DOCKER NOTE BY RAJAN DEVKOTA
-r.devkota.98@gmail.com

INTRODUCTION

What is Docker?

- Docker is an open platform for developing, shipping, and running applications.
- Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
- By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly
- reduce the delay between writing code and running it in production.

Where will docker be useful?

- **For Fast, consistent delivery of your applications.**
- **Responsive deployment and scaling**
    - ☐ Docker containers can run on a developer's local laptop, on physical or virtual machines in a data center, on cloud providers, or in a mixture of environments.
    - ☐ Docker's portability and lightweight nature also make it easy to dynamically manage workloads.

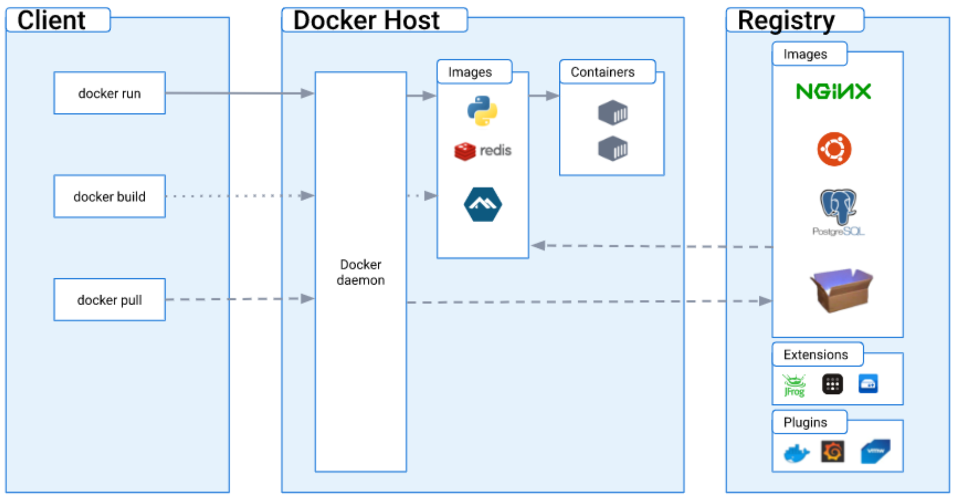
- **Running more workloads on the same hardware**
    - ☐ It provides a viable, cost-effective alternative to hypervisor-based virtual machines, so you can use more of your compute capacity to achieve your business goals.

    - ☐ Docker is perfect for high-density environments and for small and medium deployments where you need to do more with fewer resources.

Docker Architecture

- This is the Docker Architecture which shows how individual components communicate with each other.
- The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers.
- The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon(cloud).
- The Docker client and daemon communicate using a REST API.

- Another Docker client is Docker Compose, which lets you work with applications consisting of a set of containers(multiple containers).
- Now, let's have a look at some of the individual components that we have defined above.



The Docker daemon

- ☐ The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers,networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

- ☐ The Docker client (docker) is the primary way that many Docker users interact with Docker.
- ☐ When you use commands such as docker run , the client sends these commands to dockerd , which carries them out.
- ☐ The Docker client can communicate with more than one daemon.

Docker Desktop

- ☐ Docker Desktop is an easy-to-install application for your Mac, Windows or Linux environment that enables you to build and share containerized applications and microservices.

Docker registries

- ☐ A Docker registry stores Docker images.
- ☐ Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.
- ☐ There are different cloud platforms where we can store the docker image. For example, in Amazon Web Services we have Amazon Elastic Container Registry, in Microsoft Azure, we have Azure Container Registry, etc.

Docker Images

- ☐ A Docker image is a file used to execute code in a Docker container
- ☐ Docker images act as a set of instructions to build a Docker container, like a template. Docker images also act as the starting point when using Docker. Often, an image is based on another image, with some additional customization.
- ☐ To build your own image, you create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it.

Practical:

Step 1: Create a directory named DemoDocker

    ‘mkdir DemoDocker`
    `cd DemoDocker`

Step 2: Create a virtual Environment

    `conda  create -n demo_env`

Step3: Activate virtual environment

    `conda  activate demo_env`

Step4: Open the directory on code editor

    `code  ..`

Step5: Create a python file named app.py

```python
from flask import Flask


app = Flask(__name__)

@app.route('/')
def hello_word():
   return 'Hello World'

if __name__ == '__main__':
   app.run(host='0.0.0.0', port=8000)
```

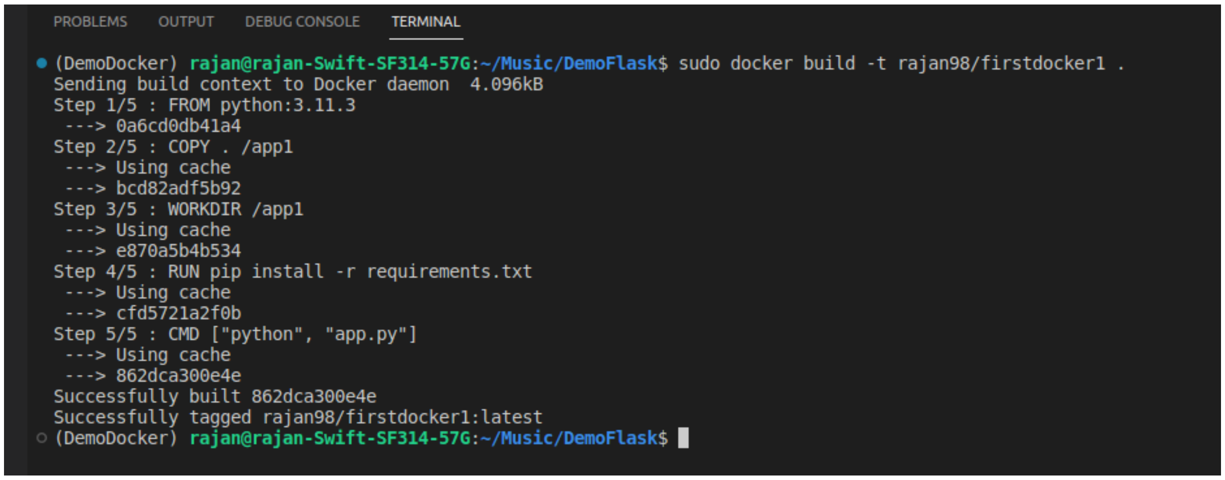
Step 6: Create requirement.txt file

    `pip freeze ->requirements.txt`

Step7: Create one file Name Dockerfile:

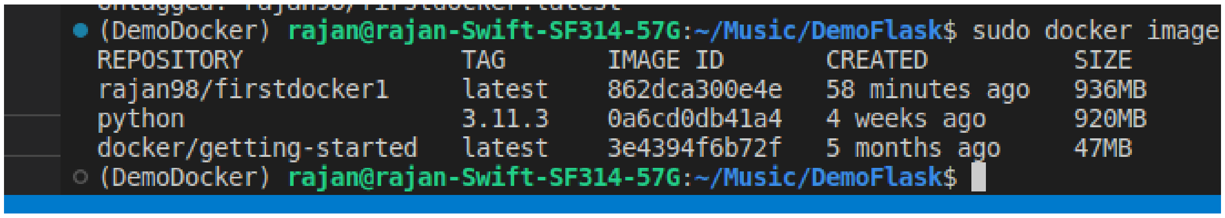
```
FROM python:3.11.3
COPY . /app1
WORKDIR /app1
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

Step7: Now build the docker image
`sudo docker  build -t rajan98/firstdocker . `



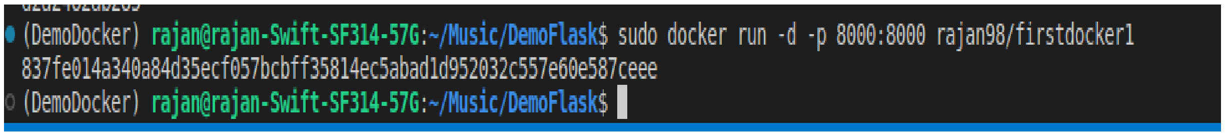
```
(DemoDocker) rajan@rajan-Swift-SF314-57G:~/Music/DemoFlask$ sudo docker build -t rajan98/firstdocker1 .
Sending build context to Docker daemon  4.096kB
Step 1/5 : FROM python:3.11.3
 ---> 0a6cd0db41a4
Step 2/5 : COPY . /app1
 ---> Using cache
 ---> bcd82adf5b92
Step 3/5 : WORKDIR /app1
 ---> Using cache
 ---> e870a5b4b534
Step 4/5 : RUN pip install -r requirements.txt
 ---> Using cache
 ---> cfd5721a2f0b
Step 5/5 : CMD ["python", "app.py"]
 ---> Using cache
 ---> 862dca300e4e
Successfully built 862dca300e4e
Successfully tagged rajan98/firstdocker1:latest
(DemoDocker) rajan@rajan-Swift-SF314-57G:~/Music/DemoFlask$
```

Step8: Docker images has been created
`sudo docker images`



```
(DemoDocker) rajan@rajan-Swift-SF314-57G:~/Music/DemoFlask$ sudo docker image
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
rajan98/firstdocker1    latest    862dca300e4e   58 minutes ago   936MB
python                  3.11.3    0a6cd0db41a4   4 weeks ago      920MB
docker/getting-started  latest    3e4394f6b72f   5 months ago     47MB
(DemoDocker) rajan@rajan-Swift-SF314-57G:~/Music/DemoFlask$
```

Step9: Run the docker image
`sudo  docker build -d -p 8000:8000 rajan98/docker1`



```
(DemoDocker) rajan@rajan-Swift-SF314-57G:~/Music/DemoFlask$ sudo docker run -d -p 8000:8000 rajan98/firstdocker1
837fe014a340a84d35ecf057bcbff35814ec5abad1d952032c557e60e587ceee
(DemoDocker) rajan@rajan-Swift-SF314-57G:~/Music/DemoFlask$
```

Congratulations, you have ran your first docker image.

You can push your docker image to docker hub. You first create an account in docker hub. Make sure you have created an image in the format of username/image_name.

`sudo docker push rajan98/firstdocker1:latest`

But before this you have to provide login credentials if this is your first time docker.

`docker login’

After pushing, you can pull the image and run it on any other devices.

Other basic commands of Docker:

1. The `docker ps` command is used to list the currently running Docker containers.
2. The `docker images` will display all the docker images.
3. The `docker stop id` will stop the currently running docer containers with specified docker id.
4. The `docker rmi -f docker_image_name` will delete the docker image.