DOCKER

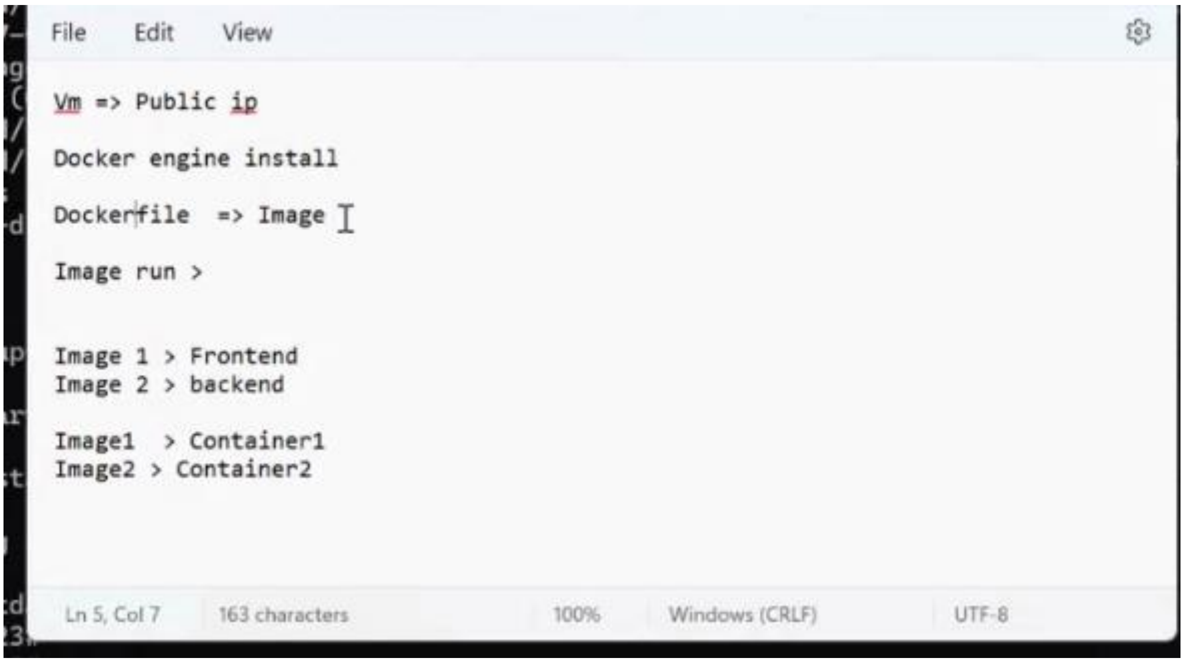
RG - rg-dooker

Frontend - VM

Backend - VM

Database – SQL

++



```
Vm => Public ip

Docker engine install

Dockerfile  => Image

Image run >


Image 1 > Frontend
Image 2 > backend

Image1  > Container1
Image2 > Container2
```

++

AGENDA - INSTALL DOCKER IN VM

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) Ctrl+c = exit

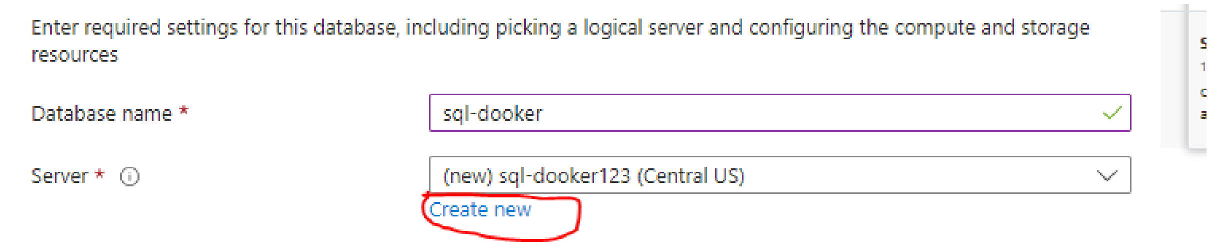
d) **docker version**

++

AGENDA - CREATE SQL DATABASE IN PORTAL

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

a) Create new server - sql-dooker666



Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *	sql-dooker
Server *	(new) sql-dooker123 (Central US)

b) SQL authentication

UN - azureuser

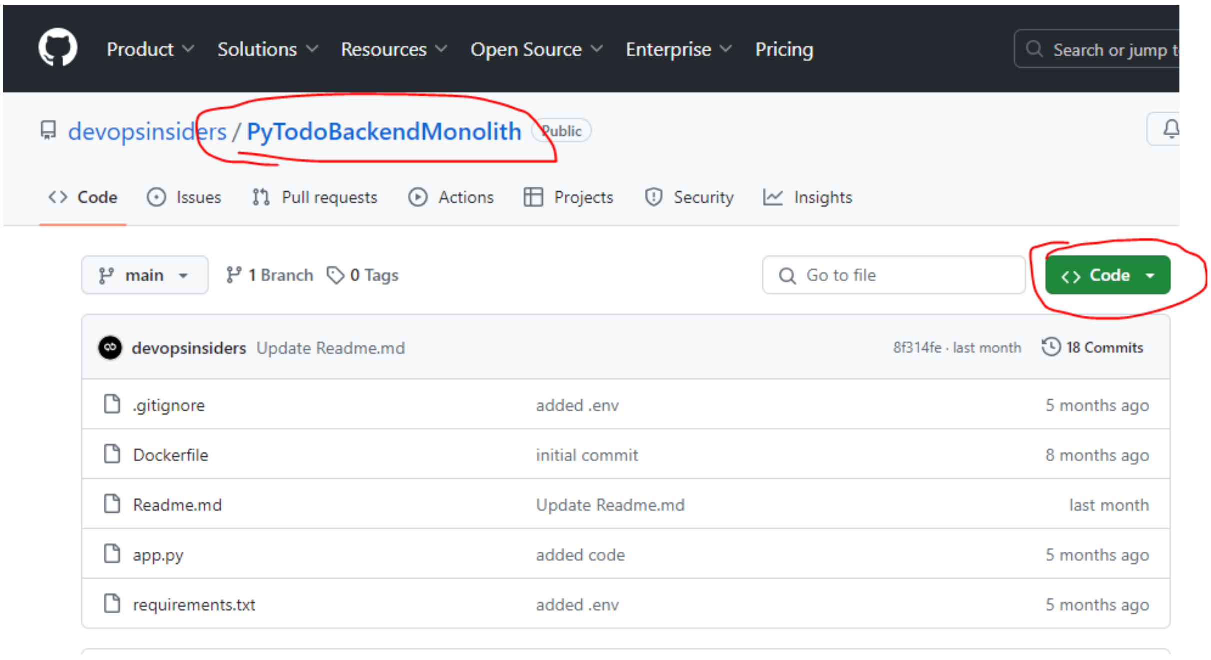
PW - Mommy7Daddy!

++

BACKEND IMAGE

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

https://github.com/devopsinsiders/PyTodoBackendMonolith



devopsinsiders Update Readme.md 8f314fe · last month 18 Commits

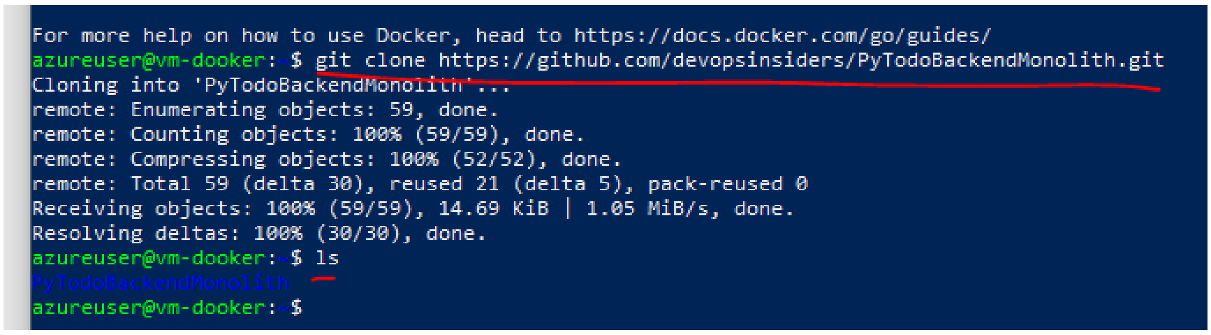
.gitignore	added .env	5 months ago
Dockerfile	initial commit	8 months ago
Readme.md	Update Readme.md	last month
app.py	added code	5 months ago
requirements.txt	added .env	5 months ago

NOTE : Here Vm is acting like our own computer.

a) In powershell do git clone of backend of todo app

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

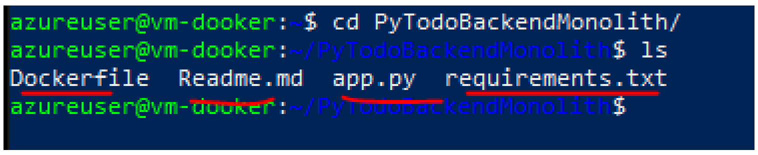
ls



```
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:~$
```

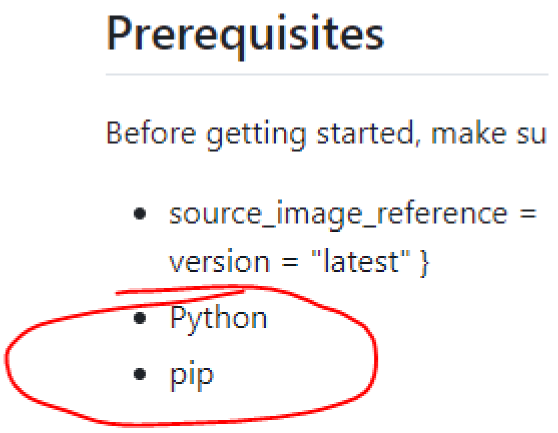
b) cd PyTodoBackendMonolith/

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 a image from docker hub that has **python** and **pip** installed in it as per perquisites



Prerequisites

Before getting started, make su

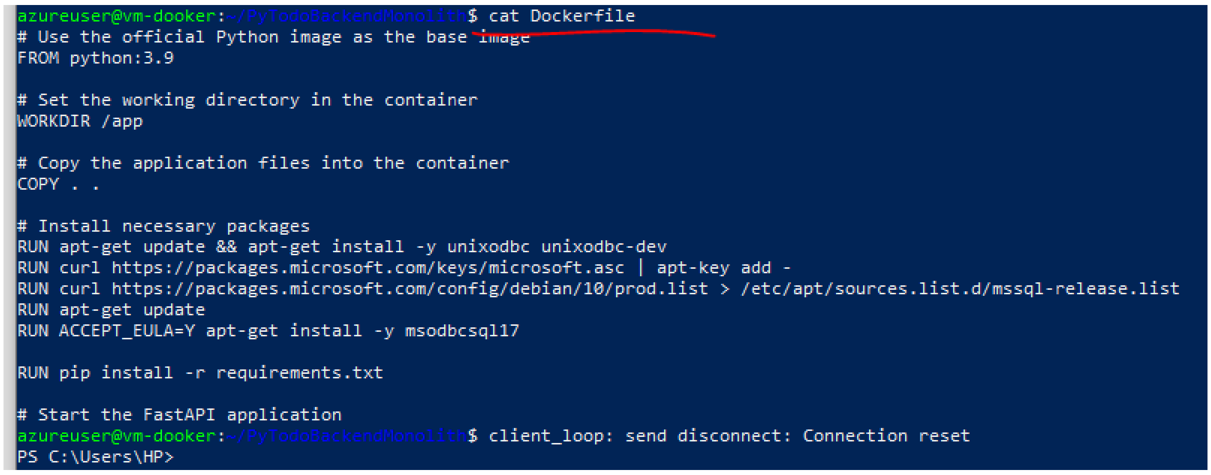
- source_image_reference =
  version = "latest" }
- Python
- pip

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

d) SEARCH – Docker reference

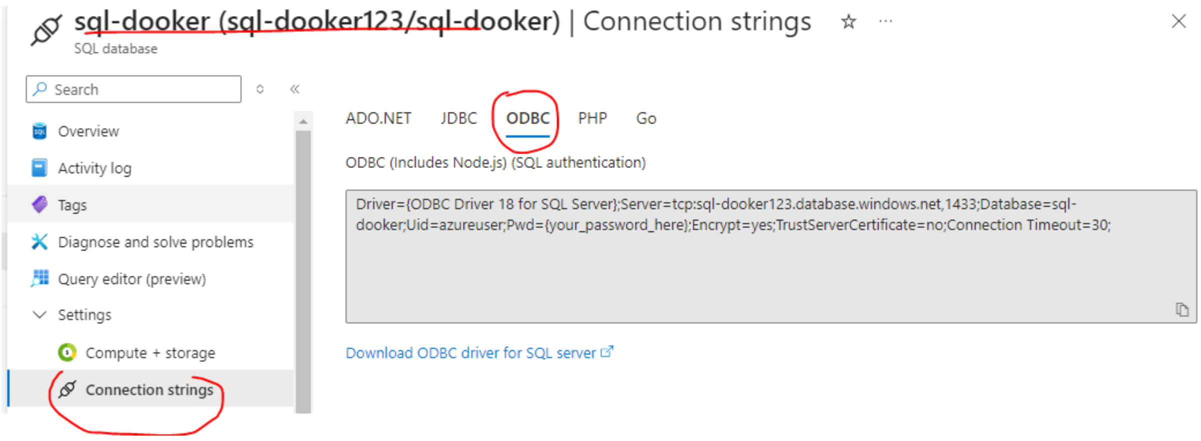
https://docs.docker.com/reference/dockerfile/

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"**


sql-dooker (sql-dooker123/sql-dooker) | Connection strings
SQL database
Search
Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)
Settings
Compute + storage
Connection strings
ADO.NET
JDBC
ODBC
PHP
Go
ODBC (Includes Node.js) (SQL authentication)
Download ODBC driver for SQL server

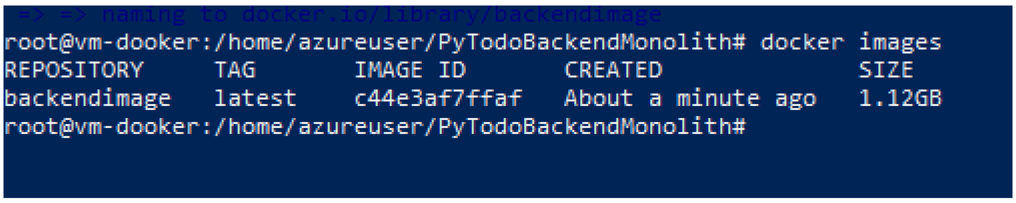
```
Driver={ODBC Driver 18 for SQL Server};Server=tcp:sql-dooker123.database.windows.net,1433;Database=sql-dooker;Uid=azureuser;Pwd={your_password_here};Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30;
```

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 .

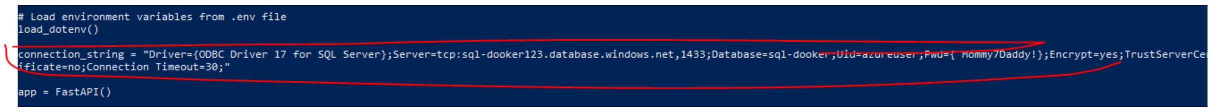
i) docker images



```
 => => 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#
```

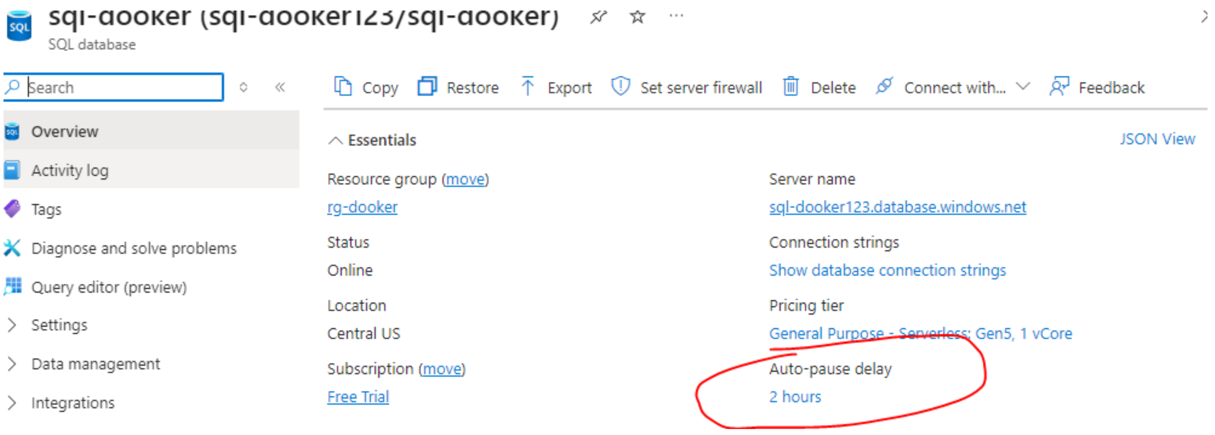
j) nano app.py - Update connection string by changing password

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
ificate=no;Connection Timeout=30;"

app = FastAPI()
```

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



sql-dooker (sql-dooker123/sql-dooker)
SQL database

Resource group (move): rg-dooker
Status: Online
Location: Central US
Subscription (move): Free Trial
Server name: sql-dooker123.database.windows.net
Connection strings: Show database connection strings
Pricing tier: General Purpose - Serverless: Gen5, 1 vCore
Auto-pause delay: 2 hours

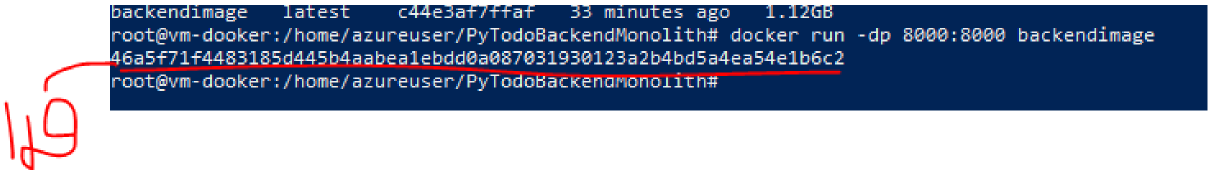
l) docker run -dp 8000:8000 backendimage

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

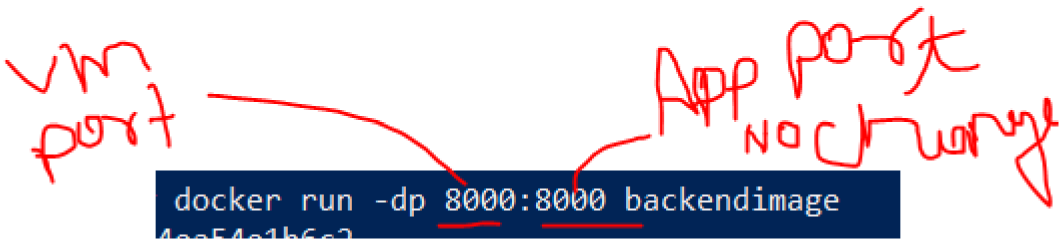
also we use i – interactive mode

t – terminal

p- port



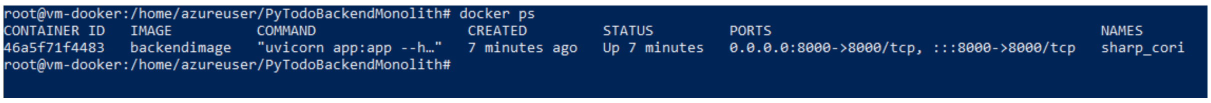
```
backendimage   latest    c44e3af7ffaf   33 minutes ago   1.12GB
root@vm-dooker:/home/azureuser/PyTodoBackendMonolith# docker run -dp 8000:8000 backendimage
46a5f71f4483185d445b4aabea1ebdd0a087031930123a2b4bd5a4ea54e1b6c2
root@vm-dooker:/home/azureuser/PyTodoBackendMonolith#
```



```
docker run -dp 8000:8000 backendimage
```

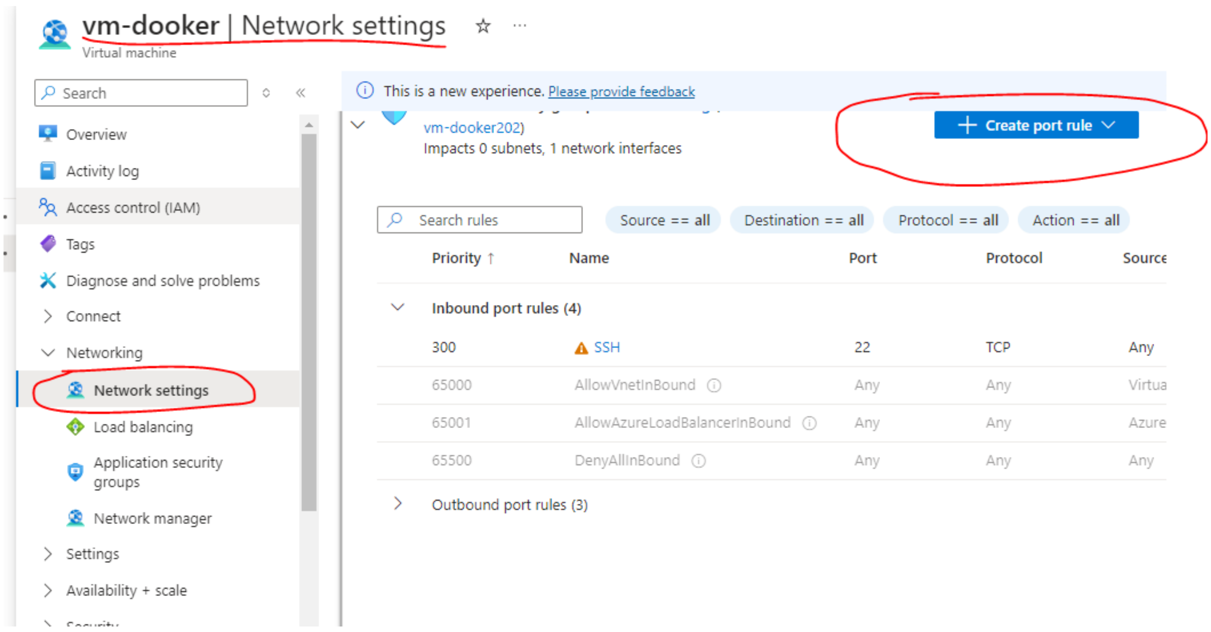
m) docker ps – to check running containers

docker ps –a – to see running + stopped containers both



```
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 in network settings


vm-dooker | Network settings

Virtual machine

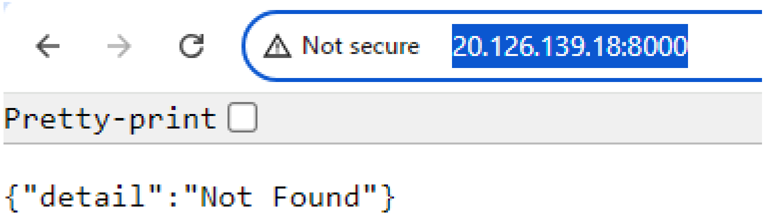
This is a new experience. Please provide feedback

vm-dooker202)
Impacts 0 subnets, 1 network interfaces

Priority ↑	Name	Port	Protocol	Source
Inbound port rules (4)				
300	SSH	22	TCP	Any
65000	AllowVnetInBound	Any	Any	Virtua
65001	AllowAzureLoadBalancerInBound	Any	Any	Azure
65500	DenyAllInBound	Any	Any	Any
Outbound port rules (3)				

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

http://20.126.139.18:8000/


Not secure 20.126.139.18:8000

Pretty-print

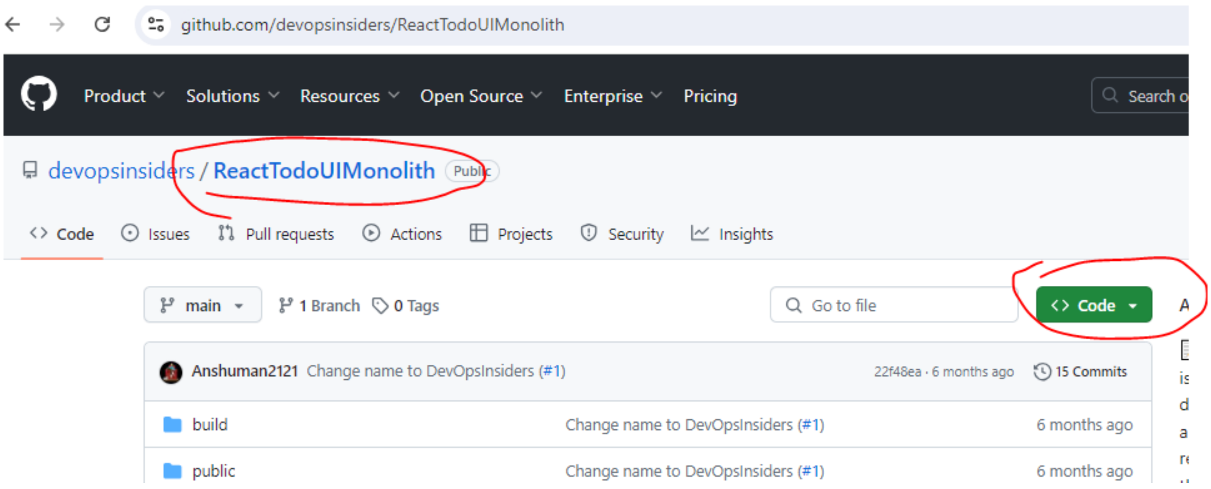
```
{"detail":"Not Found"}
```

++

FRONTEND IMAGE

1) Now putting frontend code in our vm machine from below url

https://github.com/devopsinsiders/ReactTodoUIMonolith


github.com/devopsinsiders/ReactTodoUIMonolith

devopsinsiders / ReactTodoUIMonolith Public

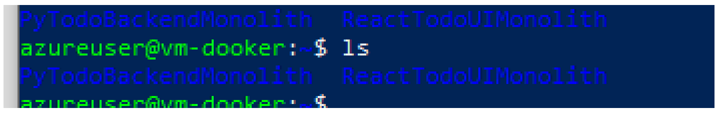
Anshuman2121 Change name to DevOpsInsiders (#1) 22f48ea · 6 months ago 15 Commits

build	Change name to DevOpsInsiders (#1)	6 months ago
public	Change name to DevOpsInsiders (#1)	6 months ago

a) In powershell do git clone of frontend of todo app in our vm

git clone https://github.com/devopsinsiders/ReactTodoUIMonolith.git

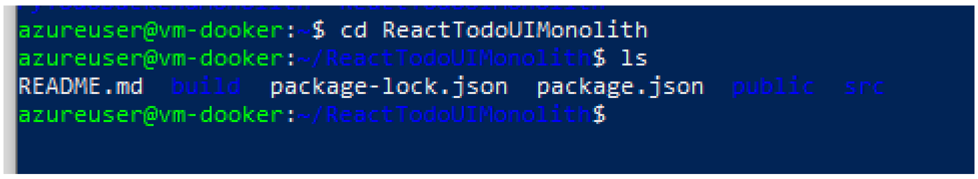
ls



```
PyTodoBackendMonolith  ReactTodoUIMonolith
azureuser@vm-dooker:~$ ls
PyTodoBackendMonolith  ReactTodoUIMonolith
azureuser@vm-dooker:~$
```

b) **cd ReactTodoUIMonolith –** Now changing into directory of frontend one

ls



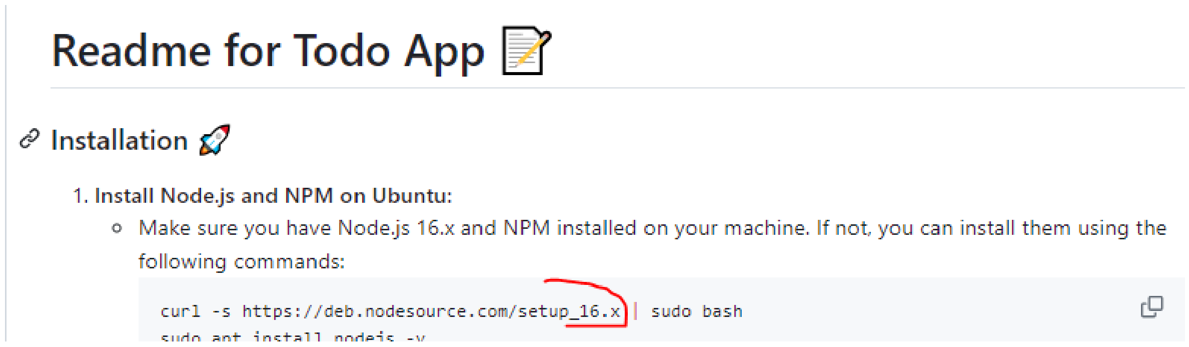
```
azureuser@vm-dooker:~$ cd ReactTodoUIMonolith
azureuser@vm-dooker:~/ReactTodoUIMonolith$ ls
README.md  build  package-lock.json  package.json  public  src
azureuser@vm-dooker:~/ReactTodoUIMonolith$
```

c) Versions types

i) major

ii) minor

iii) patch



Readme for Todo App 📝

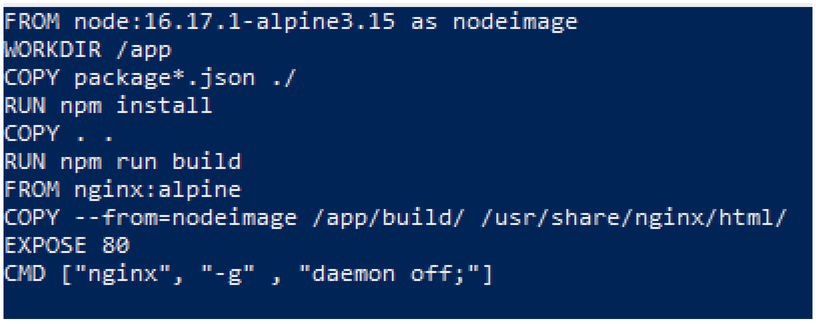
Installation 🚀

1. Install Node.js and NPM on Ubuntu:
   - Make sure you have Node.js 16.x and NPM installed on your machine. If not, you can install them using the following commands:

```
curl -s https://deb.nodesource.com/setup_16.x | sudo bash
sudo apt install nodejs -y
```

d) **touch Dockerfile -** create file names as Dockerfile

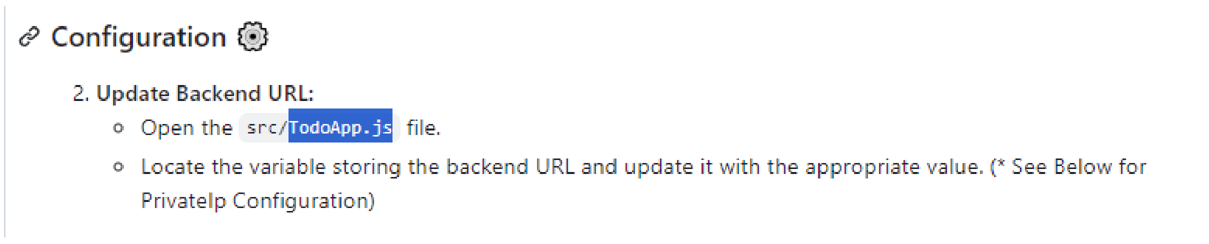
nano Dockerfile and write content in it



```
FROM node:16.17.1-alpine3.15 as nodeimage
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
FROM nginx:alpine
COPY --from=nodeimage /app/build/ /usr/share/nginx/html/
EXPOSE 80
CMD ["nginx", "-g" , "daemon off;"]
```

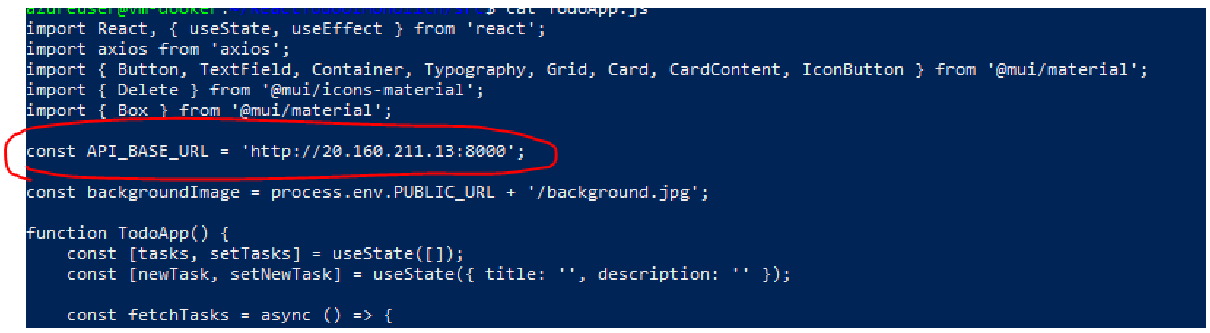
e) cd src/

f) Update backend url in **TodoApp.js file** in filed const API_BASE_URL = “Url of backend put here”



🔗 Configuration ⚙️

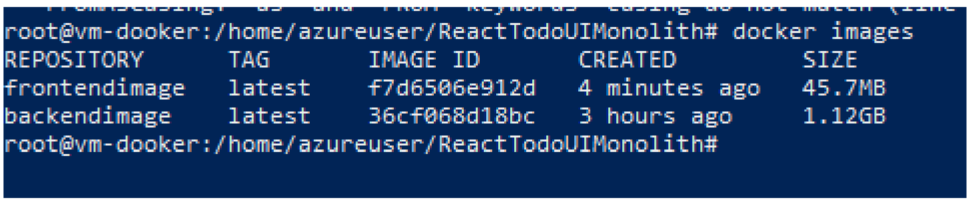
2. Update Backend URL:
   - Open the `src/TodoApp.js` file.
   - Locate the variable storing the backend URL and update it with the appropriate value. (* See Below for PrivateIp Configuration)



```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Button, TextField, Container, Typography, Grid, Card, CardContent, IconButton } from '@mui/material';
import { Delete } from '@mui/icons-material';
import { Box } from '@mui/material';

const API_BASE_URL = 'http://20.160.211.13:8000';

const backgroundImage = process.env.PUBLIC_URL + '/background.jpg';

function TodoApp() {
    const [tasks, setTasks] = useState([]);
    const [newTask, setNewTask] = useState({ title: '', description: '' });

    const fetchTasks = async () => {
```

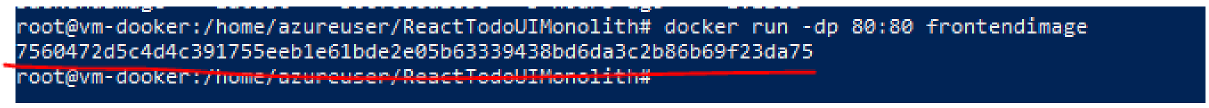
g) docker build -t frontendimage . – to build our frontend image

h) docker images



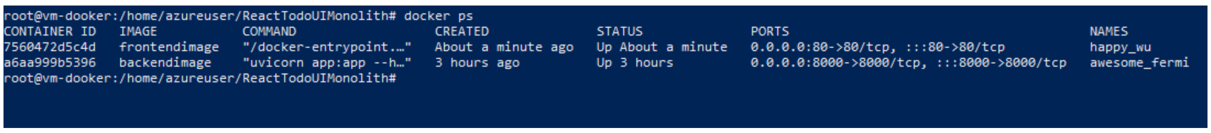
```
root@vm-dooker:/home/azureuser/ReactTodoUIMonolith# docker images
REPOSITORY      TAG       IMAGE ID       CREATED         SIZE
frontendimage   latest    f7d6506e912d   4 minutes ago   45.7MB
backendimage    latest    36cf068d18bc   3 hours ago     1.12GB
root@vm-dooker:/home/azureuser/ReactTodoUIMonolith#
```

i) **docker run -dp 80:80 frontendimage**



```
root@vm-dooker:/home/azureuser/ReactTodoUIMonolith# docker run -dp 80:80 frontendimage
7560472d5c4d4c391755eeb1e61bde2e05b63339438bd6da3c2b86b69f23da75
root@vm-dooker:/home/azureuser/ReactTodoUIMonolith#
```

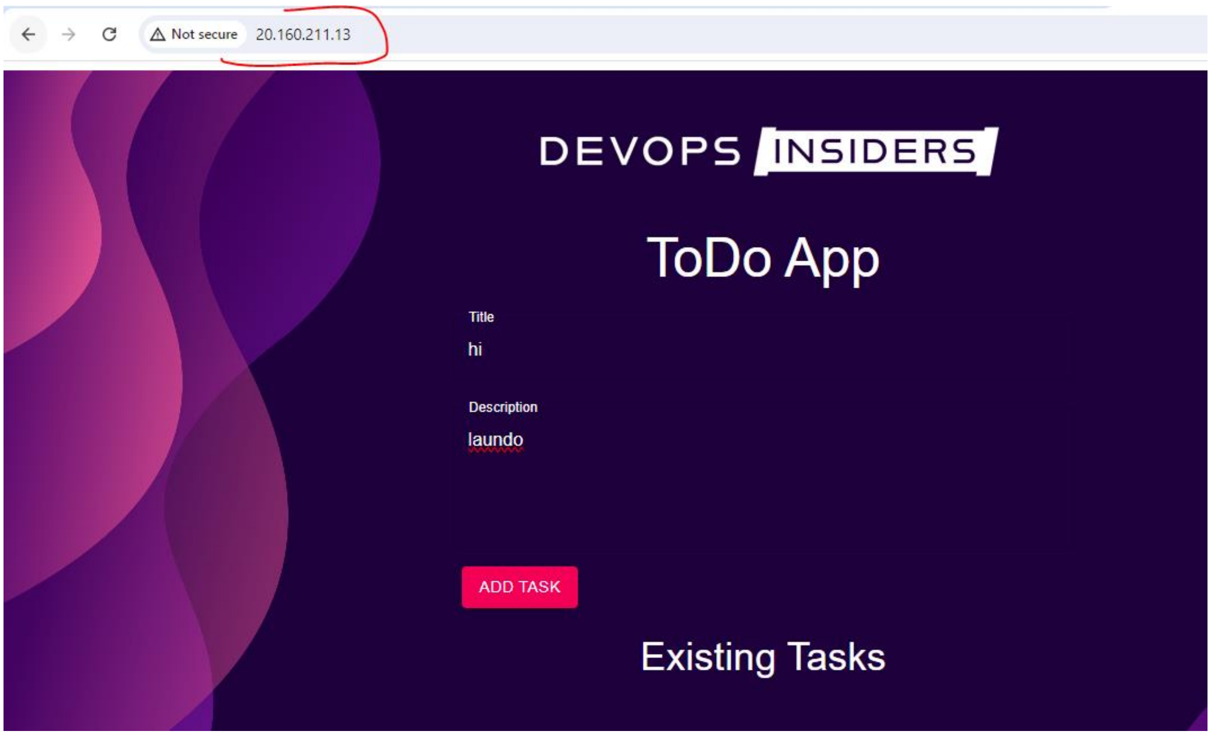
j) **docker ps**



```
root@vm-dooker:/home/azureuser/ReactTodoUIMonolith# docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED              STATUS              PORTS                                       NAMES
7560472d5c4d   frontendimage   "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:80->80/tcp, :::80->80/tcp           happy_wu
a6aa999b5396   backendimage    "uvicorn app:app --h…"   3 hours ago          Up 3 hours          0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   awesome_fermi
root@vm-dooker:/home/azureuser/ReactTodoUIMonolith#
```

k) Now open port 80 in vm for frontend

l) Run ip of vm in browser



m) Now suppose we have to stop the container

docker stop id of that container

docker stop 7560472d5c4d