Docker setup

Docker setup for Django project dependent on **MYSQL** and **REDIS**

Prepared By: Muhammad Abubakar

Github: Abubakar26

Document version: v1.0

Table of Contents

Prerequisites 3
System Requirements 3
Docker desktop requirements for Windows 3
Installing Docker-Desktop for windows 4
Setting up Docker-Desktop windows 4
Setting up docker environment for your project 5
Writing a Docker file 5
Writing Docker-compose.yml file 6
Executing and Running the project 7

Prerequisites

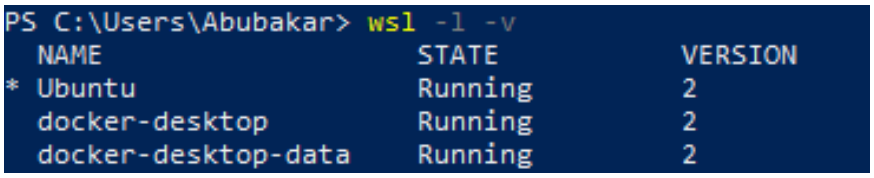
System Requirements

- Windows 10 64-bit: Pro, Enterprise, or Education (Build 17134 or higher).
- The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
  - 64-bit processor with Second Level Address Translation (SLAT)
  - 4GB system RAM
  - BIOS-level hardware virtualization support must be enabled in the BIOS settings. For more information, see Virtualization.

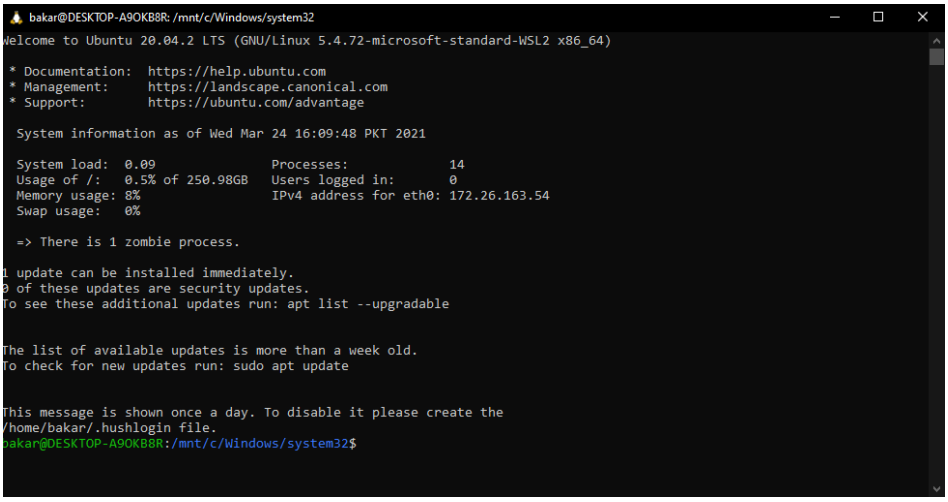
Docker desktop requirements for Windows

- Install Windows subsystem for Linux WSL using this link https://docs.microsoft.com/en-us/windows/wsl/install-win10
- Make sure your WSL is WSL 2. To check the version of your installed WSL run the following command on your PowerShell

wsl -l -v



- If your WSL is not on WSL 2 follow this link to convert your WSL 1 to WSL 2 https://ubuntu.com/blog/ubuntu-on-wsl-2-is-generally-available
- After installing the WSL install the Linux distro to run on your WSL you can download it from the Microsoft store recommended distro is Ubuntu.
- Now you can run WSL as a standalone app also you can run WSL inside Power Shell by just writing WSL in shell

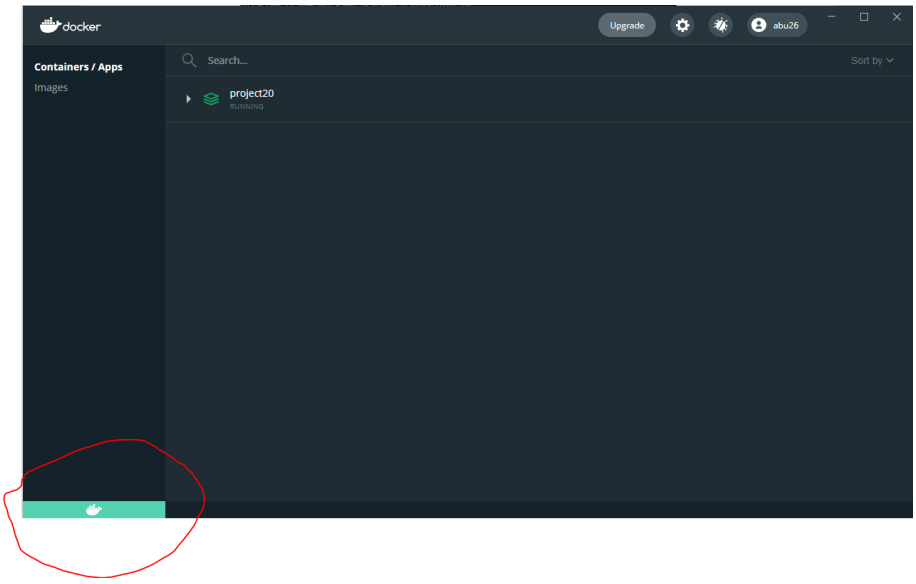


Installing Docker-Desktop for windows

- To install docker desktop setup for windows from the following link https://www.docker.com/products/docker-desktop

Setting up Docker-Desktop windows

- After installing the docker desktop for windows open the docker desktop and go-to settings-> Resources -> WSL Integration and enable the Ubuntu distro.
- Sign up on Docker hub and sign in to your account in docker desktop using your docker hub credentials.
- To check if docker is running correctly the docker desktop will indicate the aqua color otherwise there will be issue. e.g. It can be the WSL integration issue or WSL version



- Sometimes docker is not installed properly but docker-desktop will show you the aqua color in docker app to check it more properly open WSL or Power Shell in Power Shell firstly type WSL to start the WSL and then type following command **docker** It will show the bunch of commands

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Abubakar> wsl
bakar@DESKTOP-A9OKB8R:/mnt/c/Users/Abubakar$ docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/home/bakar/.docker")
  -c, --context string     Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and
                           default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/home/bakar/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/home/bakar/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/home/bakar/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit
```

Setting up docker environment for your project

To containerize the new project or already created project the first step is

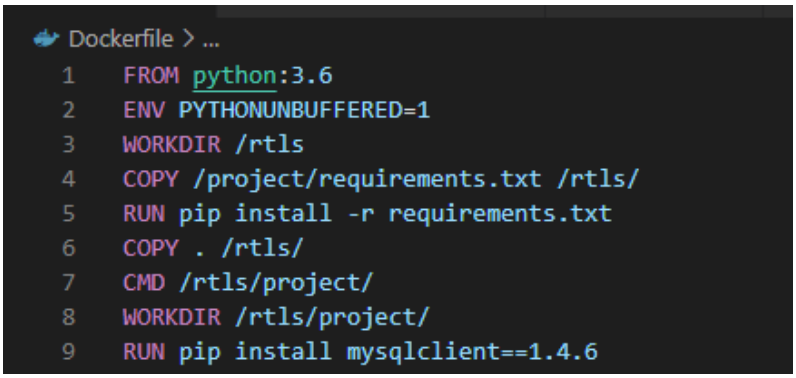
1. Create a **DOCKERFILE**
2. Create a **Docker-compose.yml** or **Docker-compose.yaml** file

Writing a Docker file

DockerFile is responsible to install the project and its dependencies inside the docker container or it can also used to copy the contents of the project folder existing in your local machine to docker container. Dockerfile can be created inside the root of your project folder. To create a dockerfile there are few steps.

1. Open the project in the visual studio code or any other editor and click on the add new file option and name that file as **Dockerfile** and VS-Code will automatically detect it and give it docker logo automatically. The file appears like this.

2. After creating docker file we have to start writing the requirements of our project inside dockerfile which will then install them inside the docker container. We will start from the base requirement so in our case we are installing a Django project which is python web framework so we started with **From python: 3.6** which is telling the python is base requirement and specifically mentioning its version 3.6 which we need. The figure given below is our project dockerfile stating all the requirements.



3. There are specific keywords / commands used to perform different tasks the purple keywords mentioned in the above figure are docker specific keywords.
4. In **ENV** it means it is creating a environment variables for python inside our container
5. **WORKDIR** is setting up our working directory by the name of rtls.
6. **COPY** is used to copy all the files from our local project directory to our container directory so copy will copy the requirements.txt file located in our project folder in local machine to our container rtls directory.
7. **RUN** will execute the pip command to install all the requirements inside our docker container project directory.
8. **COPY ./rtls/** is copying the complete local machine project directory to our rtls folder created inside container.
9. **CMD** command is used to execute the container so when we start our project or we build it it will give us error that it is not finding manage.py file because our current directory will be rtls and our project directory will be rtls/project which have manage.py file so CMD will start the container and navigate us to rtls/project folder.
10. **WORKDIR** will set this as our working directory.
11. **RUN** is used to install the mysqlclient compatible to our project.

Writing Docker-compose.yml file

In docker compose file we add all the dependencies and their credentials for example if we are running a web service on which port it should be running etc. The docker compose file can be of .yml or .yaml docker support both of these extensions.

1. In our docker-compose file we have **version: "3.9"** stated which is our docker compose version.
2. So, we want to use following services to support our project which are **Redis, MYSQL, Django**. They all are mentioned as under the services.
3. We are using redis as a **Web-socket service** so we first name it as web-socket it is treated as service and then start to add the credentials like image and port. For image we can pull the redis image from docker-hub also by mentioning it here docker can install it on the time of building the docker image. We also mentioned the ports etc.
4. Same with other services but in our web service we have two extra mentions **Volume** and **Depends on**. In volume we have mentioned our root directory it allocates space of our project in rtls folder in container and in depends on our project is dependent on database service.

docker-compose.yml

```yaml
version: "3.9"
services:
  web-socket:
    container_name: project20_web-socket_1
    image: redis:5.0.9
    ports:
      - '6379:6379'
  db:
    container_name: Project20_db_1
    image: mysql:5.7
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=rtls_database
      - MYSQL_USER=root
      - MYSQL_PASSWORD=root
    ports:
      - '3306:3306'
  web:
    container_name: Project20_web_1
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/rtls
    ports:
      - "8000:8000"
    depends_on:
      - db
```

Executing and Running the project

After writing both Docker and docker compose file we need to build this image to build Image run following command

Docker-compose build

After successful build completion to run our project run the following command

Docker-compose up

```
PS D:\docs\Ren-testing\project2.0> docker-compose up
Starting project20_web-socket_1 ... done
Starting Project20_db_1         ... done
Starting Project20_web_1        ... done
```

This states that our container is working perfectly on command line and you will also see this on your docker-desktop