LAB 4 – DISCOVERING DOCKER

4.1 Installing docker on Ubuntu server:

4.1.1 Concepts container and image with docker:

Docker is a platform that simplifies the process of installing, running and managing applications using container technology. It allows developers to package their application with dependencies into containers ensuring consistent behavior across various enviroments – from local development to production.

Docker image is read only template that contains:

- The application code.
- Runtime (Python, Node.js)
- Libraries and dependencies
- System tools and settings

Characteristic of docker image:

- Immutable means "Once built, image don't change"
- Layered means "Built from series of layers that defined by dockerfile"
- Reusable means the same image could be shared across systems or teams

Example about dockerfile:

FROM ubuntu:24.04

RUN apt-get update && app-get install -y python3

COPY . /app

CMD ["python3", "/app/app.py"]

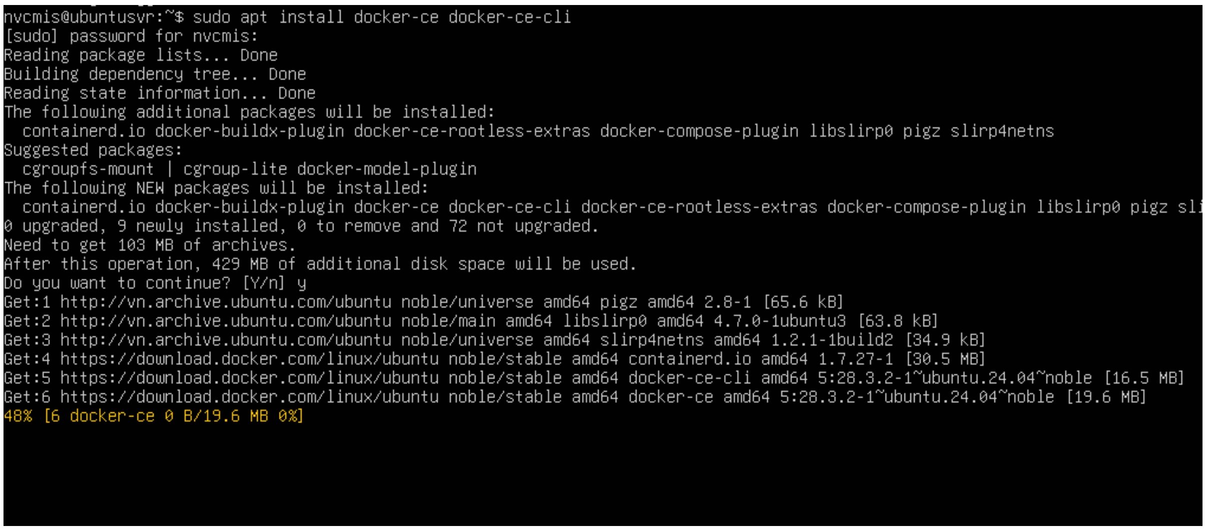
The dockerfile builds an image with Ubuntu, python installed, copy source code to app and set default command.

Docker Container is a running instance of docker image. It add a writable layer on top of the image. It is isolated from the host system and shared the kernel with the host.

4.1.2 Installing docker on Ubuntu server:

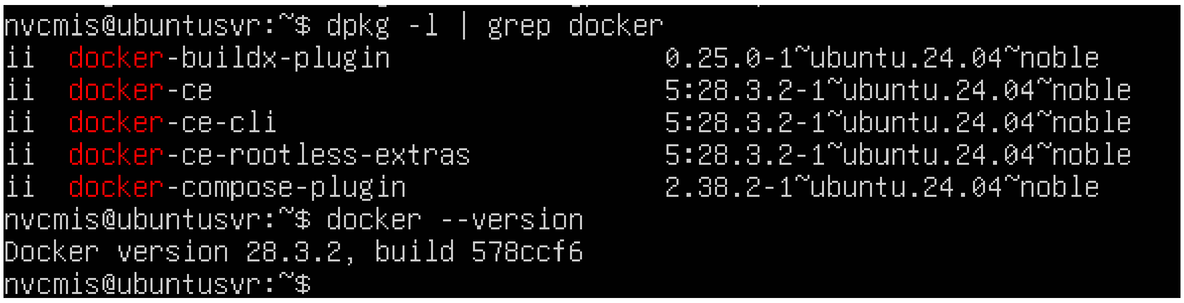
We have just noticed that sudo apt install <package_name> to install package in ubuntu operating system, docker is not an exception. What packages should be installed to use docker, we will clarify as the following:

The minimum docker for use consists of two essential packages. They are docker-ce represents docker engine and docker-ce-cli represents commands for docker. These includes docker build -t <image_name>, docker ps or docker run commands.



```
nvcmis@ubuntusvr:~$ sudo apt install docker-ce docker-ce-cli
[sudo] password for nvcmis:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-rootless-extras docker-compose-plugin libslirp0 pigz slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite docker-model-plugin
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libslirp0 pigz sli
0 upgraded, 9 newly installed, 0 to remove and 72 not upgraded.
Need to get 103 MB of archives.
After this operation, 429 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://vn.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://vn.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:3 http://vn.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:4 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.27-1 [30.5 MB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:28.3.2-1~ubuntu.24.04~noble [16.5 MB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:28.3.2-1~ubuntu.24.04~noble [19.6 MB]
48% [6 docker-ce 0 B/19.6 MB 0%]
```

Image 4.1 Install docker-ce and docker-ce-cli packages.



```
nvcmis@ubuntusvr:~$ dpkg -l | grep docker
ii  docker-buildx-plugin                0.25.0-1~ubuntu.24.04~noble
ii  docker-ce                           5:28.3.2-1~ubuntu.24.04~noble
ii  docker-ce-cli                       5:28.3.2-1~ubuntu.24.04~noble
ii  docker-ce-rootless-extras           5:28.3.2-1~ubuntu.24.04~noble
ii  docker-compose-plugin               2.38.2-1~ubuntu.24.04~noble
nvcmis@ubuntusvr:~$ docker --version
Docker version 28.3.2, build 578ccf6
nvcmis@ubuntusvr:~$
```

Image 4.2 Check if docker was installed or not.

4.2Common docker commands:

Check if docker has been installed in the system or not, use command:

docker --version

List running containers, use command: **docker ps**

Build an image: docker build -t <image_name> .

- -t <image_name>: Tags the image as <image_name>.
- . : Refers to the current directory (which has your Dockerfile).

Create a new container from a docker image and run the container:

```
docker run -d --name <container_name> <image_name>
```

Stop a container: docker stop <container_name>

Remove a container: docker rm <container_name>

Remove an image: docker rmi <image_name>

4.3Deploy DBMS MySql and Postgre with docker:

4.3.1 Deploy DBMS MySql on Docker:

Dockerfile is used to build an image that wraps all components needed. MySql itself has the image stored in the Docker hub. So we don't need create dockerfile to build another image. The name of mysql images available in Docker hub consists of mysql:latest, mysql:8.3

According to therory you could create a container with name mysql-server associated with image mysql:8.3 and run the container by the command:

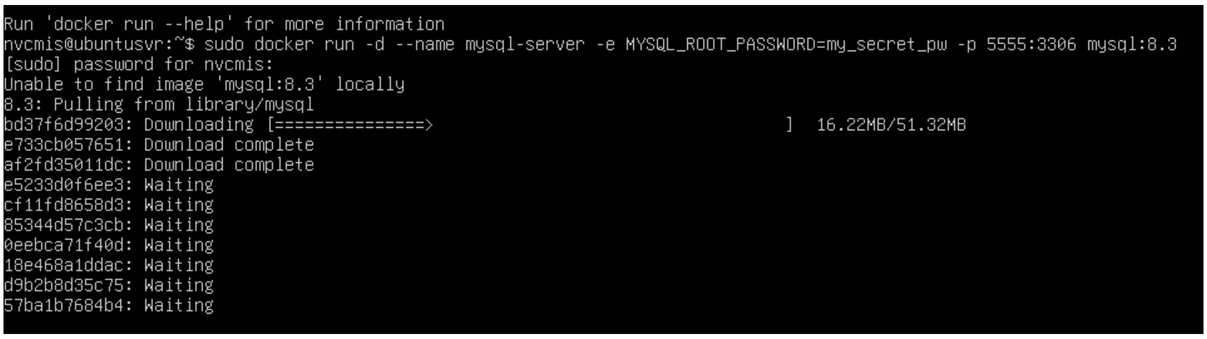
```
docker run -d -name mysql-server mysql:8.3
```

However, you couldn't do that if you don't set up enviroments variable such as password of root user, port number that access from outside to docker and port number inside mysql of docker.

The full version working command with docker to use mysql:

```
docker run -d \
  --name mysql-server \
  -e MYSQL_ROOT_PASSWORD=my-secret_pw \
  -p 3306:3306 \
  mysql:8.3
```

The -p 3306:3306, the first 3306 is the listening port from outside or port of ubuntu server, the second 3306 is the port of mysql inside container where mysql actual runs.

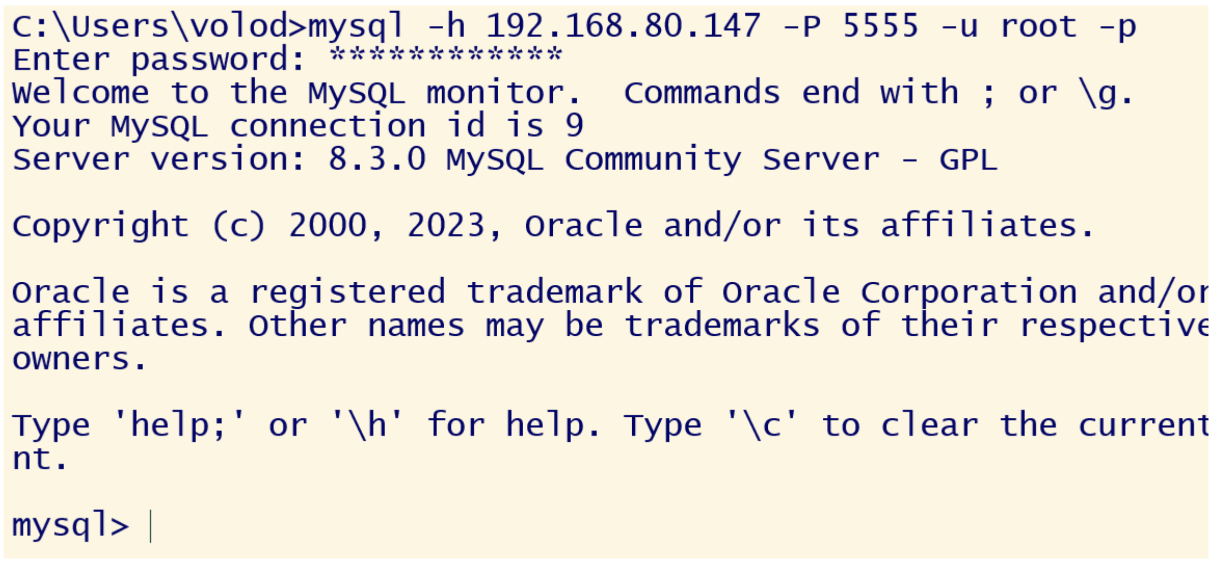


```
Run 'docker run --help' for more information
nvcmis@ubuntusvr:~$ sudo docker run -d --name mysql-server -e MYSQL_ROOT_PASSWORD=my_secret_pw -p 5555:3306 mysql:8.3
[sudo] password for nvcmis:
Unable to find image 'mysql:8.3' locally
8.3: Pulling from library/mysql
bd37f6d99203: Downloading [===============>                                    ]  16.22MB/51.32MB
e733cb057651: Download complete
af2fd35011dc: Download complete
e5233d0f6ee3: Waiting
cf11fd8658d3: Waiting
85344d57c3cb: Waiting
0eebca71f40d: Waiting
18e468a1ddac: Waiting
d9b2b8d35c75: Waiting
57ba1b7684b4: Waiting
```

Image 4.3 Create and run container associated with mysql:8.3 downloaded from docker hub

After finish installing mysql on docker with container mysql-server as image 3.3, I will try to connect to mysql from outside with port 5555 user root and password my_secret_pw

mysql -h <server-ip> -P 3306 -u root -p



```
C:\Users\volod>mysql -h 192.168.80.147 -P 5555 -u root -p
Enter password: ************
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current
nt.

mysql>
```

Image 4.4 Connect to mysql in docker from outside via port 5555

4.3.2 Deploy DBMS PostGreSql on Docker:

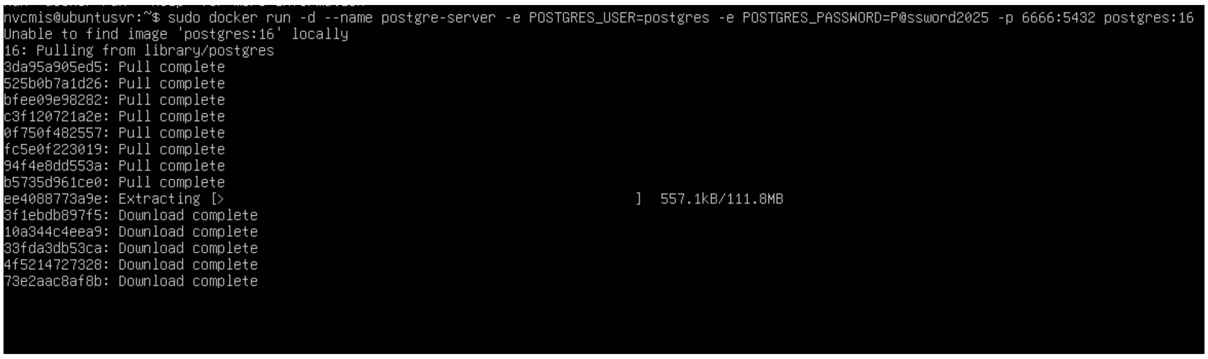
Similar to MySql, PostgreSQL has its own image in Docker Hub, for example postgres:16. You have to provide password for user postgre or admin to connect to database associated with port number for outside connection and port number for container connect to PostgreSQL.

Command to create and run container associated with postgres:16 image

```
docker run -d \
  --name postgres \
  -e POSTGRES_USER=admin \
  -e POSTGRES_PASSWORD=secretpassword \
  -e POSTGRES_DB=mydatabase \
  -v pgdata:/var/lib/postgresql/data \
  -p 5432:5432 \
  postgres:16
```

We could omit the line "-e POSTGRES_DB=mydatabase \" and change POSTGRES_USER to postgres and create new database later.

Practice: Deploy PostgreSQL DBMS with Docker on ubuntu server that listening connections from outside at port 6666, port container connect to PostgreSQL at 5432, postgres_user is postgres with password P@ssword2025



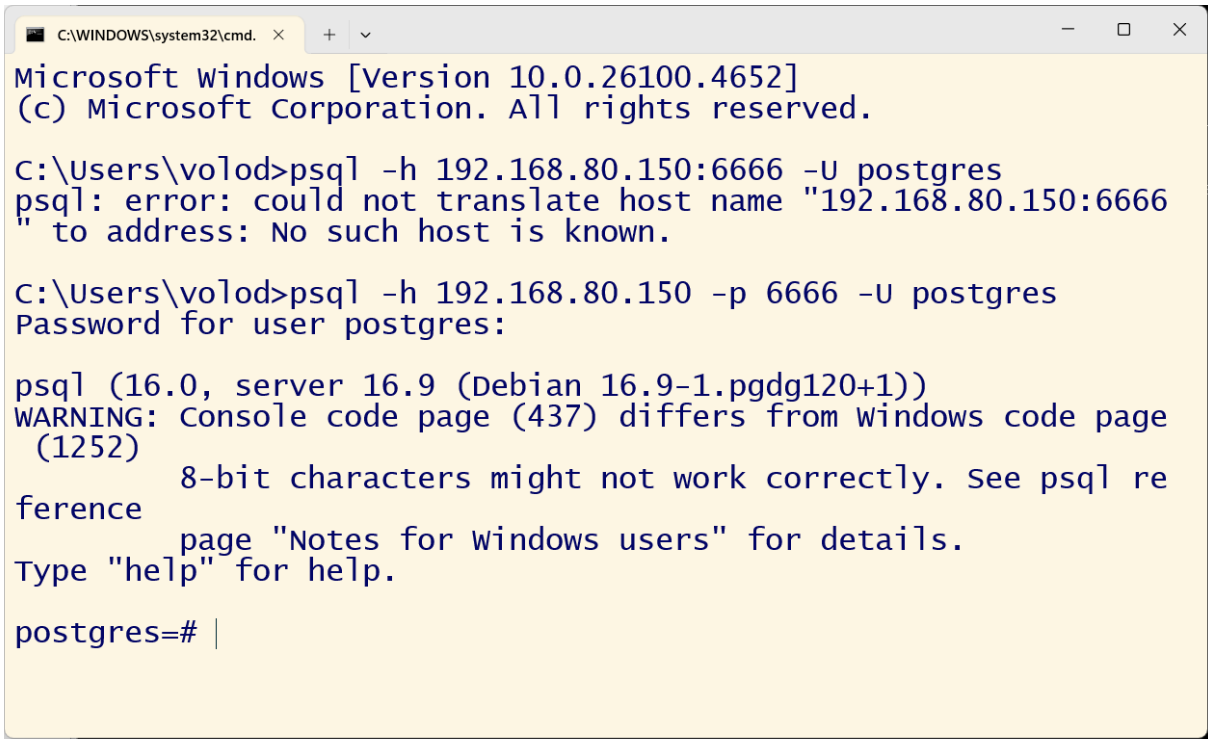
```
nvcmis@ubuntusvr:~$ sudo docker run -d --name postgre-server -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=P@ssword2025 -p 6666:5432 postgres:16
Unable to find image 'postgres:16' locally
16: Pulling from library/postgres
3da95a905ed5: Pull complete
525b0b7a1d26: Pull complete
bfee09e98282: Pull complete
c3f120721a2e: Pull complete
0f750f482557: Pull complete
fc5e0f223019: Pull complete
94f4e8dd553a: Pull complete
b5735d961ce0: Pull complete
ee4088773a9e: Extracting [>                                                  ]  557.1kB/111.8MB
3f1ebdb897f5: Download complete
10a344c4eea9: Download complete
33fda3db53ca: Download complete
4f5214727328: Download complete
73e2aac8af8b: Download complete
```

Image 4.5 Create and run container with postgres:16 image from Docker Hub

Now, we will connect to postgreSQL in docker via another machine, to do so, you have to make sure that postgresql-client must be installed in the remote machine

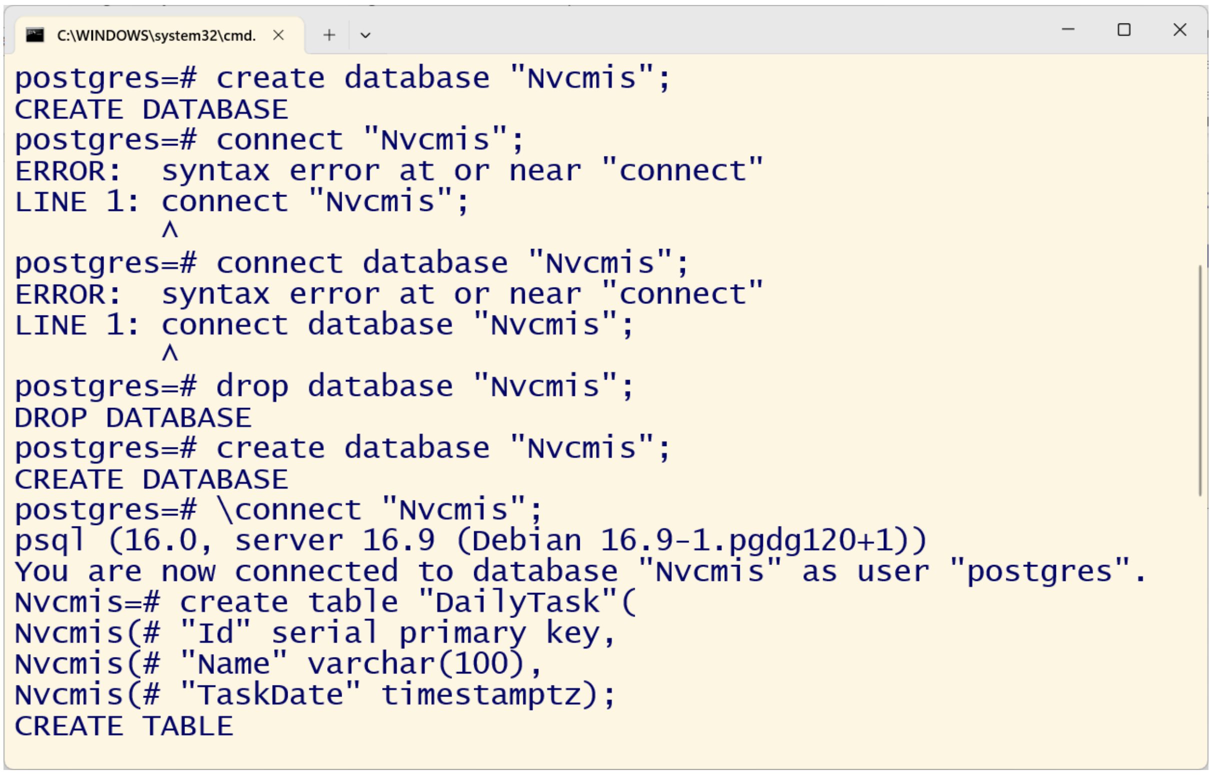
```
sudo apt install -y postgresql-client
```

```
psql -h <server_ip> -U <user_name> -p <port_number> -d mydatabase
```



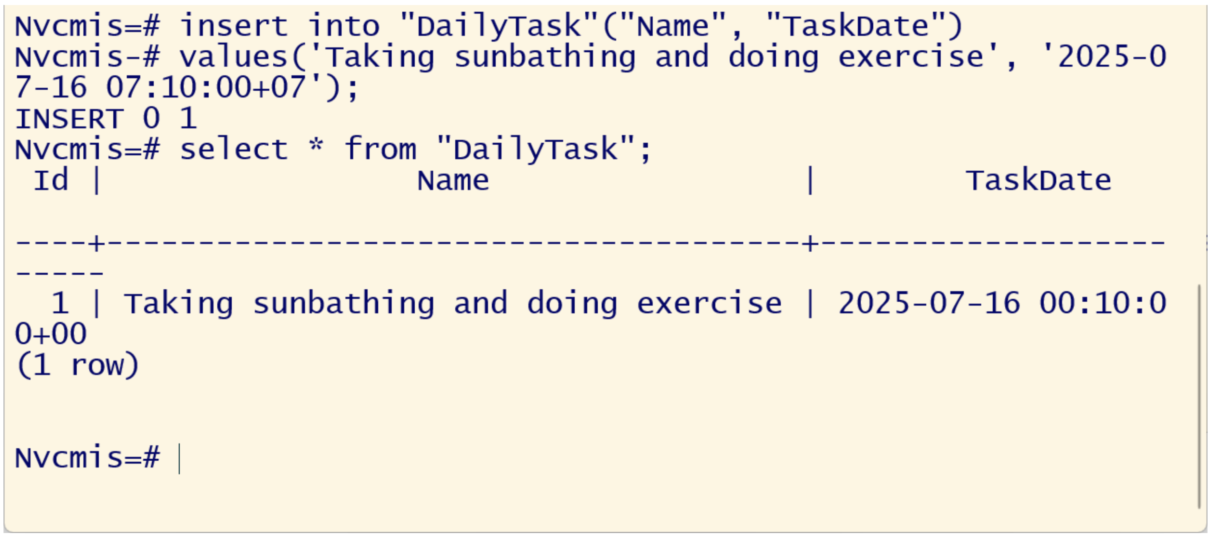
```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\volod>psql -h 192.168.80.150:6666 -U postgres
psql: error: could not translate host name "192.168.80.150:6666
" to address: No such host is known.

C:\Users\volod>psql -h 192.168.80.150 -p 6666 -U postgres
Password for user postgres:

psql (16.0, server 16.9 (Debian 16.9-1.pgdg120+1))
WARNING: Console code page (437) differs from Windows code page
 (1252)
         8-bit characters might not work correctly. See psql re
ference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=#
```

Image 4.6 Connect to PostgreSQL from remote machine.



```
postgres=# create database "Nvcmis";
CREATE DATABASE
postgres=# connect "Nvcmis";
ERROR:  syntax error at or near "connect"
LINE 1: connect "Nvcmis";
        ^
postgres=# connect database "Nvcmis";
ERROR:  syntax error at or near "connect"
LINE 1: connect database "Nvcmis";
        ^
postgres=# drop database "Nvcmis";
DROP DATABASE
postgres=# create database "Nvcmis";
CREATE DATABASE
postgres=# \connect "Nvcmis";
psql (16.0, server 16.9 (Debian 16.9-1.pgdg120+1))
You are now connected to database "Nvcmis" as user "postgres".
Nvcmis=# create table "DailyTask"(
Nvcmis(# "Id" serial primary key,
Nvcmis(# "Name" varchar(100),
Nvcmis(# "TaskDate" timestamptz);
CREATE TABLE
```

Image 4.7 Create database Nvcmis, create table DailyTask in Nvcmis



```
Nvcmis=# insert into "DailyTask"("Name", "TaskDate")
Nvcmis-# values('Taking sunbathing and doing exercise', '2025-0
7-16 07:10:00+07');
INSERT 0 1
Nvcmis=# select * from "DailyTask";
 Id |                 Name                  |        TaskDate

----+--------------------------------------+-------------------
-----
  1 | Taking sunbathing and doing exercise | 2025-07-16 00:10:0
0+00
(1 row)


Nvcmis=# 
```

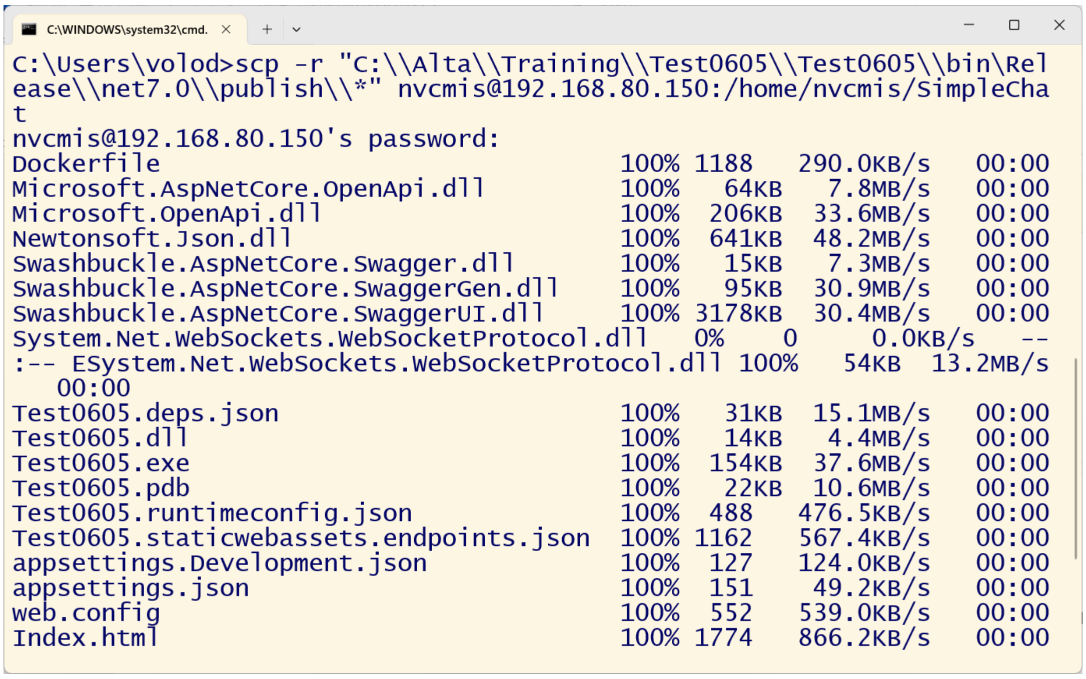
Image 4.8 Add row to table DailyTask and view it.

4.4 Deploy .net core application with docker:

Step 1: Publish application to Folder in .net using Visual studio, for example publish built source to:

C:\Alta\Training\Test0605\Test0605\bin\Release\net7.0\publish

Step 2: Using scp to copy all files in the folder to ubuntu server machine by command:



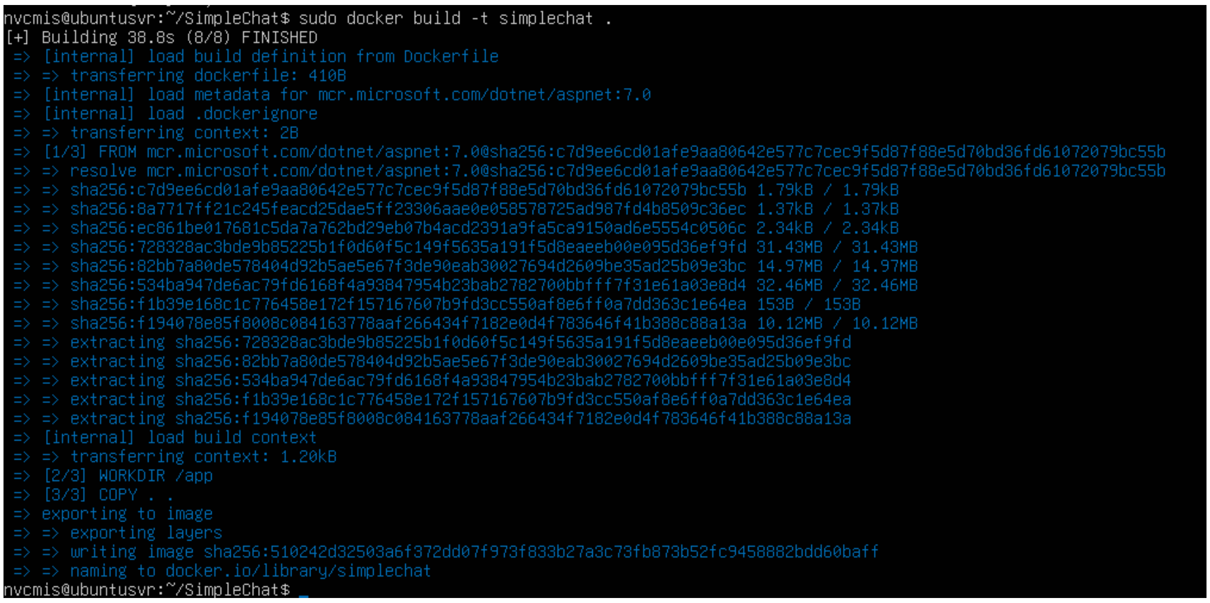
```
C:\Users\volod>scp -r "C:\\Alta\\Training\\Test0605\\Test0605\\bin\Rel
ease\\net7.0\\publish\\*" nvcmis@192.168.80.150:/home/nvcmis/SimpleCha
t
nvcmis@192.168.80.150's password:
Dockerfile                                  100% 1188   290.0KB/s   00:00
Microsoft.AspNetCore.OpenApi.dll            100%   64KB   7.8MB/s   00:00
Microsoft.OpenApi.dll                       100%  206KB  33.6MB/s   00:00
Newtonsoft.Json.dll                         100%  641KB  48.2MB/s   00:00
Swashbuckle.AspNetCore.Swagger.dll          100%   15KB   7.3MB/s   00:00
Swashbuckle.AspNetCore.SwaggerGen.dll       100%   95KB  30.9MB/s   00:00
Swashbuckle.AspNetCore.SwaggerUI.dll        100% 3178KB  30.4MB/s   00:00
System.Net.WebSockets.WebSocketProtocol.dll    0%    0      0.0KB/s   --
:-- ESystem.Net.WebSockets.WebSocketProtocol.dll 100%    54KB  13.2MB/s
   00:00
Test0605.deps.json                          100%   31KB  15.1MB/s   00:00
Test0605.dll                                100%   14KB   4.4MB/s   00:00
Test0605.exe                                100%  154KB  37.6MB/s   00:00
Test0605.pdb                                100%   22KB  10.6MB/s   00:00
Test0605.runtimeconfig.json                 100%  488    476.5KB/s   00:00
Test0605.staticwebassets.endpoints.json     100% 1162    567.4KB/s   00:00
appsettings.Development.json                100%  127    124.0KB/s   00:00
appsettings.json                            100%  151     49.2KB/s   00:00
web.config                                  100%  552    539.0KB/s   00:00
Index.html                                  100% 1774    866.2KB/s   00:00
```

Image 4.9 Copy source .net core application to ubuntu server

Step 3: Prepare Dockerfile with the following content:

```
FROM mcr.microsoft.com/dotnet/aspnet:7.0

WORKDIR /app

COPY . .

ENTRYPOINT ["dotnet", "Test0605.dll"]
```

Step 4: Build docker image

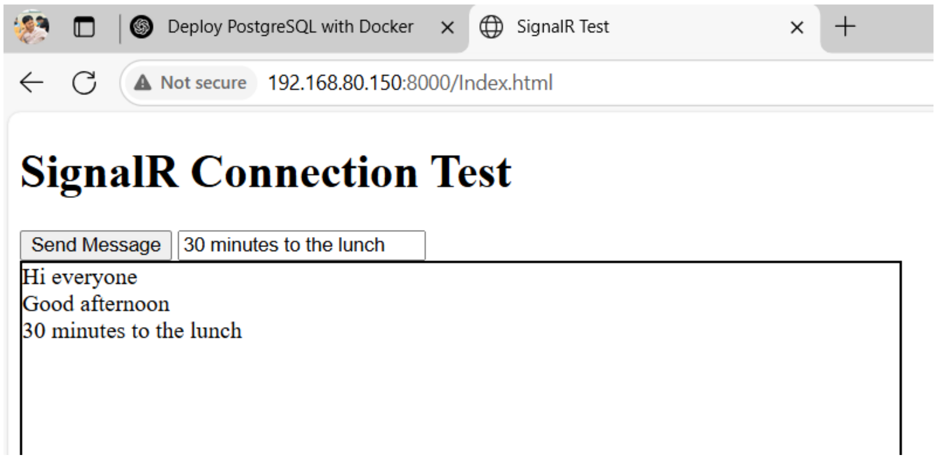


```
nvcmis@ubuntusvr:~/SimpleChat$ sudo docker build -t simplechat .
[+] Building 38.8s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 410B
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/3] FROM mcr.microsoft.com/dotnet/aspnet:7.0@sha256:c7d9ee6cd01afe9aa80642e577c7cec9f5d87f88e5d70bd36fd61072079bc55b
 => => resolve mcr.microsoft.com/dotnet/aspnet:7.0@sha256:c7d9ee6cd01afe9aa80642e577c7cec9f5d87f88e5d70bd36fd61072079bc55b
 => => sha256:c7d9ee6cd01afe9aa80642e577c7cec9f5d87f88e5d70bd36fd61072079bc55b 1.79kB / 1.79kB
 => => sha256:8a7717ff21c245feacd25dae5ff23306aae0e058578725ad987fd4b8509c36ec 1.37kB / 1.37kB
 => => sha256:ec861be017681c5da7a762bd29eb07b4acd2391a9fa5ca9150ad6e5554c0506c 2.34kB / 2.34kB
 => => sha256:728328ac3bde9b85225b1f0d60f5c149f5635a191f5d8eaeeb00e095d36ef9fd 31.43MB / 31.43MB
 => => sha256:82bb7a80de578404d92b5ae5e67f3de90eab30027694d2609be35ad25b09e3bc 14.97MB / 14.97MB
 => => sha256:534ba947de6ac79fd6168f4a93847954b23bab2782700bbfff7f31e61a03e8d4 32.46MB / 32.46MB
 => => sha256:f1b39e168c1c776458e172f157167607b9fd3cc550af8e6ff0a7dd363c1e64ea 153B / 153B
 => => sha256:f194078e85f8008c084163778aaf266434f7182e0d4f783646f41b388c88a13a 10.12MB / 10.12MB
 => => extracting sha256:728328ac3bde9b85225b1f0d60f5c149f5635a191f5d8eaeeb00e095d36ef9fd
 => => extracting sha256:82bb7a80de578404d92b5ae5e67f3de90eab30027694d2609be35ad25b09e3bc
 => => extracting sha256:534ba947de6ac79fd6168f4a93847954b23bab2782700bbfff7f31e61a03e8d4
 => => extracting sha256:f1b39e168c1c776458e172f157167607b9fd3cc550af8e6ff0a7dd363c1e64ea
 => => extracting sha256:f194078e85f8008c084163778aaf266434f7182e0d4f783646f41b388c88a13a
 => [internal] load build context
 => => transferring context: 1.20kB
 => [2/3] WORKDIR /app
 => [3/3] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:510242d32503a6f372dd07f973f833b27a3c73fb873b52fc9458882bdd60baff
 => => naming to docker.io/library/simplechat
nvcmis@ubuntusvr:~/SimpleChat$ _
```

Image 4.10 Build docker image simplechat with docker

Step 5: Create and run container associated with the simplechat image

```
docker run -d -p 8000:80 simplechat
```



SignalR Connection Test

30 minutes to the lunch

Hi everyone
Good afternoon
30 minutes to the lunch

Image 4.11 Run the .net core application via docker on ubuntu server