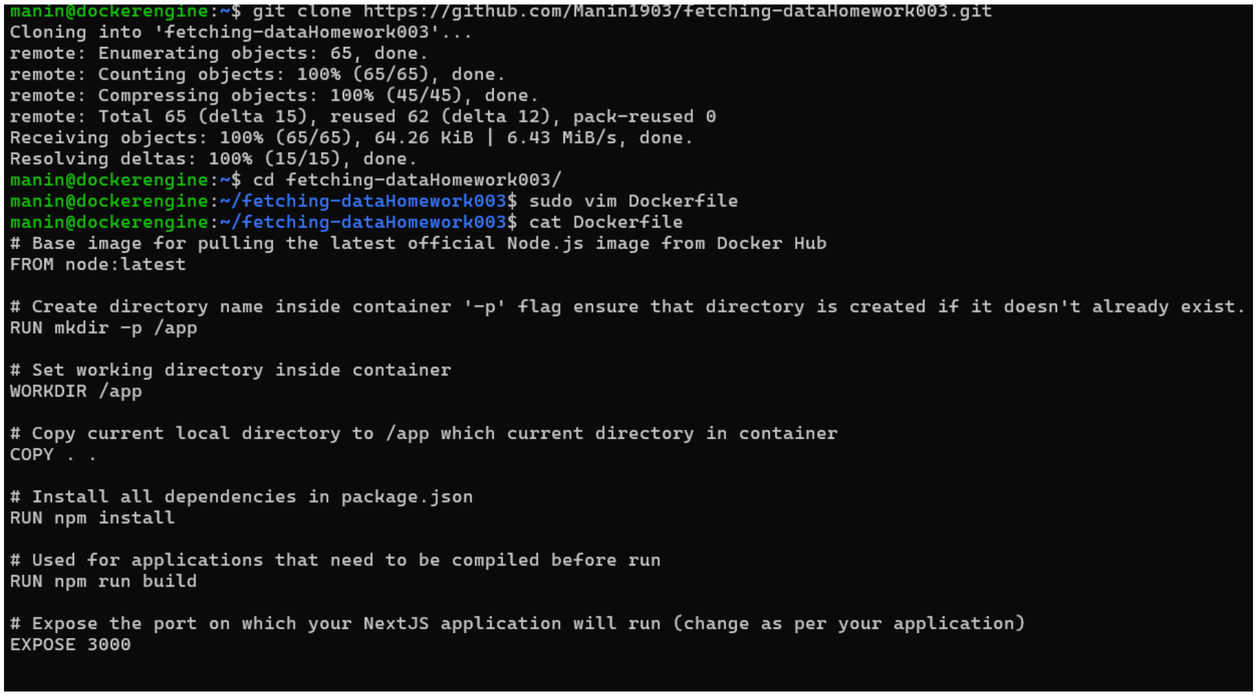
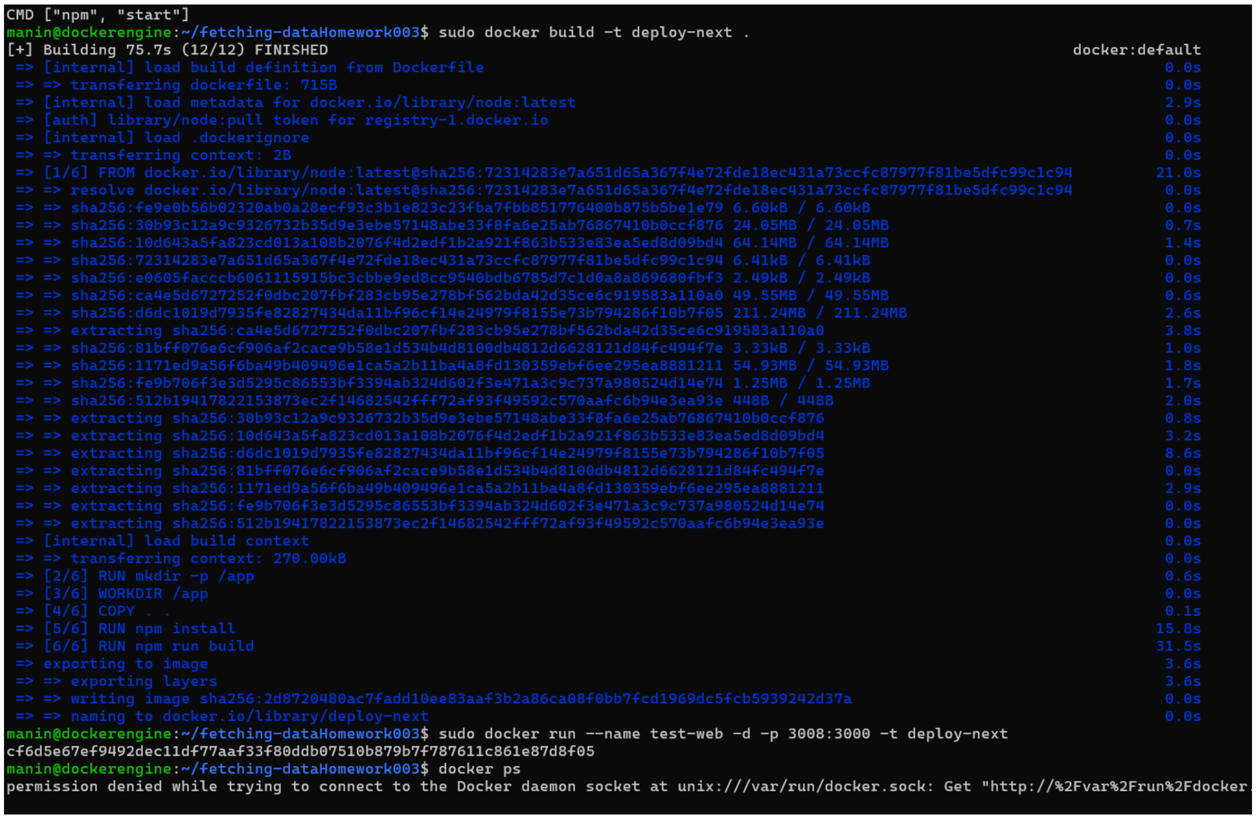
Homework(deploy)

- Docker deploy spring boot



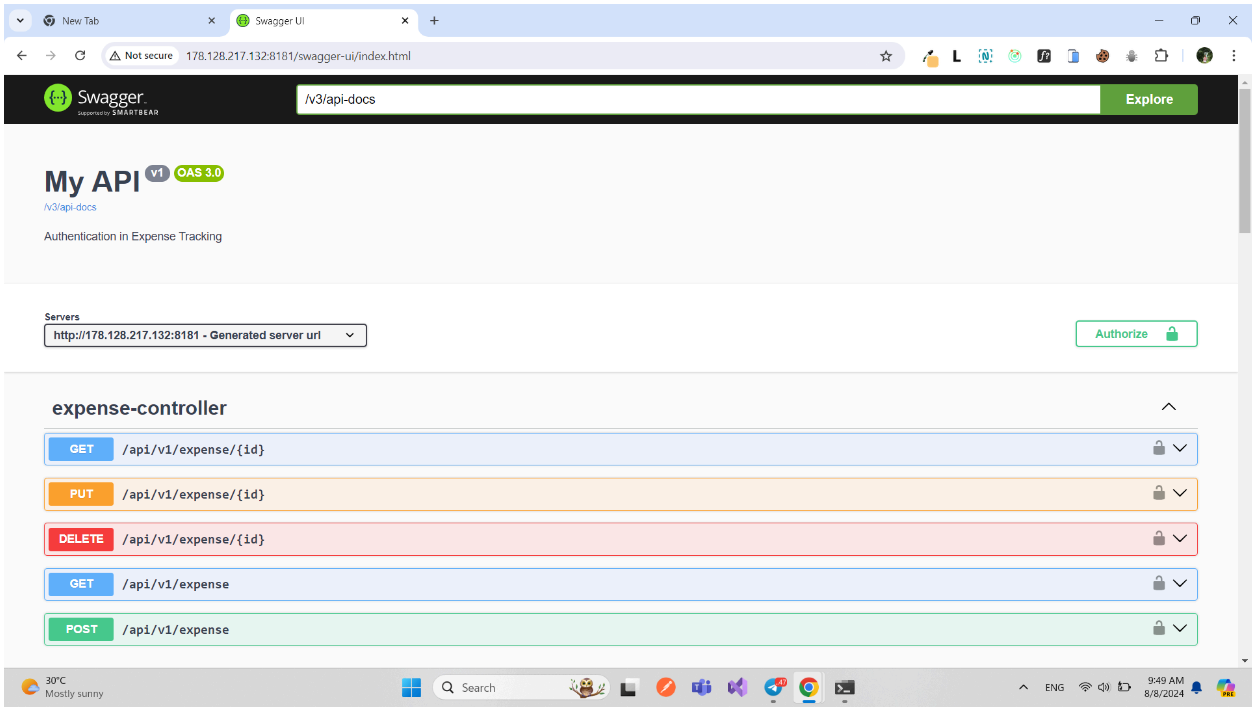
```
manin@dockerengine:~$ git clone https://github.com/Manin1903/fetching-dataHomework003.git
Cloning into 'fetching-dataHomework003'...
remote: Enumerating objects: 65, done.
remote: Counting objects: 100% (65/65), done.
remote: Compressing objects: 100% (45/45), done.
remote: Total 65 (delta 15), reused 62 (delta 12), pack-reused 0
Receiving objects: 100% (65/65), 64.26 KiB | 6.43 MiB/s, done.
Resolving deltas: 100% (15/15), done.
manin@dockerengine:~$ cd fetching-dataHomework003/
manin@dockerengine:~/fetching-dataHomework003$ sudo vim Dockerfile
manin@dockerengine:~/fetching-dataHomework003$ cat Dockerfile
# Base image for pulling the latest official Node.js image from Docker Hub
FROM node:latest

# Create directory name inside container '-p' flag ensure that directory is created if it doesn't already exist.
RUN mkdir -p /app

# Set working directory inside container
WORKDIR /app

# Copy current local directory to /app which current directory in container
COPY . .

# Install all dependencies in package.json
RUN npm install

# Used for applications that need to be compiled before run
RUN npm run build

# Expose the port on which your NextJS application will run (change as per your application)
EXPOSE 3000
```



```
CMD ["npm", "start"]
manin@dockerengine:~/fetching-dataHomework003$ sudo docker build -t deploy-next .
[+] Building 75.7s (12/12) FINISHED                                                                            docker:default
 => [internal] load build definition from Dockerfile                                                                     0.0s
 => => transferring dockerfile: 715B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/node:latest                                                           2.9s
 => [auth] library/node:pull token for registry-1.docker.io                                                              0.0s
 => [internal] load .dockerignore                                                                                        0.0s
 => => transferring context: 2B                                                                                          0.0s
 => [1/6] FROM docker.io/library/node:latest@sha256:72314283e7a651d65a367f4e72fde18ec431a73ccfc87977f81be5dfc99c1c94    21.0s
 => => resolve docker.io/library/node:latest@sha256:72314283e7a651d65a367f4e72fde18ec431a73ccfc87977f81be5dfc99c1c94     0.0s
 => => sha256:fe9e0b56b02320ab0a28ecf93c3b1e823c23fba7fbb851776400b875b5be1e79 6.60kB / 6.60kB                           0.0s
 => => sha256:30b93c12a9c9326732b35d9e3ebe57148abe33f8fa6e25ab76867410b0ccf876 24.05MB / 24.05MB                         0.7s
 => => sha256:10d643a5fa823cd013a108b2076f4d2edf1b2a921f863b533e83ea5ed8d09bd4 64.14MB / 64.14MB                         1.4s
 => => sha256:72314283e7a651d65a367f4e72fde18ec431a73ccfc87977f81be5dfc99c1c94 6.41kB / 6.41kB                           0.0s
 => => sha256:e0605faccb6061115915bc3cbbe9ed8cc9540bdb6785d7c1d0a8a869680fbf3 2.49kB / 2.49kB                            0.0s
 => => sha256:ca4e5d6727252f0dbc207fbf283cb95e278bf562bda42d35ce6c919583a110a0 49.55MB / 49.55MB                         0.6s
 => => sha256:d6dc1019d7935fe82827434da11bf96cf14e24979f8155e73b794286f10b7f05 211.24MB / 211.24MB                       2.6s
 => => extracting sha256:ca4e5d6727252f0dbc207fbf283cb95e278bf562bda42d35ce6c919583a110a0                                3.8s
 => => sha256:81bff076e6cf906af2cace9b58e1d534b4d8100db4812d6628121d84fc494f7e 3.33kB / 3.33kB                           1.0s
 => => sha256:1171ed9a56f6ba49b409496e1ca5a2b11ba4a8fd130359ebf6ee295ea8881211 54.93MB / 54.93MB                         1.8s
 => => sha256:fe9b706f3e3d5295c86553bf3394ab324d602f3e471a3c9c737a980524d14e74 1.25MB / 1.25MB                           1.7s
 => => sha256:512b19417822153873ec2f14682542fff72af93f49592c570aafc6b94e3ea93e 448B / 448B                               2.0s
 => => extracting sha256:30b93c12a9c9326732b35d9e3ebe57148abe33f8fa6e25ab76867410b0ccf876                                0.8s
 => => extracting sha256:10d643a5fa823cd013a108b2076f4d2edf1b2a921f863b533e83ea5ed8d09bd4                                3.2s
 => => extracting sha256:d6dc1019d7935fe82827434da11bf96cf14e24979f8155e73b794286f10b7f05                                8.6s
 => => extracting sha256:81bff076e6cf906af2cace9b58e1d534b4d8100db4812d6628121d84fc494f7e                                0.0s
 => => extracting sha256:1171ed9a56f6ba49b409496e1ca5a2b11ba4a8fd130359ebf6ee295ea8881211                                2.9s
 => => extracting sha256:fe9b706f3e3d5295c86553bf3394ab324d602f3e471a3c9c737a980524d14e74                                0.0s
 => => extracting sha256:512b19417822153873ec2f14682542fff72af93f49592c570aafc6b94e3ea93e                                0.0s
 => [internal] load build context                                                                                        0.0s
 => => transferring context: 270.00kB                                                                                    0.0s
 => [2/6] RUN mkdir -p /app                                                                                              0.6s
 => [3/6] WORKDIR /app                                                                                                   0.0s
 => [4/6] COPY . .                                                                                                       0.1s
 => [5/6] RUN npm install                                                                                               15.8s
 => [6/6] RUN npm run build                                                                                             31.5s
 => exporting to image                                                                                                   3.6s
 => => exporting layers                                                                                                  3.6s
 => => writing image sha256:2d8720480ac7fadd10ee83aaf3b2a86ca08f0bb7fcd1969dc5fcb5939242d37a                             0.0s
 => => naming to docker.io/library/deploy-next                                                                           0.0s
manin@dockerengine:~/fetching-dataHomework003$ sudo docker run --name test-web -d -p 3008:3000 -t deploy-next
cf6d5e67ef9492dec11df77aaf33f80ddb07510b879b7f787611c861e87d8f05
manin@dockerengine:~/fetching-dataHomework003$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.
```

Result:

Link of Spring Boot API:http://178.128.217.132:8181/swagger-ui/index.html



My API v1 OAS 3.0

/v3/api-docs

Authentication in Expense Tracking

Servers

http://178.128.217.132:8181 - Generated server url

Authorize

expense-controller

GET /api/v1/expense/{id}

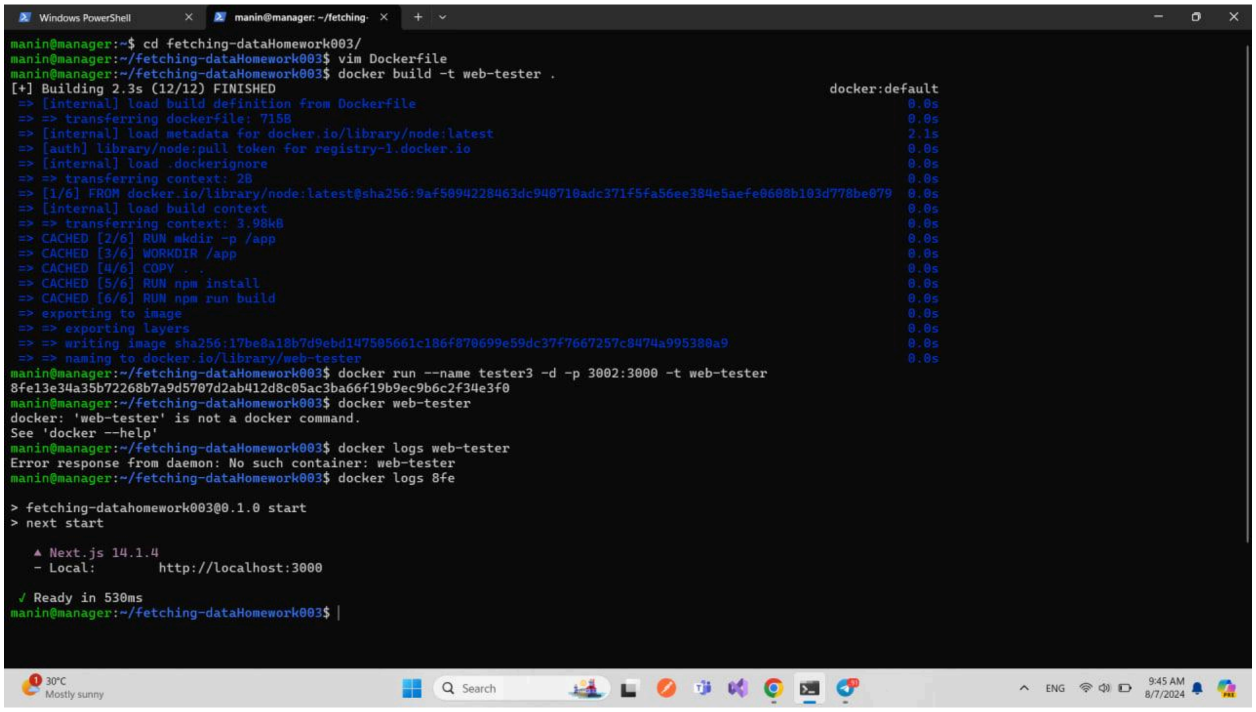
PUT /api/v1/expense/{id}

DELETE /api/v1/expense/{id}

GET /api/v1/expense

POST /api/v1/expense

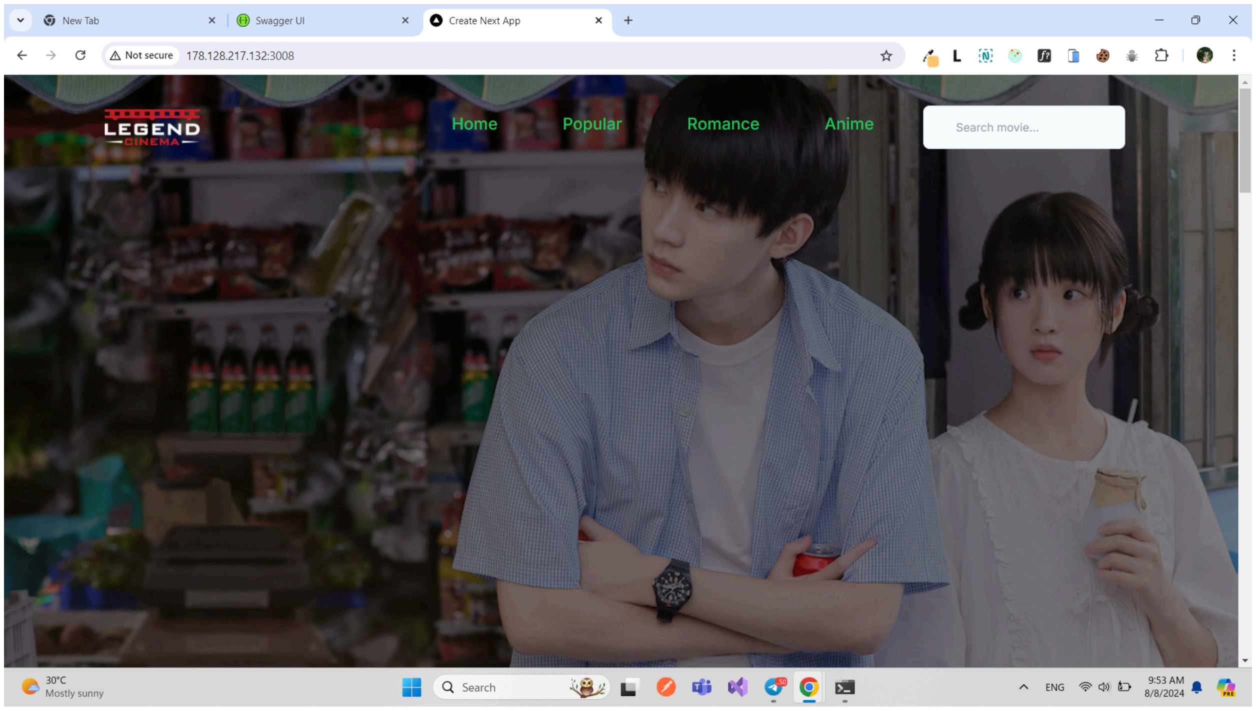
- Docker deploy next-js



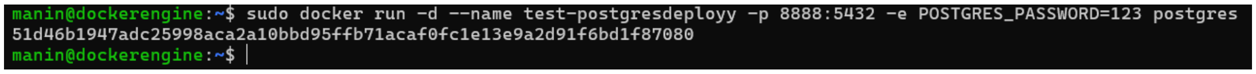
```
manin@manager:~$ cd fetching-dataHomework003/
manin@manager:~/fetching-dataHomework003$ vim Dockerfile
manin@manager:~/fetching-dataHomework003$ docker build -t web-tester .
[+] Building 2.3s (12/12) FINISHED                                                                 docker:default
 => [internal] load build definition from Dockerfile                                                         0.0s
 => => transferring dockerfile: 715B                                                                         0.0s
 => [internal] load metadata for docker.io/library/node:latest                                               2.1s
 => [auth] library/node:pull token for registry-1.docker.io                                                  0.0s
 => [internal] load .dockerignore                                                                            0.0s
 => => transferring context: 2B                                                                              0.0s
 => [1/6] FROM docker.io/library/node:latest@sha256:9af5094228463dc940710adc371f5fa56ee384e5aefe0608b103d778be079 0.0s
 => [internal] load build context                                                                            0.0s
 => => transferring context: 3.98kB                                                                          0.0s
 => CACHED [2/6] RUN mkdir -p /app                                                                           0.0s
 => CACHED [3/6] WORKDIR /app                                                                                0.0s
 => CACHED [4/6] COPY . .                                                                                    0.0s
 => CACHED [5/6] RUN npm install                                                                             0.0s
 => CACHED [6/6] RUN npm run build                                                                           0.0s
 => exporting to image                                                                                       0.0s
 => => exporting layers                                                                                      0.0s
 => => writing image sha256:17be8a18b7d9ebd147505661c186f870699e59dc37f7667257c8474a995380a9                 0.0s
 => => naming to docker.io/library/web-tester                                                                0.0s
manin@manager:~/fetching-dataHomework003$ docker run --name tester3 -d -p 3002:3000 -t web-tester
8fe13e34a35b72268b7a9d5707d2ab412d8c05ac3ba66f19b9ec9b6c2f34e3f0
manin@manager:~/fetching-dataHomework003$ docker web-tester
docker: 'web-tester' is not a docker command.
See 'docker --help'
manin@manager:~/fetching-dataHomework003$ docker logs web-tester
Error response from daemon: No such container: web-tester
manin@manager:~/fetching-dataHomework003$ docker logs 8fe

> fetching-datahomework003@0.1.0 start
> next start

   ▲ Next.js 14.1.4
   - Local:        http://localhost:3000

 ✓ Ready in 530ms
manin@manager:~/fetching-dataHomework003$ |
```

Result:

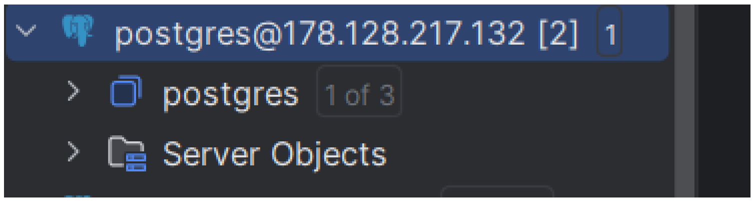
Link of Next-js:http://178.128.217.132:3009



- Deploy postgres



```
manin@dockerengine:~$ sudo docker run -d --name test-postgresdeployy -p 8888:5432 -e POSTGRES_PASSWORD=123 postgres
51d46b1947adc25998aca2a10bbd95ffb71acaf0fc1e13e9a2d91f6bd1f87080
manin@dockerengine:~$ |
```

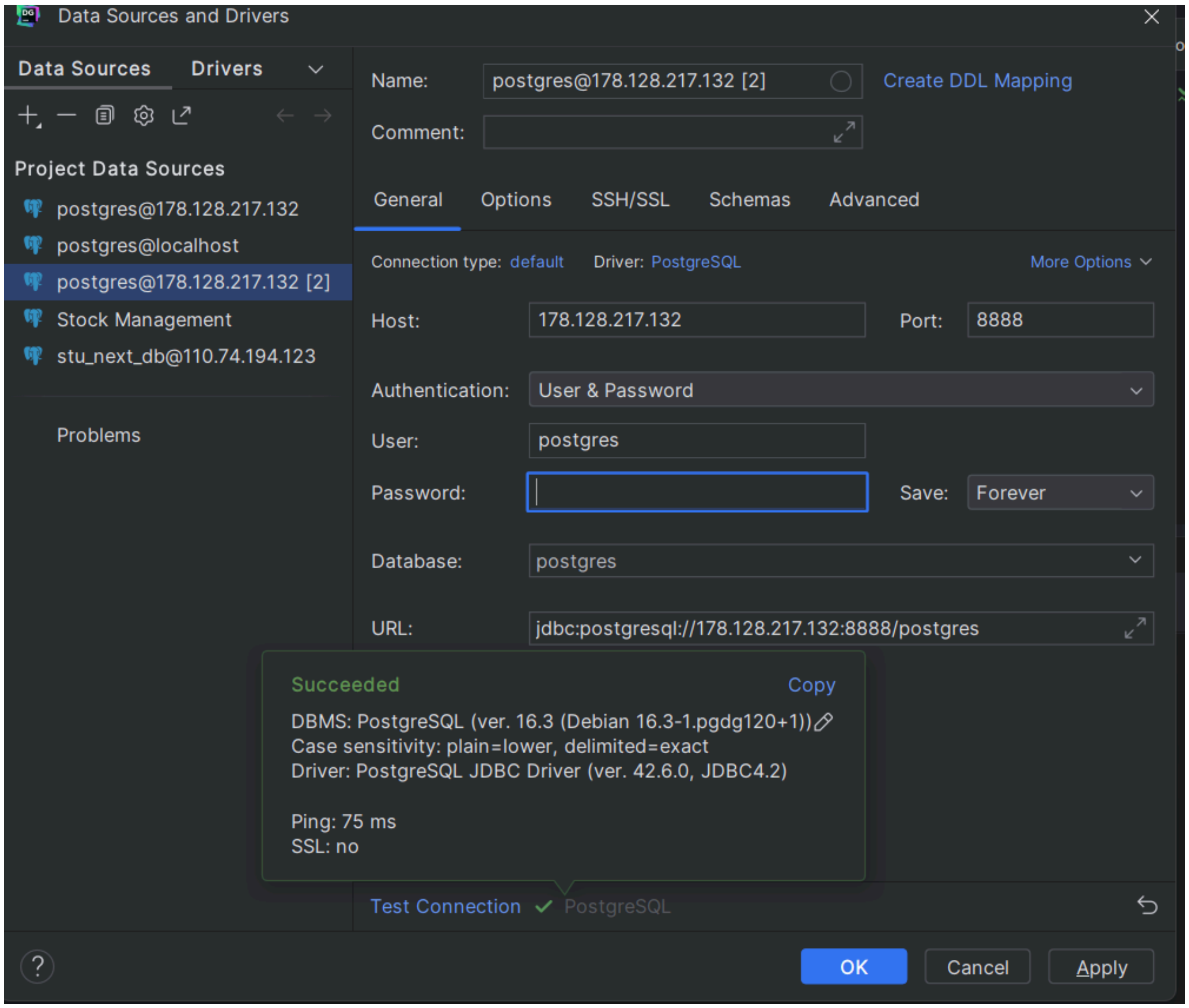


Host ip : 178.128.217.132

Port : 8888

User : postgres

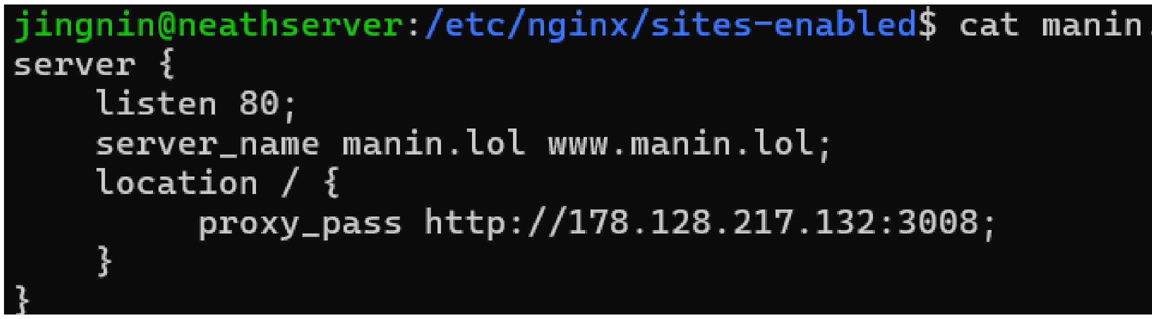
Password: 123



Name: postgres@178.128.217.132 [2]
Comment:
Connection type: default Driver: PostgreSQL
Host: 178.128.217.132 Port: 8888
Authentication: User & Password
User: postgres
Password: Save: Forever
Database: postgres
URL: jdbc:postgresql://178.128.217.132:8888/postgres

```
Succeeded                                                   Copy
DBMS: PostgreSQL (ver. 16.3 (Debian 16.3-1.pgdg120+1))
Case sensitivity: plain=lower, delimited=exact
Driver: PostgreSQL JDBC Driver (ver. 42.6.0, JDBC4.2)

Ping: 75 ms
SSL: no
```

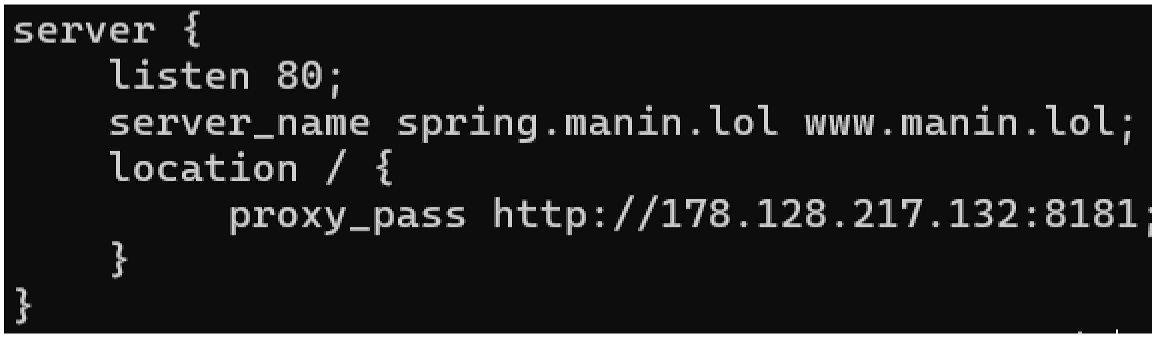
- Config Domain for next-js
    - First step install nginx [sudo apt install nginx]
    - Check status active or not [sudo systemctl status anginx]
    - After seen active we change directory to [cd /etc/nginx/sites-available]
    - Create file for config[sudo vim manin.lol]
    - Insert this config to this file

- 

```
jingnin@neathserver:/etc/nginx/sites-enabled$ cat manin.
server {
    listen 80;
    server_name manin.lol www.manin.lol;
    location / {
         proxy_pass http://178.128.217.132:3008;
    }
}
```

    - Write this command [sudo ln -s /etc/nginx/sites-available/manin.lol /etc/nginx/sites-enabled/]
    - Than we use this command to start [systemctl restart nginx]

Result link: http://manin.lol/

- Config Domain for Spring Boot
    - Insert more config
    - 

```
server {
    listen 80;
    server_name spring.manin.lol www.manin.lol;
    location / {
         proxy_pass http://178.128.217.132:8181;
    }
}
```

    - Than run this command [systemctl restart nginx]
    - Get result link here: Swagger UI (manin.lol)http://spring.manin.lol/swagger-ui/index.html
- Config https next-js
    - First install certbot
        - sudo apt install certbot python3-certbot-nginx
    - Install ufw
        - sudo apt update && sudo apt install ufw

    - Enable ufw
        - sudo ufw enable
    - Check ufw status
        - sudo ufw status
    - Add rules
        - sudo ufw allow   ‘Nginx Full’
        - sudo ufw delete allow   ‘Nginx HTTP’   # do not allow to access http
        - sudo ufw allow   ‘OpenSSH’
    - Check ufw again
        - sudo ufw status
    - Obtaining an SSL Certificate
        - sudo certbot --nginx -d manin.lol -d http://manin.lol
    - Now is successful here is link : https://manin.lol/
- Config https spring boot
    - Obtaining an SSL Certificate
        - sudo certbot --nginx -d spring.manin.lol
    - Now is successful here is link : [Swagger UI (manin.lol)https://spring.manin.lol/swagger-ui/index.html](https://spring.manin.lol/swagger-ui/index.html)