Lecture 08

SOFTWARE CONFIGURATION MANAGEMENT

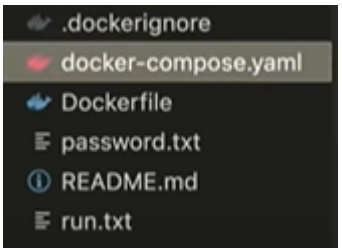
Docker Compass

Running multiple components at once.

Docker ps : Shows the running component.Intially there are no components for run.

If we need to run one we can use “docker run”.

1. First we need to identify , what are the docker component need to run in same time.
   So , create a file named
   “docker-compose. yaml”



When we develop application we need database and tools to interact with and manage databases effectively.

PostgreSQL (Postgres): PostgreSQL is a popular relational database management system (RDBMS) commonly used in various applications, especially those requiring robust data management.

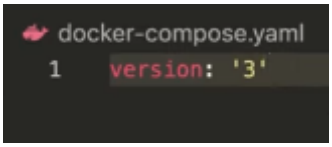
PGAdmin: PGAdmin as a web-based database management tool that provides a graphical interface for interacting with PostgreSQL databases. This tool simplifies database administration tasks and allows users to execute queries, manage data, and perform other database-related operations through a user-friendly interface.

Docker Compose Configuration

By defining services for both PostgreSQL and PGAdmin in a Docker Compose file, you can arrange the deployment and configuration of these services with a single command.

Steps

Firstly , we can define the version of the docker compose.



Next, we want to define the services. We mention the docker images that we want to run.

For this, first, we want to define the service name.

And then we want to provide the image we are going to run.

Then we provide the container name (given by us)

Then provide environment variable(A Docker environment variable is a variable that is set within a Docker container to configure various aspects of its behavior or to pass information to the applications running inside the container.)

```yaml
version: '3'
services:
  postgresql:
    image: postgres:15.4
    container_name: test_postgresql
    environment:
      - POSTGRES_DB: app_db
      - POSTGRES_USER: pguser
      - POSTGRES_PASSWORD: pguser_password
```

Then we want to run a command:

```
docker compose up
```

Now we going to Add service for PGAdmin.For this we are following same steps.

```yaml
pgadmin:
  image: dpage/pgadmin4:7.7
  container_name: test_pgadmin
  environment:
    PGADMIN_DEFAULT_EMAIL: admin@gmail.com
    PGADMIN_DEFAULT_PASSWORD: pguser_password2
  ports:
    - 8080:80
```

Ports are defined to map the network ports of the container to the host machine. In this case, the port 8080 is being mapped from the container to the host.

By mapping ports, services running inside the container become accessible from the host machine.

In the configuration you provided, **PGAdmin** service is being exposed on port 8080 of the host machine.

Defining ports allows external clients or applications to connect to the service running inside the container. In this scenario, users can access the **PGAdmin** web interface.

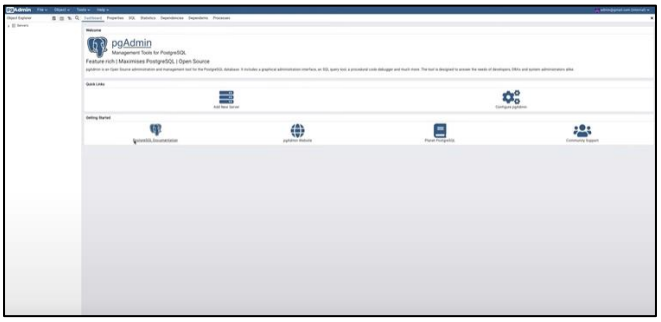
Then we want to run:

```
docker compose up
```

After that we can use PGAdmin web interface by using http://localhost:8080

Then we use these thing to login PGAdmin:

```
PGADMIN_DEFAULT_EMAIL: admin@gmail.com
PGADMIN_DEFAULT_PASSWORD: pguser_password2
```



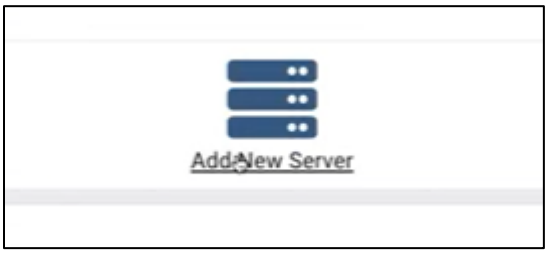
Then we are doing some changes in postgresql:

```
services:
  postgresql:
    image: postgres:15.4
    container_name: test_postgresql
    environment:
      POSTGRES_DB: app_db
      POSTGRES_USER: pguser
      POSTGRES_PASSWORD: pguser_password
    ports:
      - 5432:5432
```

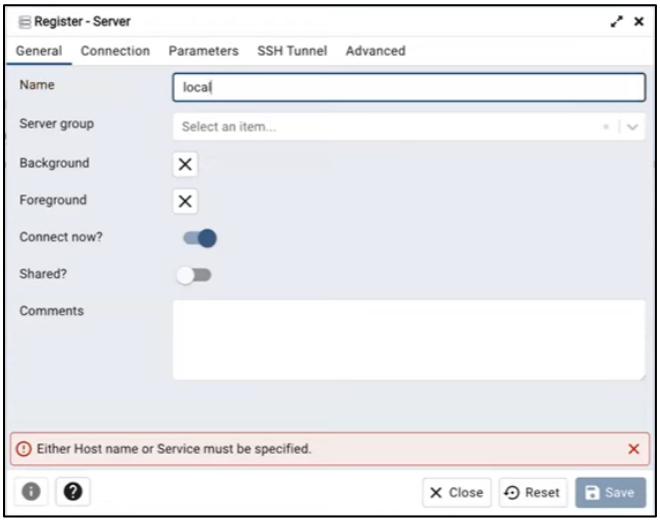
Then we need to Terminal the running container.Then we nee to run with these changes:

```
docker compose up
```

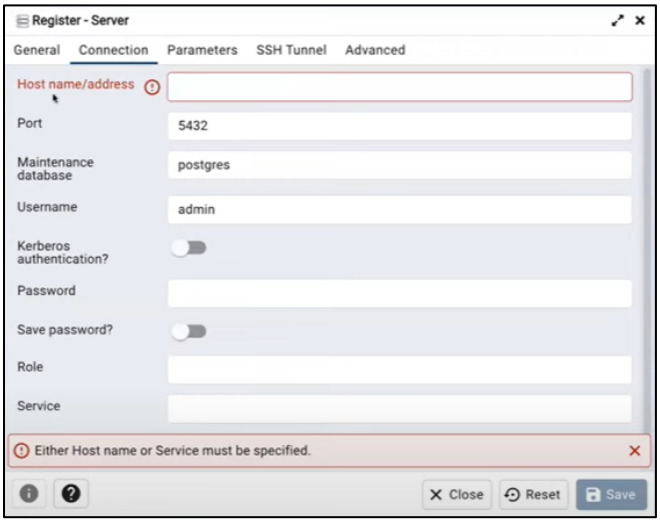
Now we need to connect the PGAdmin and Postgresql together.



So, go Add new Server.



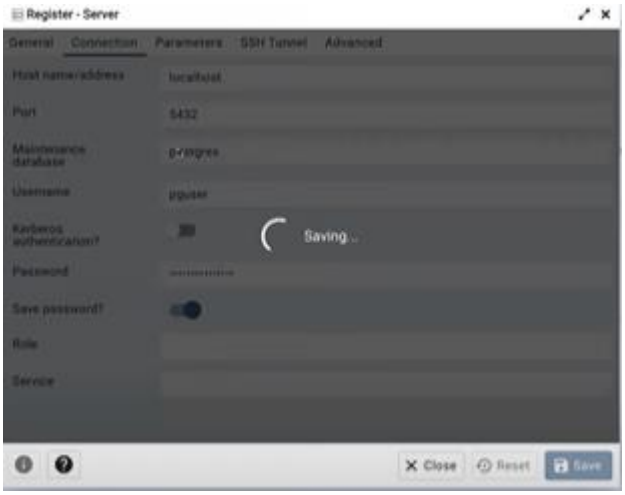
In general , Add name as “Local_Connection”



Host name is “Local Host” and add these username and password:

```
- POSTGRES_USER: pguser
- POSTGRES_PASSWORD: pguser_password
```

Then save the all things.



Some time it is not saving.

The reason is they are running in isolated Network. They are not connecting each other.

```
docker network ls
```

This command use to check the network.For solving previous problem we need to create new network:

First we need to destroy these 2 containers with its network:

```
docker compose down
```

Output:

```
(base) randika@Randikas-MacBook-Pro-2 docker % docker compose down
[+] Running 3/2
 ✓ Container test_pgadmin      Removed
 ✓ Container test_postgresql   Removed
 ✓ Network docker_default      Removed
(base) randika@Randikas-MacBook-Pro-2 docker %
```

Now add this to YAML file.

```
networks:
  test_scm_network:
```

And also Add this network name to Postgresql:

```
services:
  postgresql:
    image: postgres:15.4
    container_name: test_postgresql
    environment:
      POSTGRES_DB: app_db
      POSTGRES_USER: pguser
      POSTGRES_PASSWORD: pguser_password
    ports:
      - 5432:5432
    networks:
      - test_scm_network
```

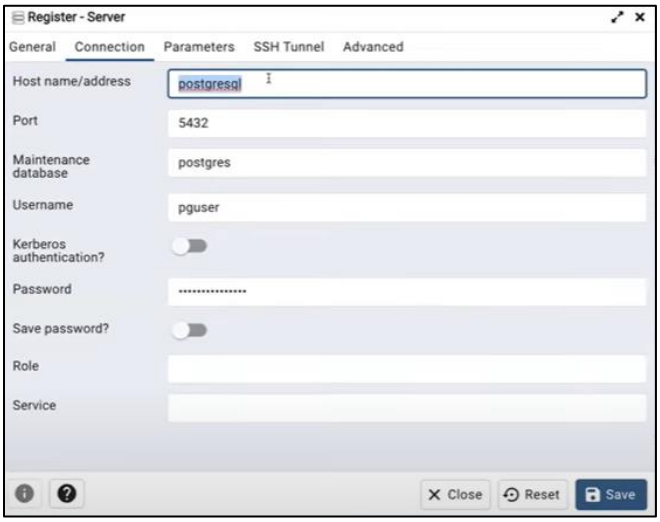
And also Add this network name to pgadmin:

```
pgadmin:
  image: dpage/pgadmin4:7.7
  container_name: test_pgadmin
  environment:
    PGADMIN_DEFAULT_EMAIL: admin@gmail.com
    PGADMIN_DEFAULT_PASSWORD: pguser_password2
  ports:
    - 8080:80
    networks:
    - test_scm_network
```

Now run this:

```
docker compose up
```

Now add the Host address as “Postgresql” and save:



Now we are in the same network , So we do not need this part:

```
ports:
  - 5432:5432
```

To stop the running container, we can use this command:

```
docker % docker compose stop
```

If we want to change the PostgreSQL database version and still need the data in it, we can get the backup of the data.

Other than that we can use the “VOLUME” for this.

```
volumes:
  postgresql-vol:
    driver: local
```

Add volume for each services available:

```yaml
services:
  postgresql:
    image: postgres:15.4
    container_name: test_postgresql
    environment:
      POSTGRES_DB: app_db
      POSTGRES_USER: pguser
      POSTGRES_PASSWORD: pguser_password
    networks:
      - test_scm_network
    volumes:
      - postgresql-vol:
```

Then we want to mention , what type of data is store in this volume.

```yaml
services:
  postgresql:
    image: postgres:15.4
    container_name: test_postgresql
    environment:
      POSTGRES_DB: app_db
      POSTGRES_USER: pguser
      POSTGRES_PASSWORD: pguser_password
    networks:
      - test_scm_network
    volumes:
      - postgresql-vol:/var/lib/postgresql/data/
```

We store these data in volume , Even container get down , volume have needed data.

Then we want to run this:

```
docker compose up
```

To see volumes , we can use this command:

```
docker volume ls
```

Output:

```
(base) randika@Randikas-MacBook-Pro-2 docker % docker volume ls
DRIVER    VOLUME NAME
local     1c6a44221c0617e80d9930f6489615ea34df8e1d66a2c0c573b5a575ea6cbc22
local     01d5f0e08f276f97aeb695736632100ad172d91f0870fa3dcfaf0f2a30b648c7
local     7f455dc463b702f76b4264305fd92978af6ea288bffe270273cab033aa0f42d7
local     8bd5008a4555a87be7c2cbf5d355606c39423a742f26905045aa5bc9cf8ce3dc
local     3149c7c4b645f7316a26ac12a1e8c74b125ed5038907be68c6427d9342c5e064
local     a159dadcf5c57aee68e1bd950ec01805e8e75a12ca320fdb0568c2991ade08c5
local     bloonsoo-api_bloonsoo-pgadmin
local     bloonsoo-api_bloonsoo-postgresql
local     db9a8adda368aca7a99472f07d88443f6f0a5790bd0c31881dcd0841d2ca1462
local     docker_postgresql-vol
local     esgsignals-2023_esg-2023-pgadmin
local     esgsignals-2023_esg-2023-postgresql
local     eshop-api_eshop-minio
local     eshop-api_eshop-pgadmin
local     eshop-api_eshop-postgresql
local     hawai_silveraisle-pgadmin
local     hawai_silveraisle-postgresql
(base) randika@Randikas-MacBook-Pro-2 docker %
```

Hosting Web Applications: The web service is typically used to host web applications or web services within Docker containers. It may contain the necessary configurations, dependencies, and code to run a web-based application.

```yaml
services:
  web:
    build: .
    container_name: test_web
```
