DOCKER

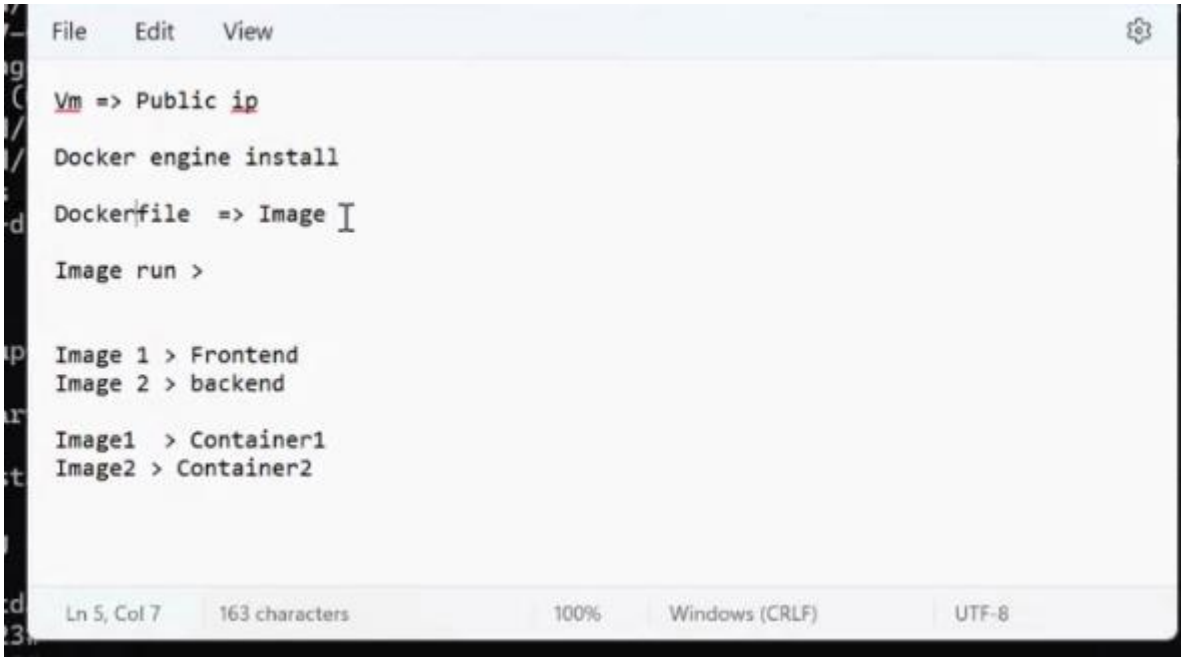
RG - rg-dooker

Frontend - VM

Backend - VM

Database – SQL

++



++

1) Create **rg** - rg-dooker

++

2) Create **vm** - vm-dooker

UN - azureuser

PW - Mommy7Daddy!

a) Open powershell

ssh azureuser@20.126.139.18

UN - azureuser

PW - Mommy7Daddy!

b) Install docker in vm using below url

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-22-04

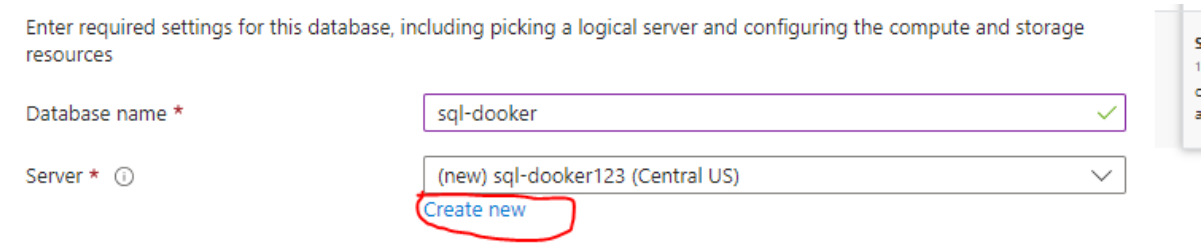
c) exit

d) **docker –version**

++

3) Create **sql database** – sql-dooker

a) Create new server - sql-dooker123



b) SQL authentication

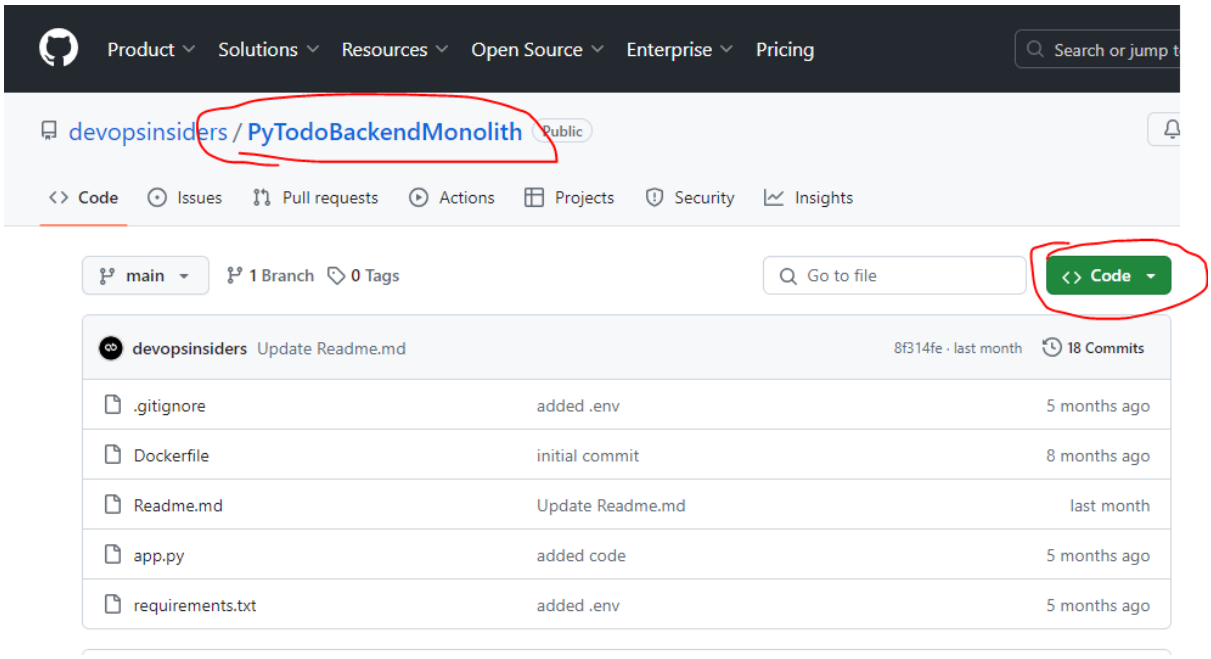
UN - azureuser

PW - Mommy7Daddy!

++

4) Now putting backend code in our vm machine from below url

https://github.com/devopsinsiders/PyTodoBackendMonolith



a) git clone https://github.com/devopsinsiders/PyTodoBackendMonolith.git

ls

```
For more help on how to use Docker, head to https://docs.docker.com/go/guides/
azureuser@vm-dooker:~$ git clone https://github.com/devopsinsiders/PyTodoBackendMonolith.git
Cloning into 'PyTodoBackendMonolith'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 59 (delta 30), reused 21 (delta 5), pack-reused 0
Receiving objects: 100% (59/59), 14.69 KiB | 1.05 MiB/s, done.
Resolving deltas: 100% (30/30), done.
azureuser@vm-dooker:~$ ls
PyTodoBackendMonolith
azureuser@vm-dooker:~$
```

b) **cd PyTodoBackendMonolith/**

ls

```
azureuser@vm-dooker:~$ cd PyTodoBackendMonolith/
azureuser@vm-dooker:~/PyTodoBackendMonolith$ ls
Dockerfile  Readme.md  app.py  requirements.txt
azureuser@vm-dooker:~/PyTodoBackendMonolith$
```

c) Now we need an image from docker hub that has python and pip installed in it as per perquisites

So by installing python image our time got reduced since image already have pip and python installed in it

d) SEARCH – Docker reference

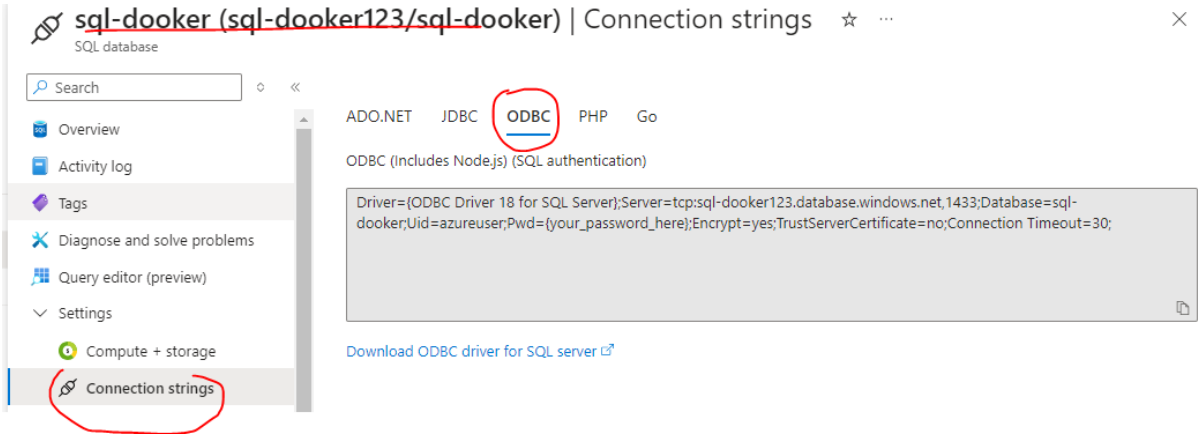
https://docs.docker.com/reference/dockerfile/

e) **cat Dockerfile**

Just to check what is there inside

```
azureuser@vm-dooker:~/PyTodoBackendMonolith$ cat Dockerfile
# Use the official Python image as the base image
FROM python:3.9

# Set the working directory in the container
WORKDIR /app

# Copy the application files into the container
COPY . .

# Install necessary packages
RUN apt-get update && apt-get install -y unixodbc unixodbc-dev
RUN curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -
RUN curl https://packages.microsoft.com/config/debian/10/prod.list > /etc/apt/sources.list.d/mssql-release.list
RUN apt-get update
RUN ACCEPT_EULA=Y apt-get install -y msodbcsql17

RUN pip install -r requirements.txt

# Start the FastAPI application
azureuser@vm-dooker:~/PyTodoBackendMonolith$ client_loop: send disconnect: Connection reset
PS C:\Users\HP>
```

f) Now go to sql db and copy “connection string”



g) docker images

To check whether any docker image is there or not

h) Now we will build docker image, so we will build docker in that folder only that contains docker files

docker build -t backendimage .

i) docker images

```
=> => naming to docker.io/library/backendimage
root@vm-dooker:/home/azureuser/PyTodoBackendMonolith# docker images
REPOSITORY     TAG       IMAGE ID       CREATED              SIZE
backendimage   latest    c44e3af7ffaf   About a minute ago   1.12GB
root@vm-dooker:/home/azureuser/PyTodoBackendMonolith#
```

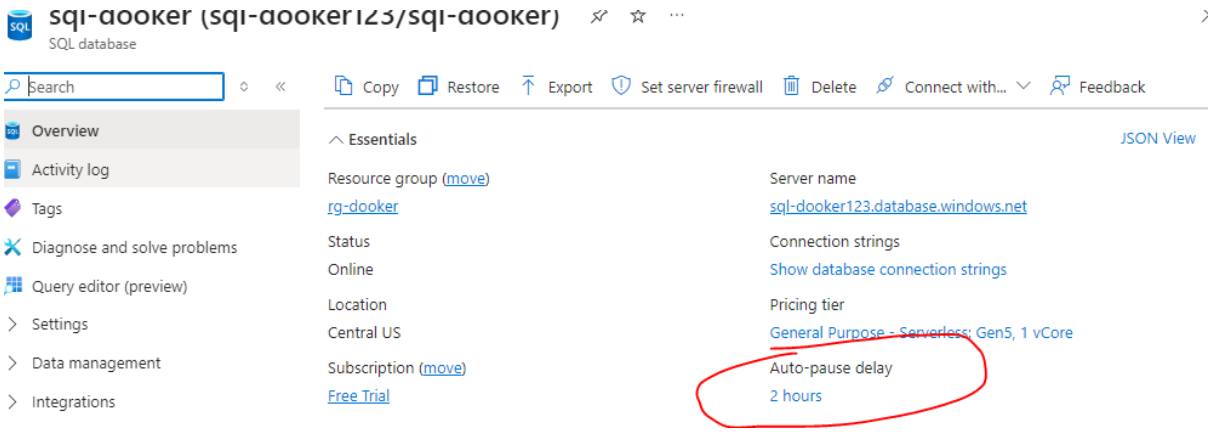
j) nano app.py

Update connection string by changing password

Driver={ODBC Driver 17 for SQL Server};Server=tcp:sql-dooker123.database.windows.net,1433;Database=sql-dooker;Uid=azureuser;Pwd={Mommy7Daddy!};Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30;

```
# Load environment variables from .env file
load_dotenv()

connection_string = "Driver={ODBC Driver 17 for SQL Server};Server=tcp:sql-dooker123.database.windows.net,1433;Database=sql-dooker;Uid=azureuser;Pwd={ Mommy7Daddy!};Encrypt=yes;TrustServerCe
rtificate=no;Connection Timeout=30;"

app = FastAPI()
```

K) We can bring sql db online by clicking and changing time as below



l) docker run -dp 8000:8000 backendimage

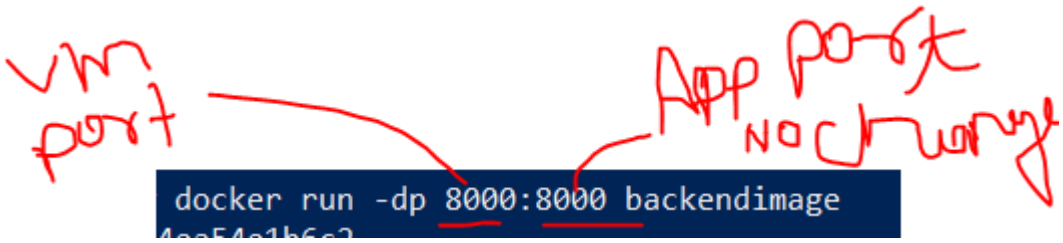
here in dp – d means detach mode and p means port exposing

also we use i – interactive mode

t – terminal

p- port

```
backendimage    latest    c44e3af7ffaf   33 minutes ago   1.12GB
root@vm-dooker:/home/azureuser/PyTodoBackendMonolith# docker run -dp 8000:8000 backendimage
46a5f71f4483185d445b4aabea1ebdd0a087031930123a2b4bd5a4ea54e1b6c2
root@vm-dooker:/home/azureuser/PyTodoBackendMonolith#
```

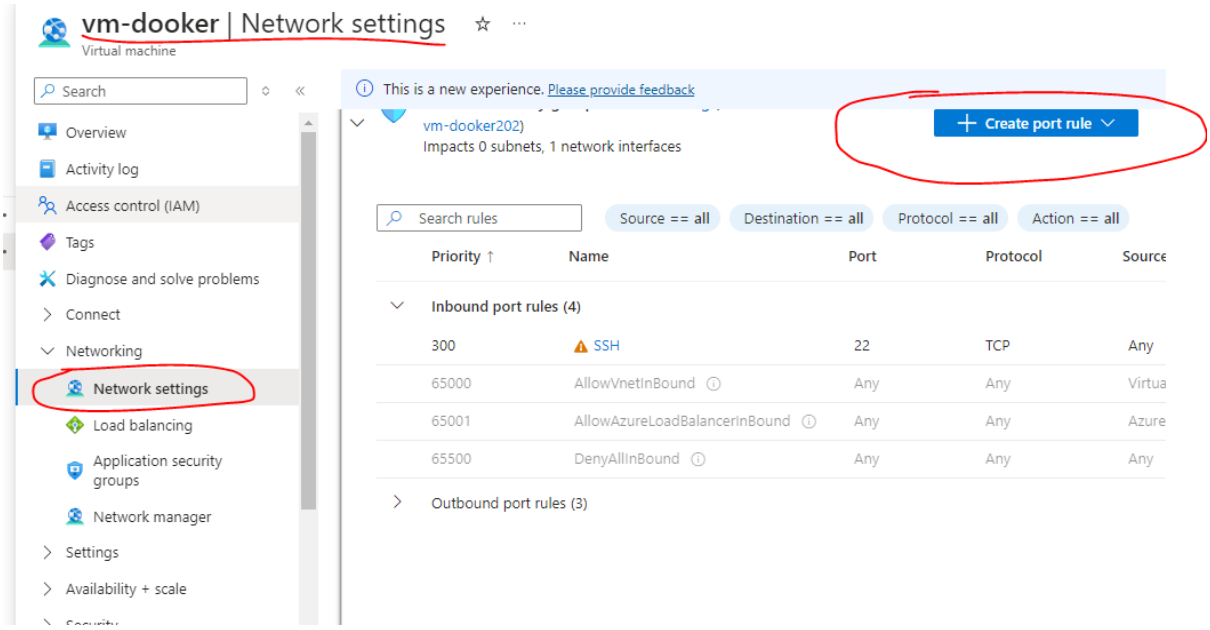


m) docker ps

docker ps -a

```
root@vm-dooker:/home/azureuser/PyTodoBackendMonolith# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                                       NAMES
46a5f71f4483   backendimage   "uvicorn app:app --h…"   7 minutes ago   Up 7 minutes   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   sharp_cori
root@vm-dooker:/home/azureuser/PyTodoBackendMonolith#
```

n) Open 8000 port on vm



o) Now run on browser – **public ip of vm:port**

http://20.126.139.18:8000/