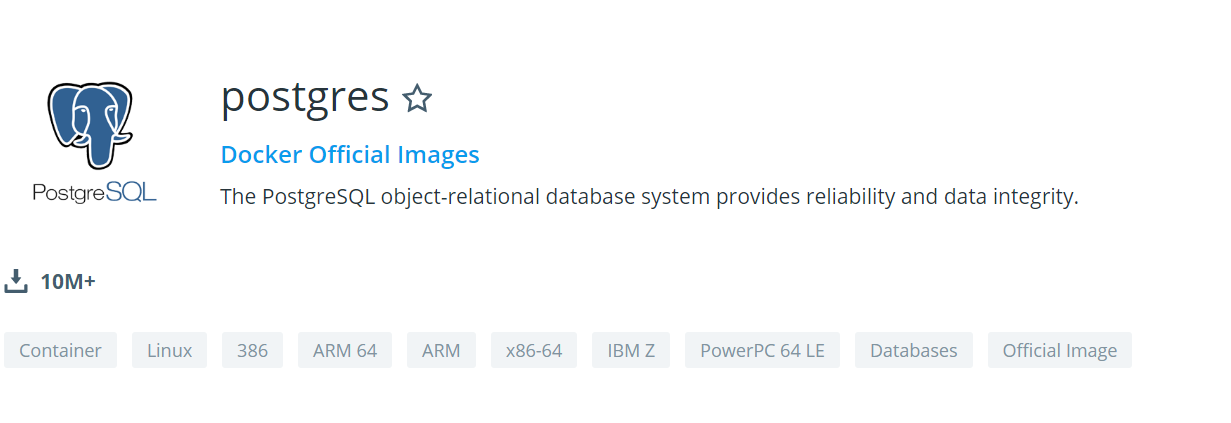
}

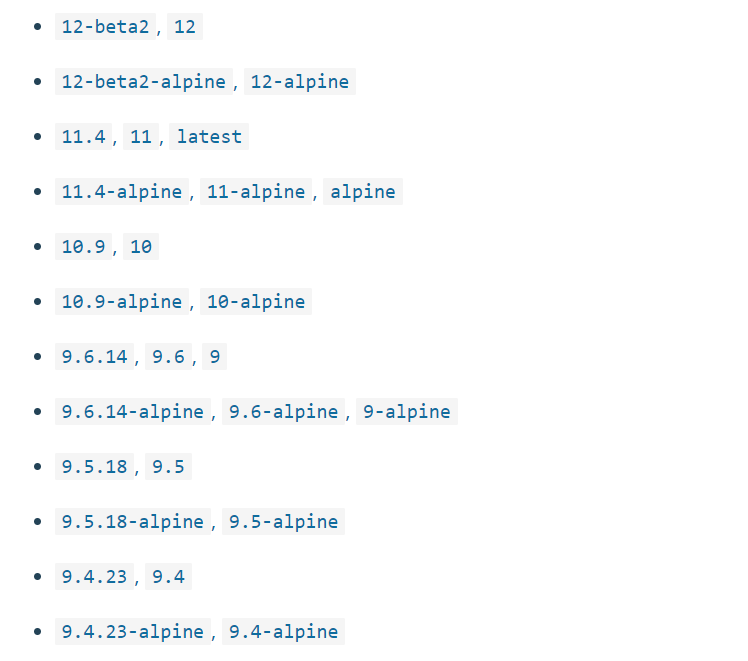
**else**

**throw** **new** ClaimServiceException("Service Not found", "ServiceException");

}

}

5. In docker hub serach for the postgres image and use the image taggerd with latest



5.Define the docker-compose.yml file with ServiceName mypostgres. In my local system postgres runing on 5432 so due to this I want expose docker image with postgres on 5430 on my local machine.Postgres service runing port number 5432 on the docker container.

Docker-compose.yml file

version: '3'

services:

mypostgres:

image: 'postgres:latest'

ports:

- "5430:5432"

environment:

POSTGRES\_PASSWORD: postgres

POSTGRES\_USER: postgres

POSTGRES\_DB: kishore

claimserviceapi-container:

build: .

ports:

- "8080:8080"

depends\_on:

- mypostgres

6. The Spring Boot Application service claimserviceapi-container Image was build through the Dockerfile and exposed the service on port number 8080

7. Define the Data Base configuration based on the postgres service defined in docker-compose.yml file

Host name:service name(mypostgres)

application.properties

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

spring.datasource.url=jdbc:postgresql://mypostgres:5432/kishore

spring.datasource.username=postgres

spring.datasource.password=postgres

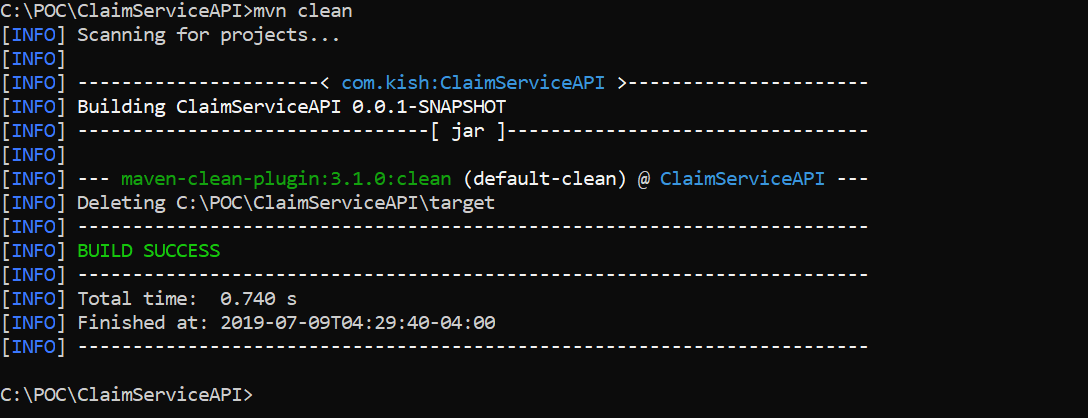
spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.show-sql=true

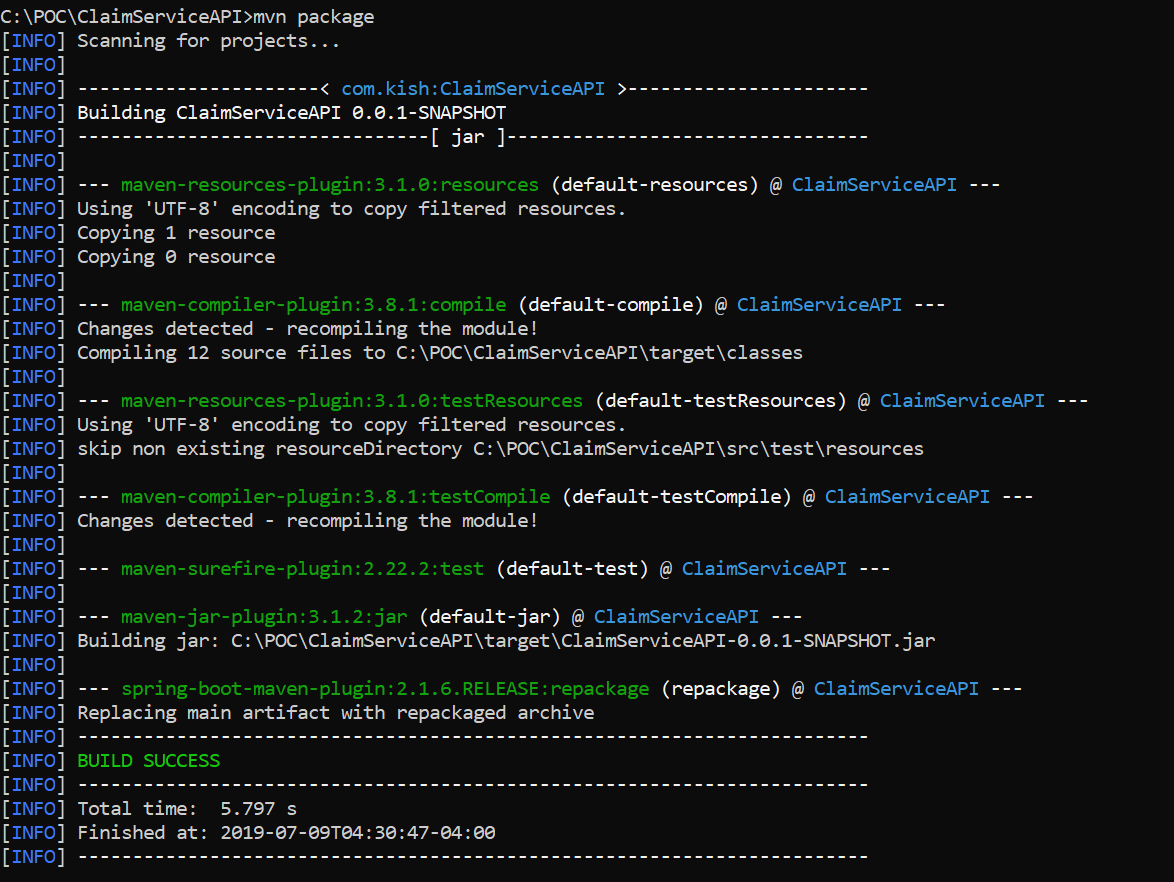
spring.jpa.hibernate.ddl-auto=create

8. create the build package with maven

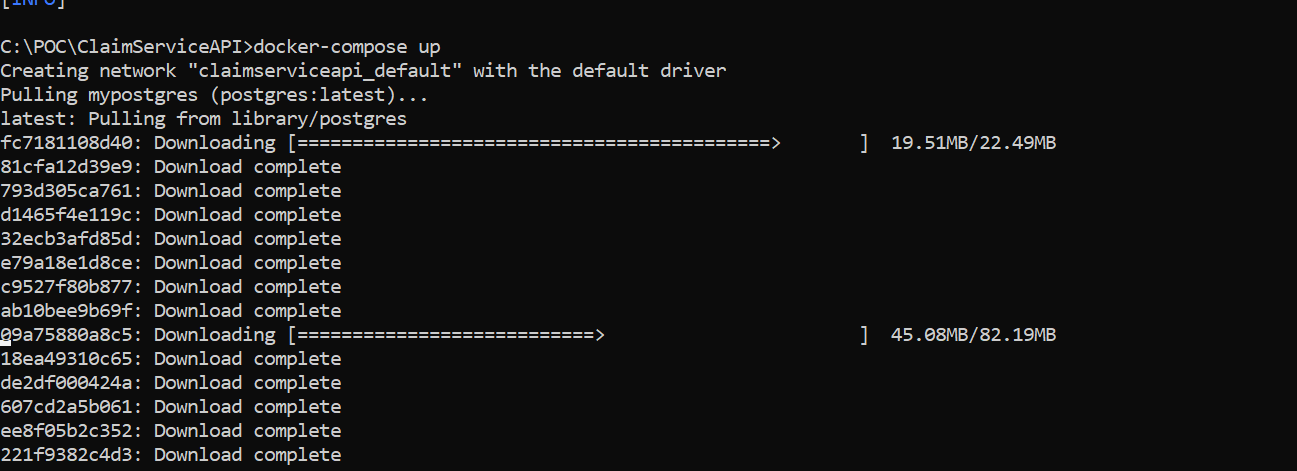
mvn clean

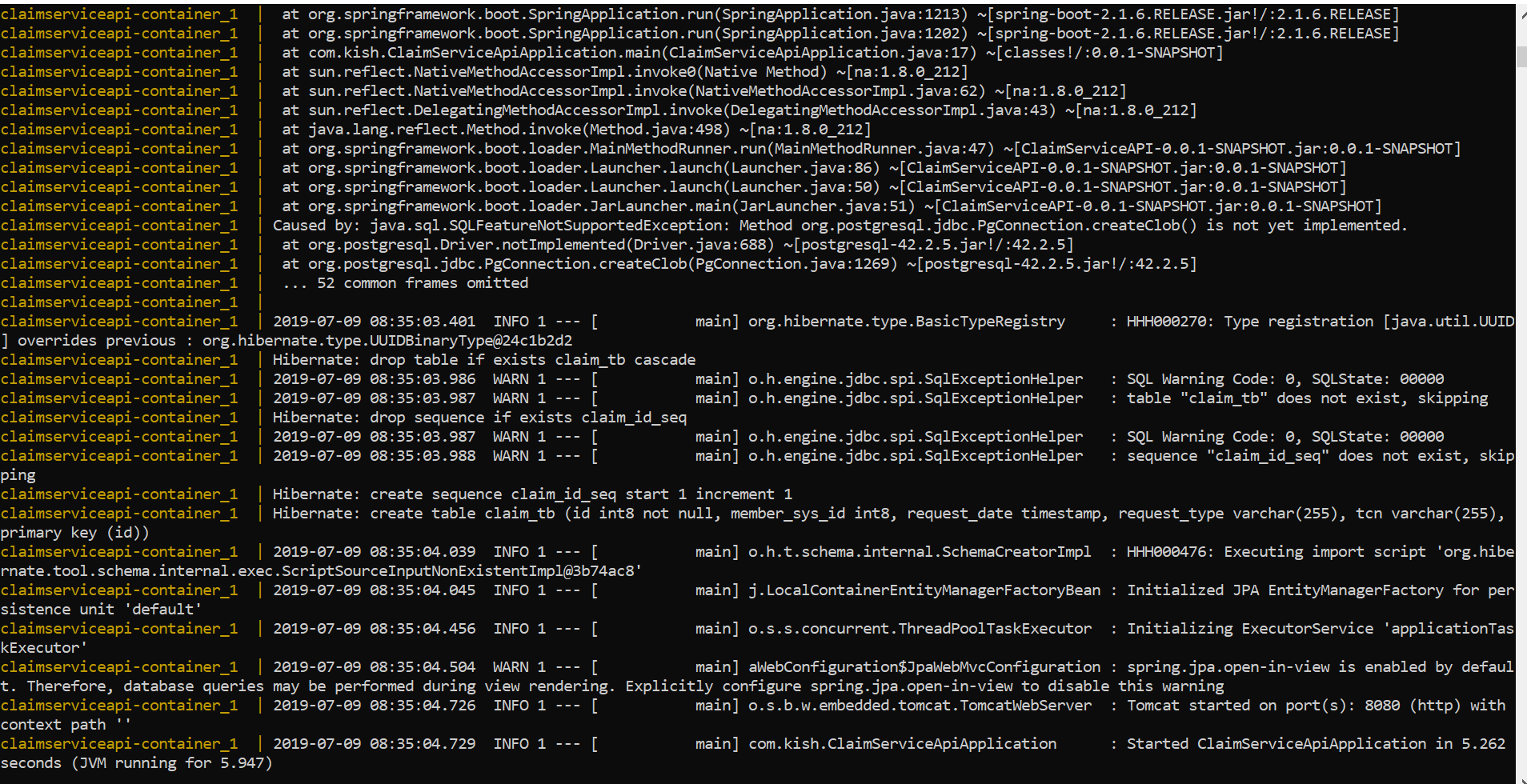


mvn package

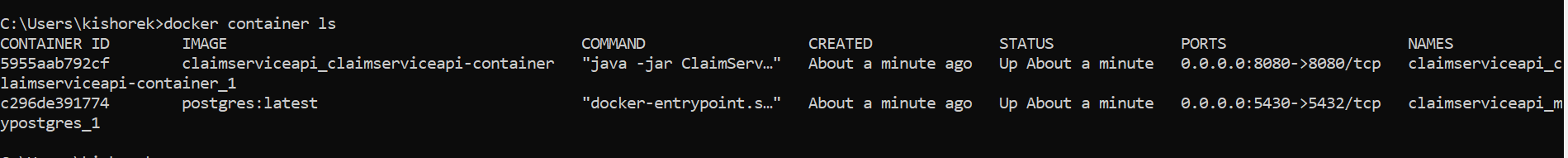


9. Run the docker-compose up command. It will take couple of minutes to build the Images





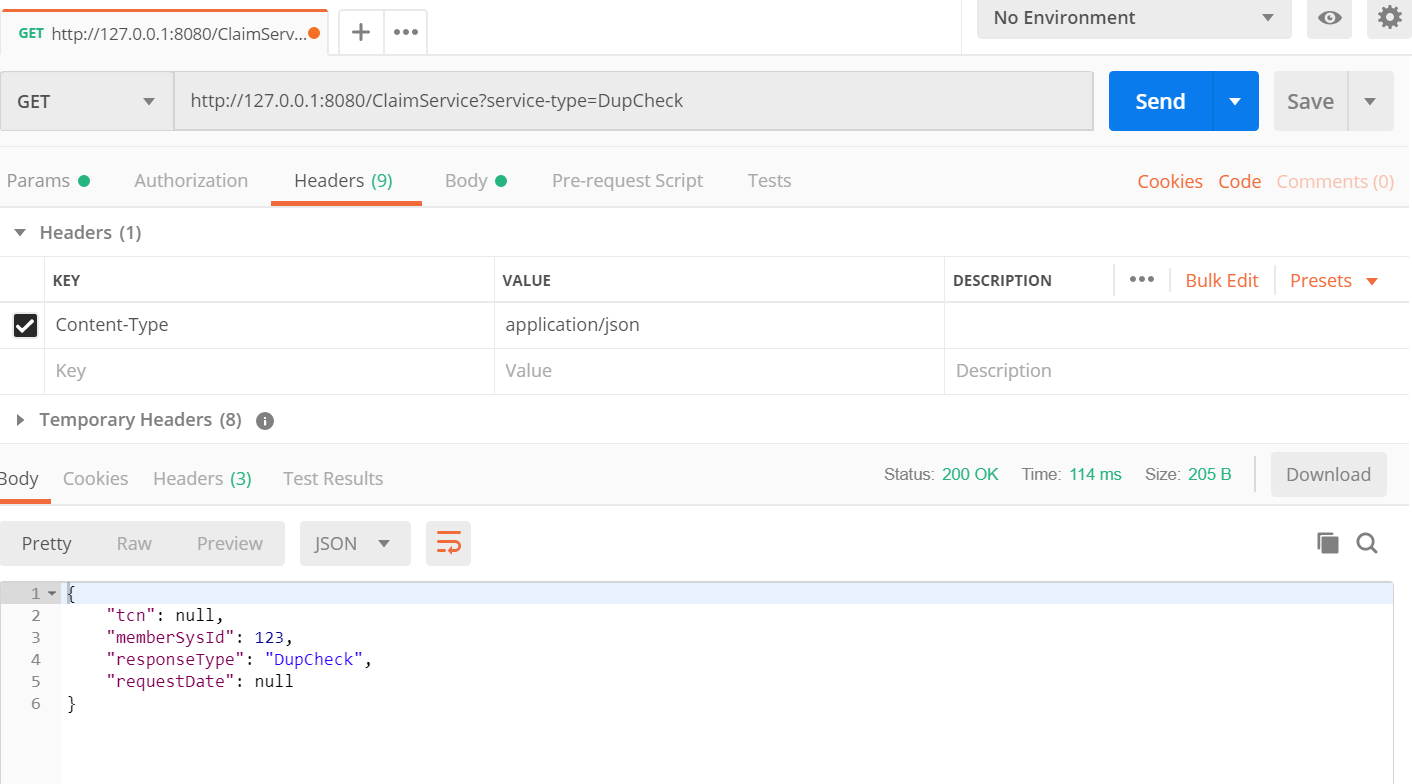
10. the docker containers are up and running



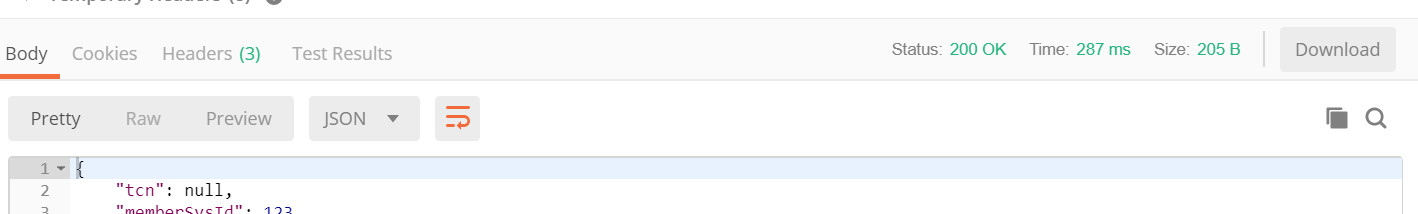
11. in windows 10 we have to use 127.0.0.1 instead of local host to connect service are running on docker container.

<http://127.0.0.1:8080/ClaimService?service-type=DupCheck>

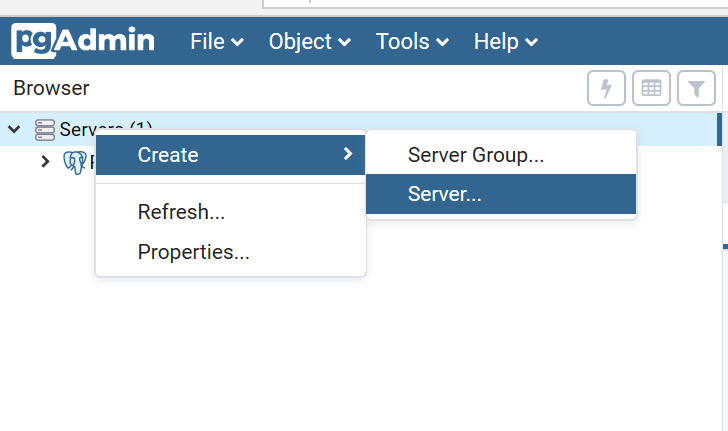
Request:

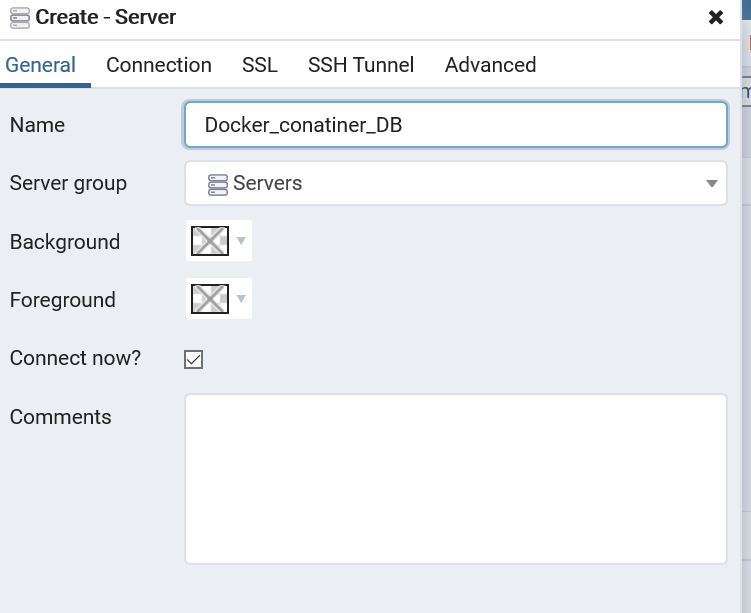


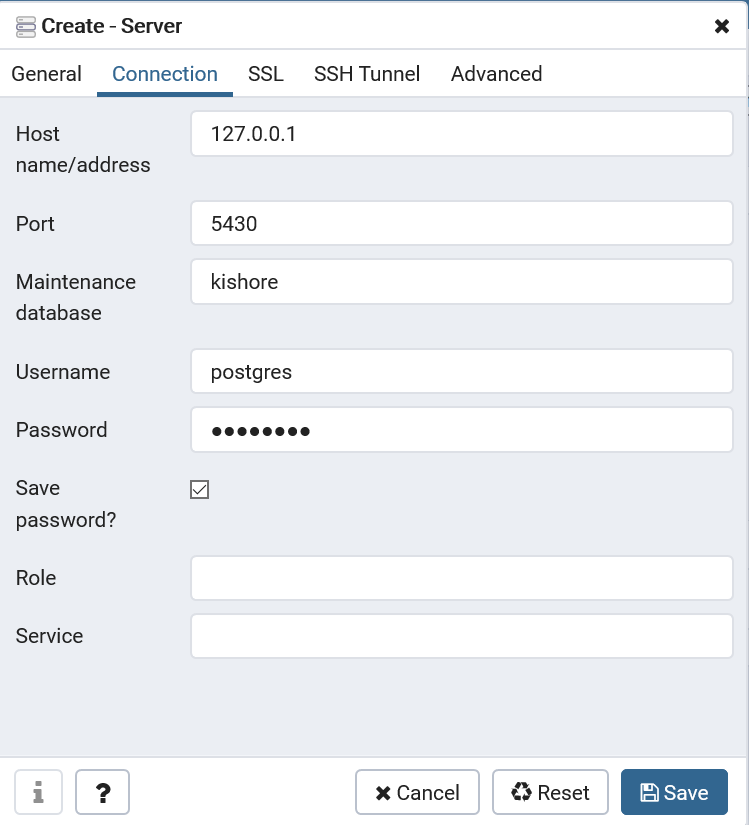
Response:



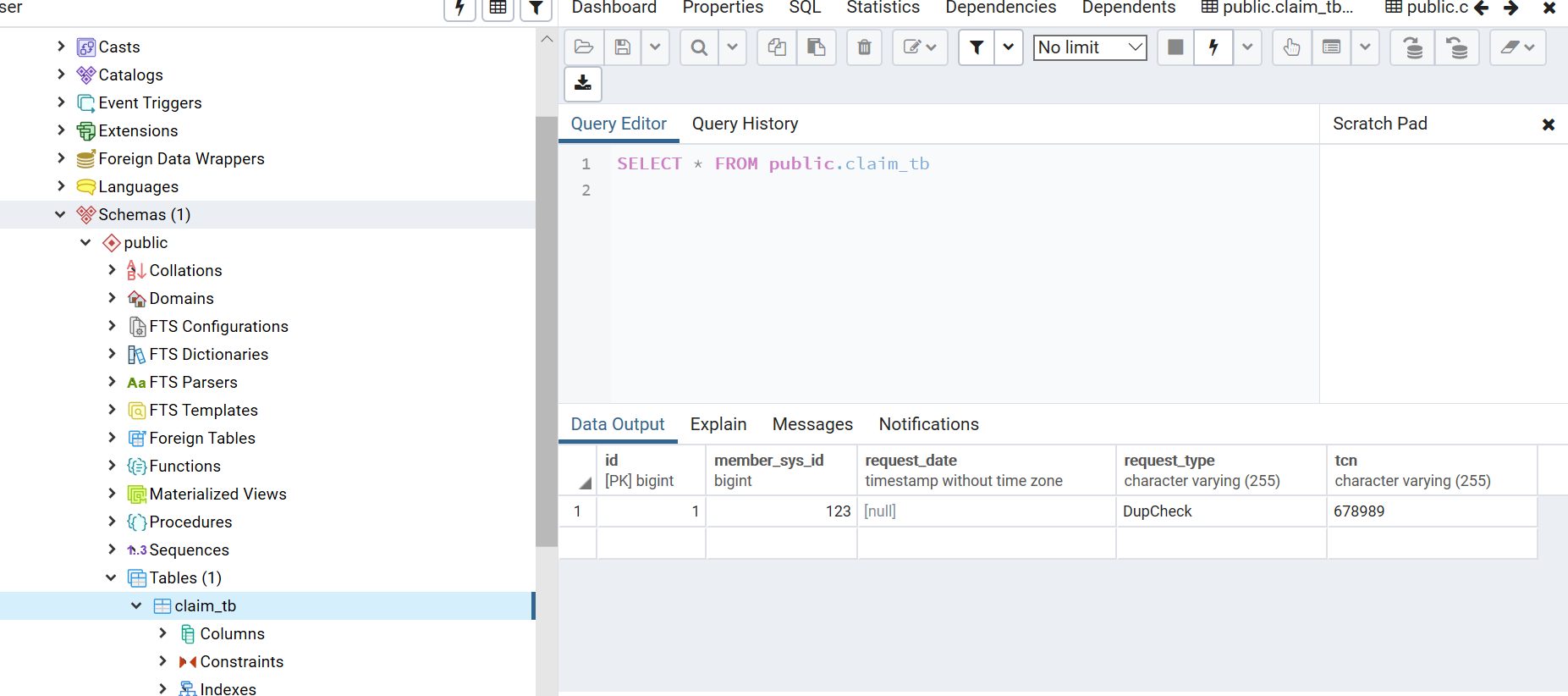
11. By using pgAdmin we can connect to the Data base which is running on docker container.







12. Record save sucessfully to the table.



Second Request:

