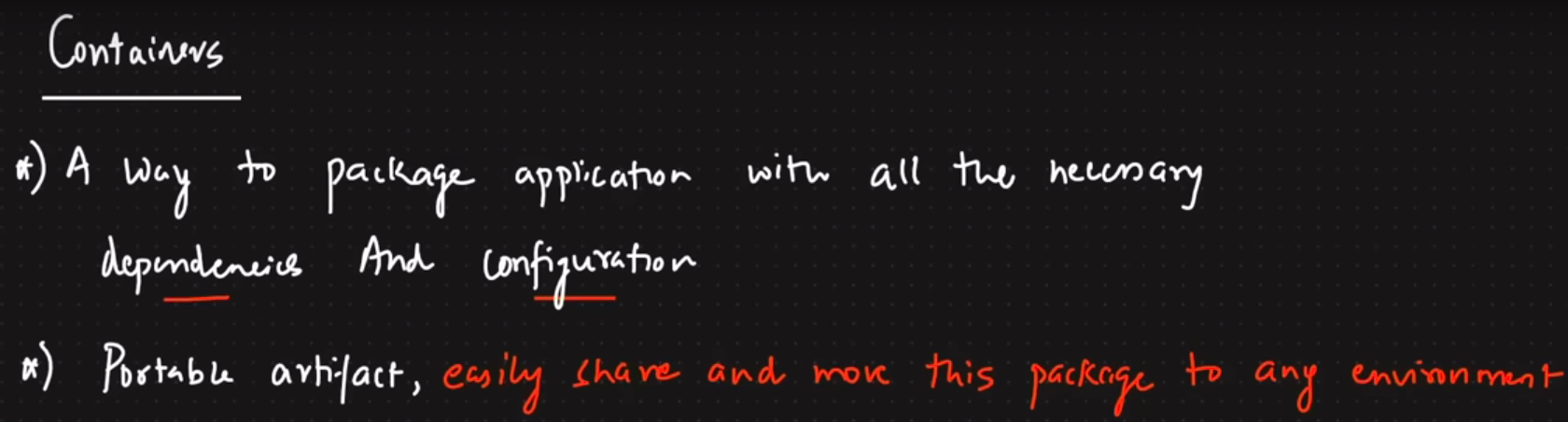
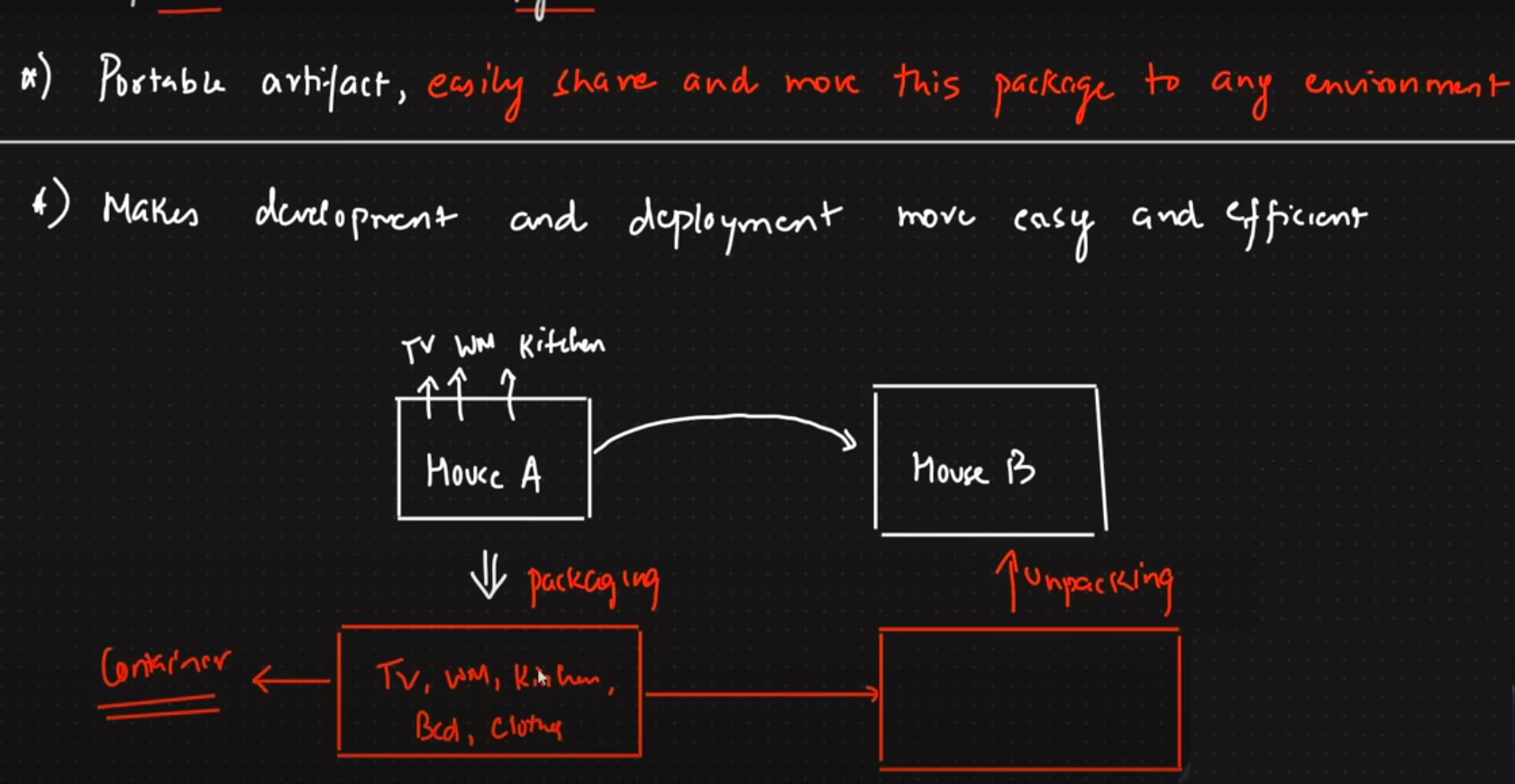
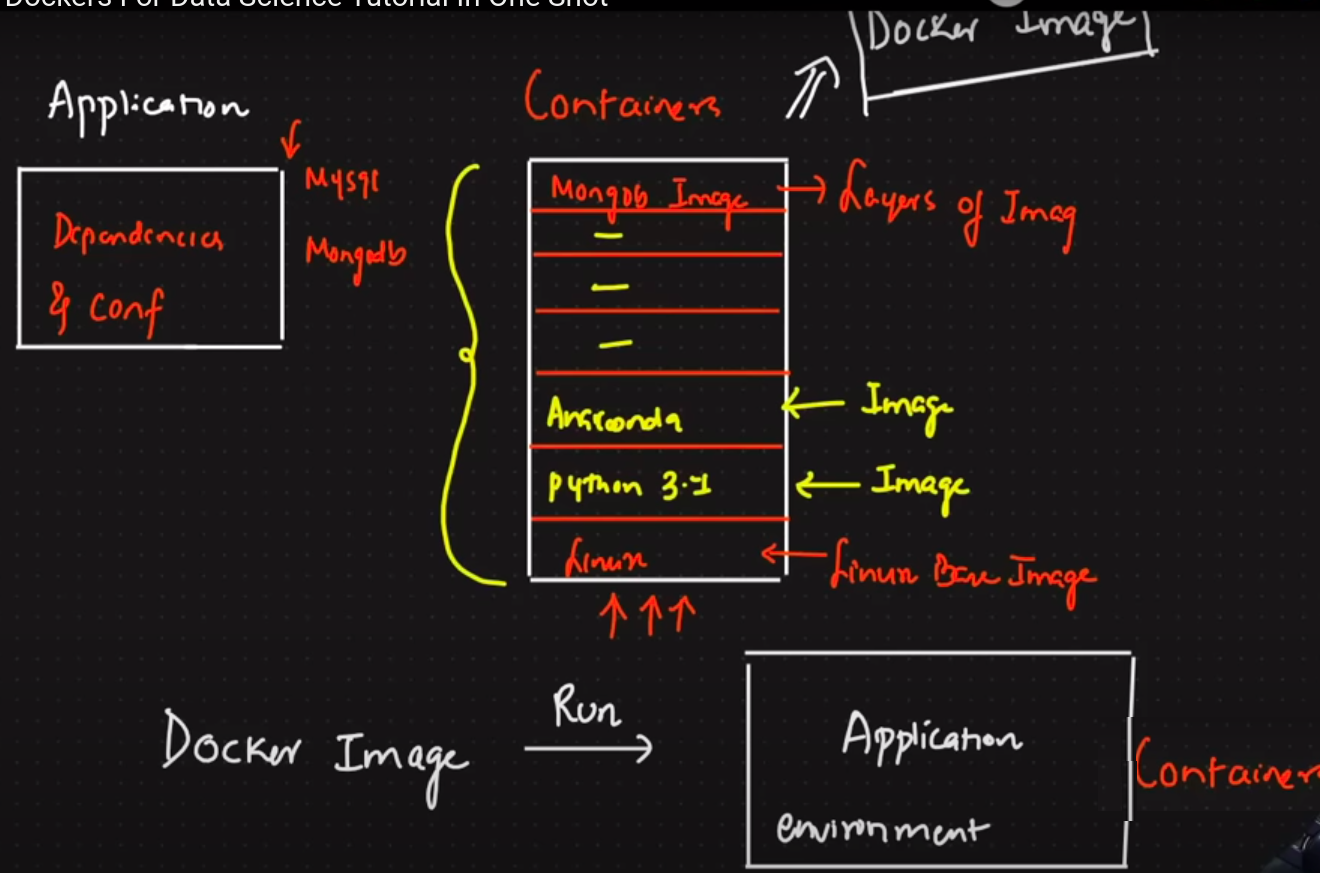
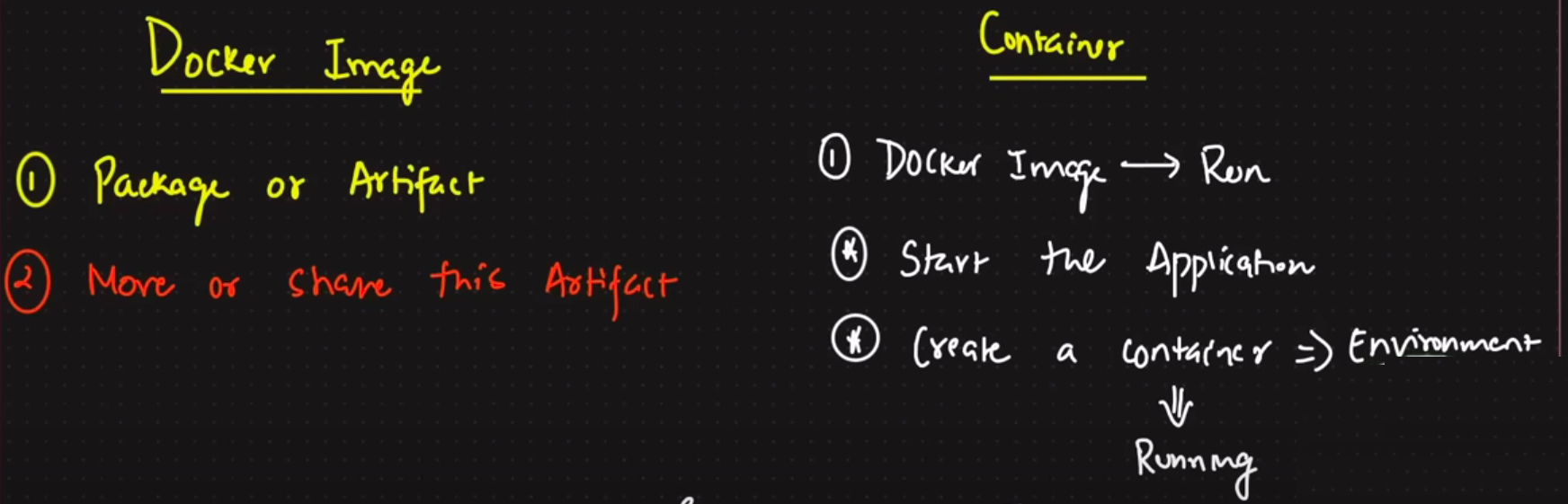
**Docker**

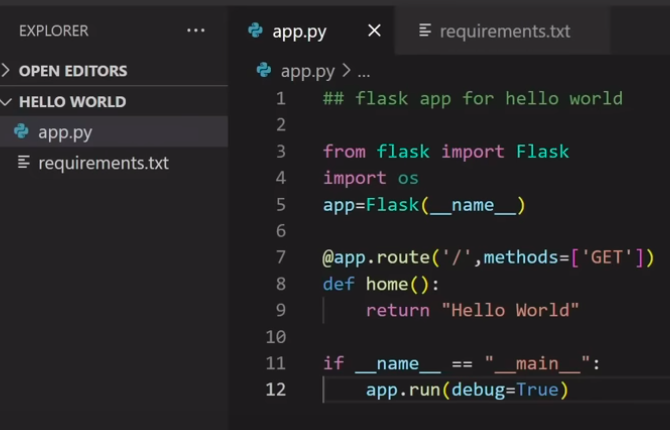






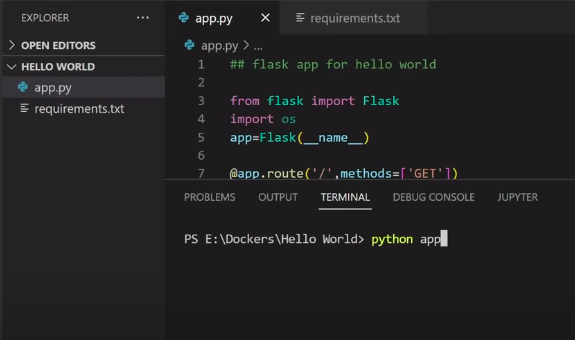


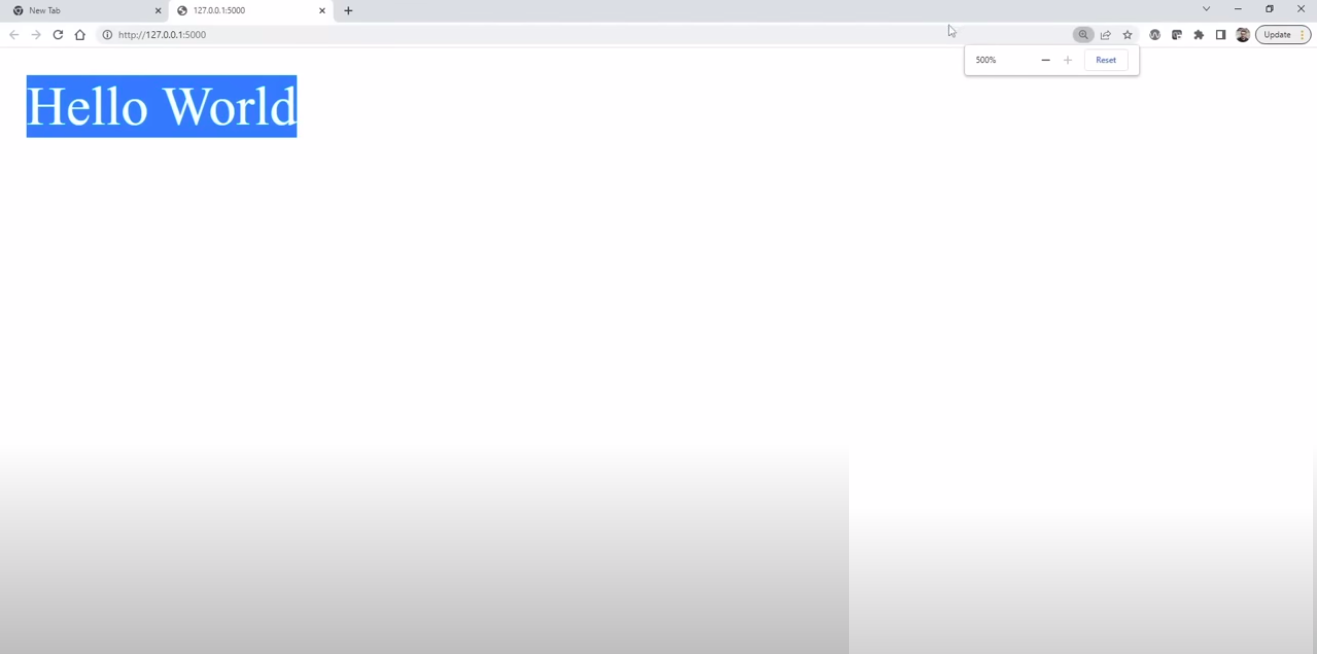
1. Now we are going to make the docker container for the following code as shown bellow, and for it we will create an docker image and than we will make it upload on the dockerhub

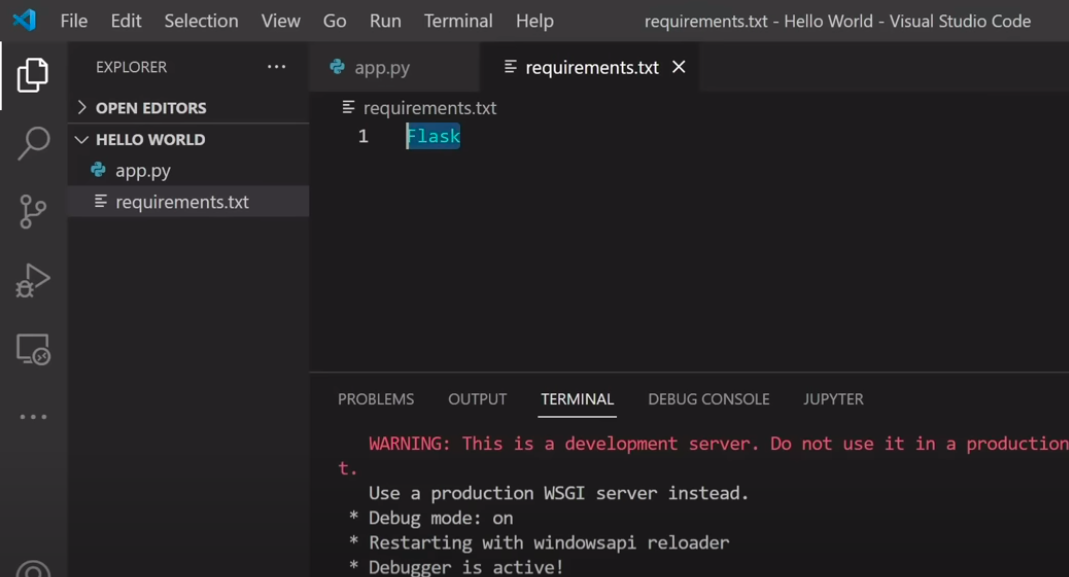


→ The above code is of the flask application and in the requirement.txt there is the Flak written in it.

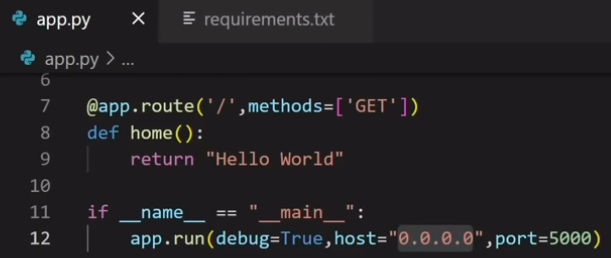
1. The first step will be to open the cmd/terminal in which we are working and in it we will first write the docker **docker -v** and if it returns back the version of the docker than docker is installed otherwise not.
2. Now let’s check how our app is running before making an docker image.



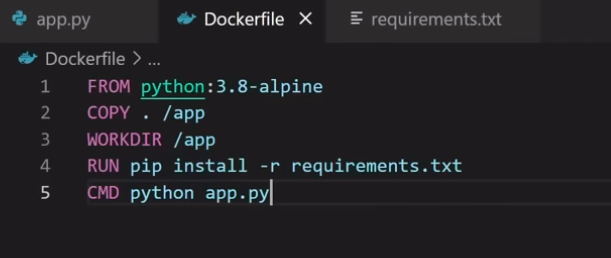




1. We can also make some additional change in the code for specifying the port number and host number, As shown below.



1. Than for making docker image we will specify the rules of the image in the **Dockerfile**, as shown below:



→ Here the **FROM <text>** indicates the main main linux layer os of the container which is compulsory to mention, And the alpine is the lightest os in the linux, so we will use it.

→ Than there is the **COPY . / app** command, and here the **. indicates** the space from where we want to copy all the things and than put into the location which is written after the space i.e. **/app**

→ All the files from the directory where this file is located will be copied to the app folder of the image

→ Than we will define the **WORKDIR** location, which is the location where we have to do work in the docker image

→ Make sure there is the space between the keyword and the location or name.

→ Than comes the **RUN** command by help of which we will run the **pip install -r requirements.txt** and which will install all the libraries written in the requirements.txt in the docker image

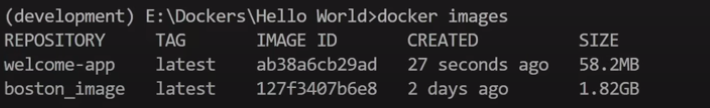
→ And the last command is on the **CMD python app.py** , this means that after setuping the docker image by help of the above commands the command written in the CMD will run the python app.py command to tun the flask app written in the app.py.

1. Now for building the image we will write the **docker build -t <new-app-image-name> .** , where the . indicates the location of the current directory from where it have to consider the Dockerfile



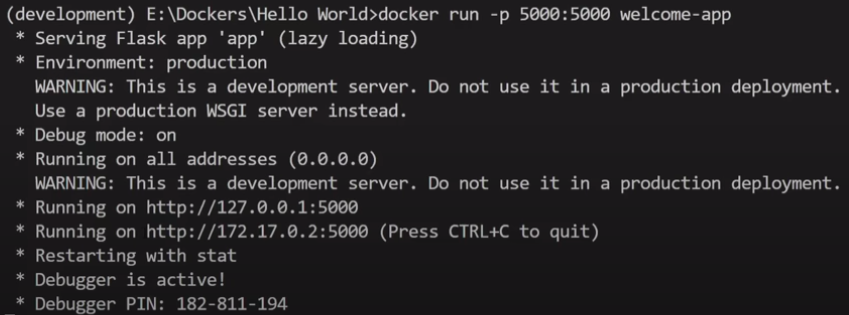
→ After writing the above code the docker image will be made

1. Now to check if the docker image is been made or not we will use the **docker images** command, as shown below:



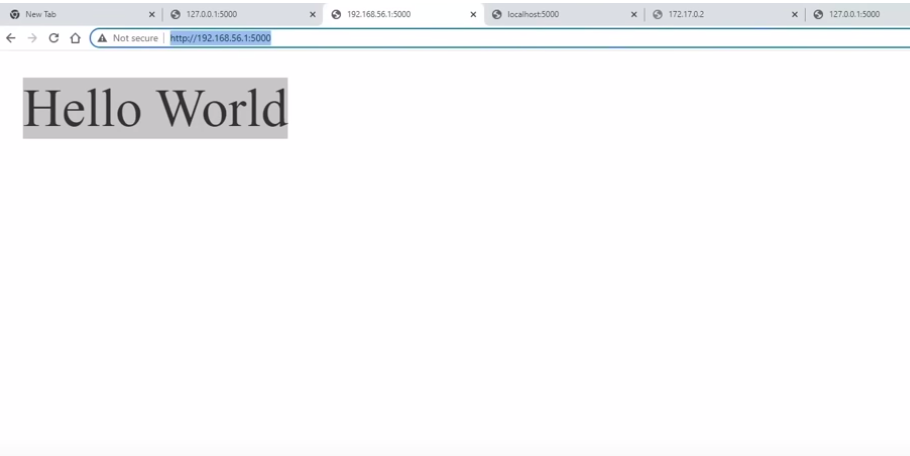
→ And the image named welcome-app is been made successfully.

1. Now to run an image that is already build we will use the **docker run -p 5000:5000 welcome-app command**, Where the original command is **docker run -p <port-container:port-app> <app-name>** where the first port indicates the port of the container on which app will be there and the second port is on the flask app that we have written, The command is as shown below:

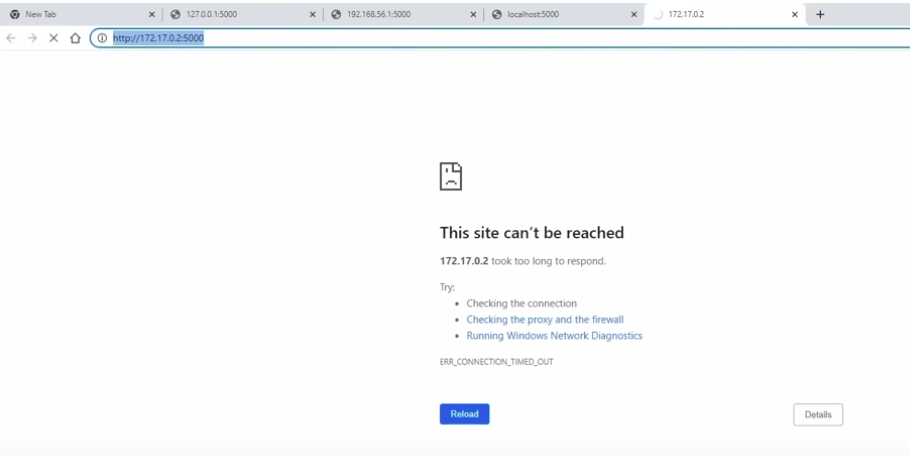


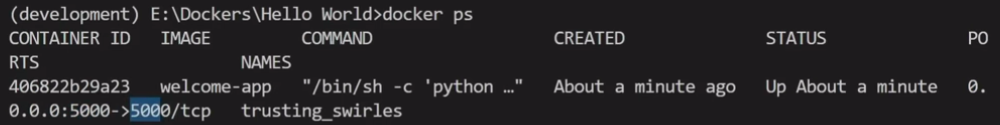
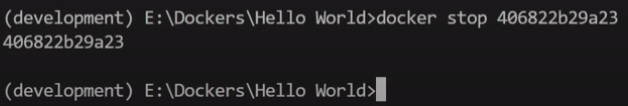
→ Now the thing to note that if we try to run the first link than the web-app will run, but rather if we try to run the app by using the second link than the app run will not be there as the second link is as per the container port while the first link is as per the our local machine.

**⇒ Running the first link in the browser:**

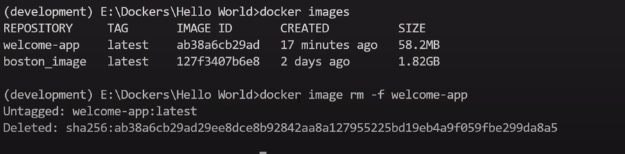


**⇒ Running the second link in the browser:**

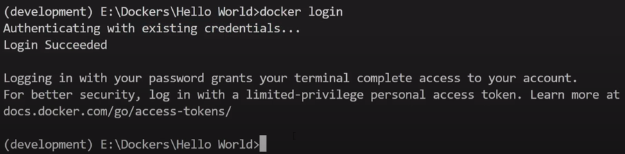


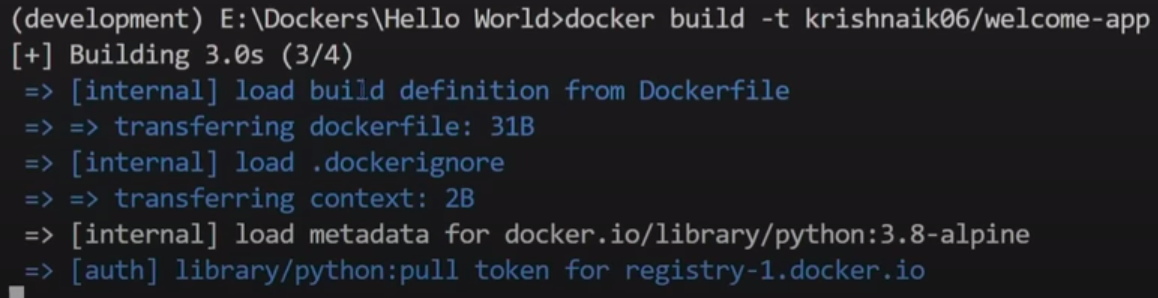
1. For checking the active docker container we will use the **docker ps** command as shown below, 
2. Now to stop the docker container we will use the docker stop <RTS/Docker id>, Example docker stop 406822, And for each container there will be unique container id, The command in the terminal is as shown below: 

→ If this command give return the container id than the container is made stop.

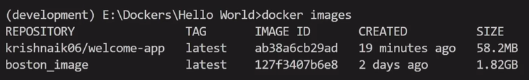
1. For deleting or removing an container we will use the docker image rm -f <app-name> command, the illustration is as shown below: 



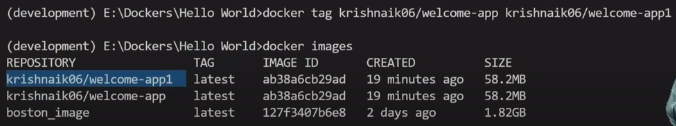
1. Now lets see how we can push our docker image to docker hub
   1. First write the docker login to login in the dockerhub, as shown below: 
   2. Than we have to build our docker image, but make sure we give the name of the docker image as per the naming convention of the dockerhub, Example <username-of-user>/<new-app/image-name>



→ And than the image will be made:

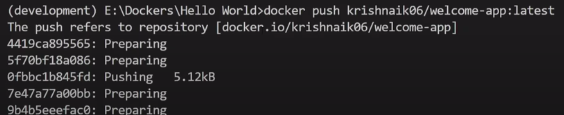


⇒ Now if we have an image and we want to upload that image only to dockerhub but the name of the image is not correct than we can make the another copy of that image by help of the **docker tag <initial-image-name> <new-image-name>**, as shown below:



* 1. Now we will push the image that we have made by help of the **docker push <image-name>:<TAG>**,for this example the image name is krishnaik06/welcome-app1 and the tag is latest

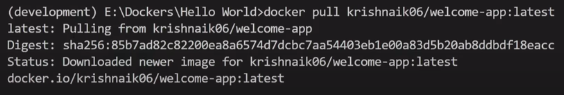
→ As we make the changes in the current image the tag keeps on changing from 0.0 to 1.0 than to 2.0 and much more.



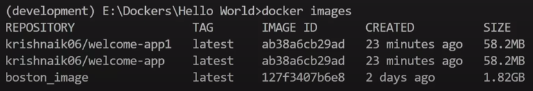
* 1. And that’s done the image is uploaded on the user’s dockerhub account

1. Now for pulling the image in the system first make sure that there is no image present of it, and if present tha use the **docker rm -f command**
   1. Now for pulling we will use the below command:

**docker pull <image-name>:<tag-name>**



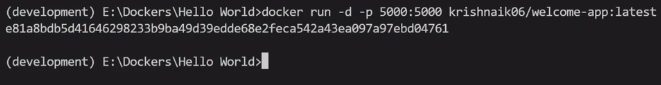
* 1. And by doing this we will get the our own copy of that image, we can check that by help of the **docker images** command



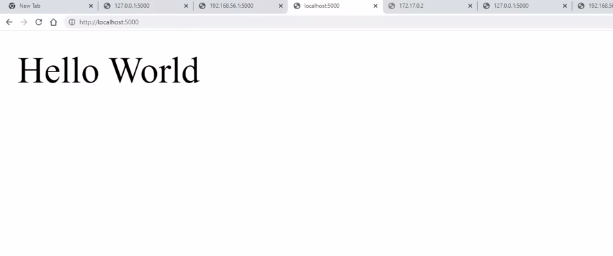
* 1. Than for running the image we will use the **command docker run -d - p <image-name>**

→ We will assign the port 5000 of local system to the port 5000 of the container

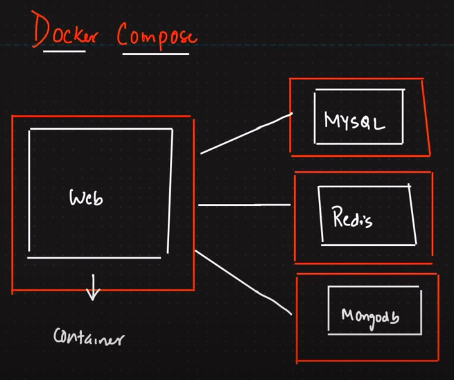
→ Where the -d indicates to open the new terminal in the detached form



1. And if we check if it is running or not than,



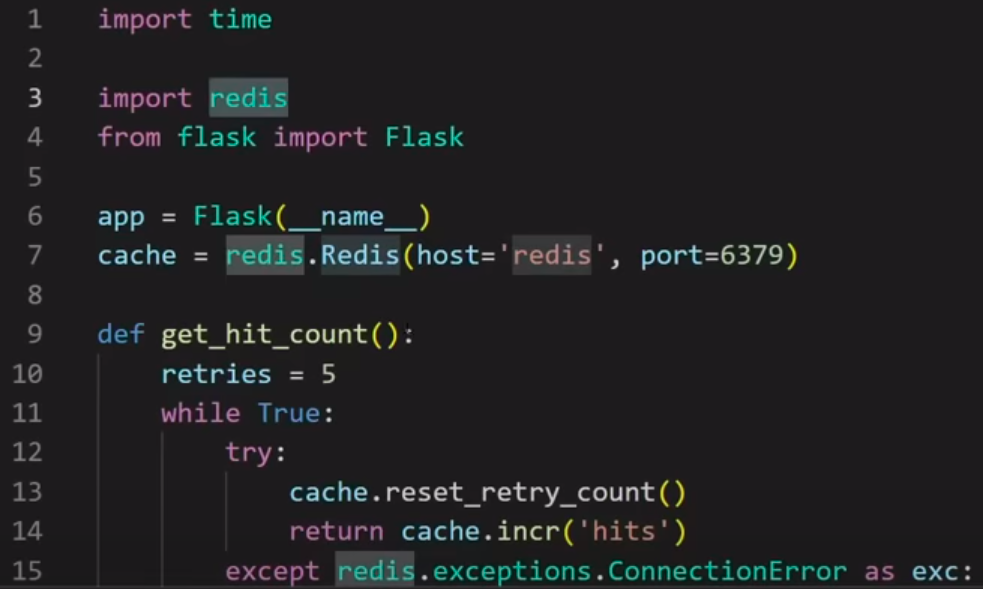
1. We can do many things with the docker we can install various of the databases and many other things with the help of the docker.
2. Docker Compose

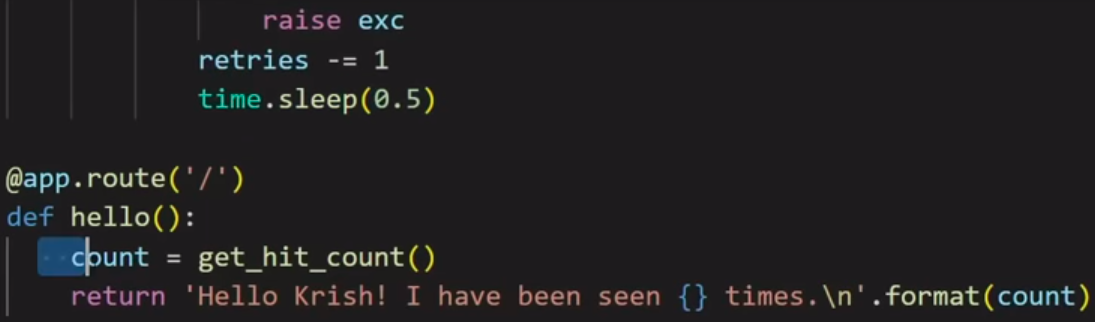
****

**→** Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application’s services. Then, with a single command, you create and start all the services from your configuration.

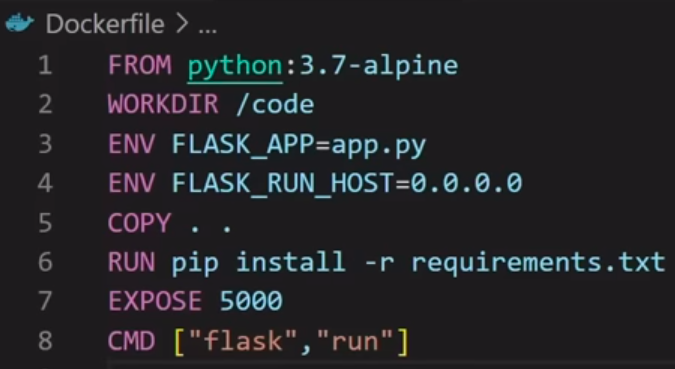
→ What the docker compose does is it run all the container that we require in an project under the one container which is the docker compose container.

1. Now for it we will use an another project which use the flask as well as the redis database, the code for that project is show as below:





1. Now the Dockerfile would be like:



→ Now the new thing over here is that we have used the ENV in this Dockerfile which is the environment variable, And we can use the different environment variables and more details about it can be get from the dockerhub documentation of that image.

→ Here the environment variables are used to set the FLAK\_APP and FLASK\_RUN\_HOST.

→ Than the FROM python:3.7-alpine is the base linux layer of the container

→ WORKDIR is the place in the container where the code will be executed

→ **COPY . .** will copy the codes from one directory where we are to the another in the container that is /code

→ Than comes the RUN that will run the command after the RUN keyword, that is used to start install the packages that are mentioned in the requirements.txt

→ Than we will write the port number after the EXPORT keyword

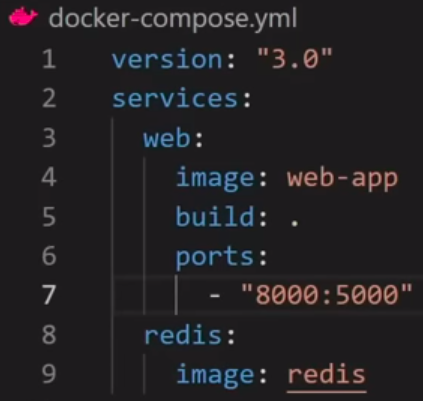
→ And last comes the CMD keyword in which we will write the commands required to start out app in the container

1. Main part comes now that is the **docker-compose.yml**

→ While writing this file make sure that the indent spacing is proper through out the file

→ And under th services: part we will write hte web: and the redis:

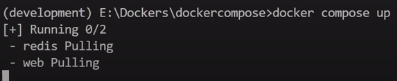
→ Under the web: we will specify the configurations of the flaks web app and under the redis: we will specify the configuration of the redis, as shown in the figure below:



1. Now suppose we want to also use the mysql in the project than the setup of it can be made as:



1. And for running the docker-composed container we will write the **docker compose up** command:

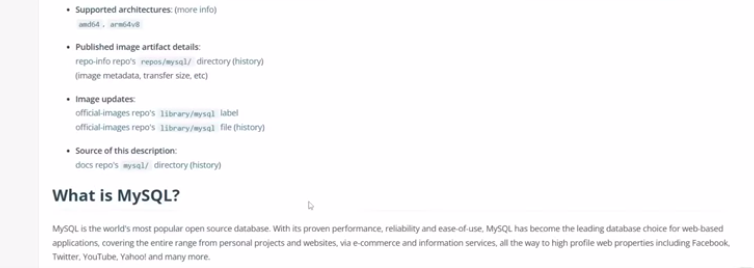


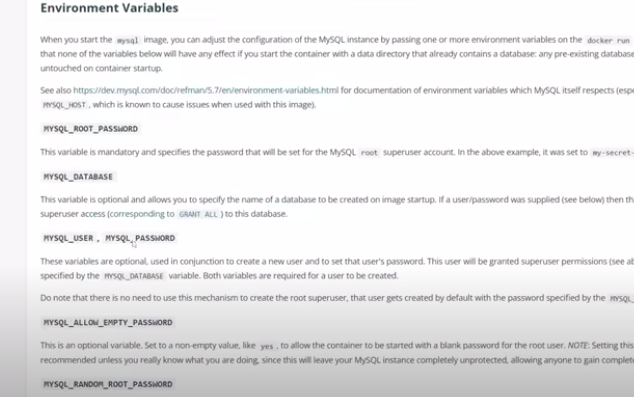
→ And to stop the composed container we will write **docker stop compose**

→Make sure that the compose can only be used as the name if we have used the docker-compose.yml file

1. Now for getting the names of environment variables we need the documentation, and the sample idea of checking the dockerhub documentation is as shown below:







1. Docker volume
2. Docker Svamy
3. Fwe
4. Fr
5. Few
6. Fe
7. Fe
8. Wfwe
9. Fe
10. Fwe
11. Wef
12. Fe