Docker

• Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.
• Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.
• Build once, run anywhere .
• A clean, safe, hygienic and portable runtime environment for your app.
• No worries about missing dependencies, packages and other pain points during subsequent deployments.
• Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying
• Eliminate inconsistencies between development, test, production, and customer environments

How to install Docker :

1. To update apt, issue the command: $ **sudo apt update**

2. Mostly the docker package is available in all linux systems. So by below command you can install docker.

```
$ sudo apt-get install docker.io
```

3.If you find any problem in the above command you will have to run an installation script provided by Docker.

```
$ sudo apt-get install wget
$ wget -qO- https://get.docker.com/ | sh
```

4.If you want to be able to run Docker containers as your user, not only as root, you should add yourself to the group called docker using the following command.

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

Then you'll want to log out and then log back in for the changes to take effect.

How to create a docker image :

Docker gives you the capability to create your own Docker images, and it can be done with the help of Docker Files. A Docker File is a simple text file with instructions on how to build your images.

Create a file called Docker File and edit that by writing appropriate commands to build your image. Please note that the name of the file has to be "Dockerfile" with "D" as capital.

Ex: **Creating a docker image for a django application**

1.Create a requirements.txt file and write all dependencies and packages needed to install to create image of that django application .

--->requirements.txt

```
#For this application requirements are
django
django-allauth
django-extensions
django-model-utils
django-rest-auth
django-webpack-loader
djangorestframework
djangorestframework-jwt
```

2.Create a Dockerfile in the root directory of your web application and in that Dockerfile write the appropriate commands to build a container running a Django web application server. The vast majority of Dockerfiles will begin by referencing a base image provided by Docker.Take “django:python3-onbuild” as base image .
(you can also take python as base image) .And then set up directory structures, environment variables, download dependencies, and many other standard tasks before finally executing the process which will run your web application.

```
# Dockerfile
# FROM directive instructing base image to build upon
  FROM django:python3-onbuild
```

3. Copy all the files which are responsible for web application into image.

```
   #COPY - copies the specified files into image (container) .
 #here djangoapp copied is the full folder which contains djangoapplication and requirements.txt

        COPY djangoapp djangoapp
```

4. Change the working directory to djangoapp where it has requirements.txt and manage.py

```
        #WORKDIR - change the working directory
        WORKDIR  djangoapp
```

5. Now install all the packages in requirements.txt

```
        #RUN - runs the command given.
        RUN pip install requirements.txt
```

6. Give a port to run the application

```
         EXPOSE 9123
```

7. Give commands to run your django application.

```
    # CMD-is the command the container executes by default when you launch theimage.
        CMD ["python", "manage.py", "migrate"]
        CMD ["python", "manage.py", "runserver", "0.0.0.0:9123"]
```

8.Now the Dockerfile looks like

```
      #Dockerfile
       FROM django:python3-onbuild
       COPY djangoapp djangoapp
       WORKDIR  djangoapp
       RUN pip install requirements.txt
       EXPOSE 9123
       CMD ["python", "manage.py", "migrate"]
       CMD ["python", "manage.py", "runserver", "0.0.0.0:9123"]
```

9. Now go to directory where Dockerfile is present and run the below command.

```
$ docker build -t <imagename:tag>  .
```

(At last the dot '.' is important to mention that the Dockerfile is present in working directory)

10. Now the image was built .Run the command "docker images" , you will see your image.

11. Now run the image as container.

```
$ docker run -p 9123:9123 <imagename:tag>
```

Check localhost:9123 in your browser, which shows your application.

How to share image :

Multiple team members may wish to share images. Images can be in production, under development or under test.
Docker Hub is a repository where images can be stored and shared (it is public).
You can also use your private repository .
Each image is tagged to allow versioning .
Any image can be "pulled" to any host (with appropriate credentials).
Tagging as "latest" allows updates to be propagated.

(a) Push your images to docker hub

(i)Create a account in dockerhub (hub.docker.com). And remember your username and password.

(ii)Now go to terminal and run the command "**docker login**"
And then enter your username and password to login.

(iii)Then tag your image to a new name such that it should be in the format like
{username}/{imagename:tag}

```
$ docker tag  your_imagename:tag   your_username/imagename:tag
```

(iv)Now push your image to docker hub.

```
$ docker push  your_username/imagename:tag
```

(v) Now the docker hub has your image.

(iv) You can also push your same image with different tag.So that it will push Into same image repository with different tag. And then the repository contains different images with same imagename and different tags.You call pull the image by differentiating the tag.

(vii) Now the person who needs to pull that image can pull with the below Command.

```
$ docker  pull  <usernameofimageholder>/<imagename:tag>
```

(b) Push your images to private (local) repository

You might have the need to have your own private repositories. You may not want to host the repositories on Docker Hub. For this, there is a repository container itself from Docker.

(i)Use the Docker run command to download the private registry. This can be done using the following command.

```
$ docker pull registry:2
```

‘registry’ is the container managed by Docker which can be used to host private Repositories.

(ii) And it is better to mount a volume which keeps a backup of all your pushed Images. So that if the registry container is stopped it will restore all your pushed images when ever it restarts.

```
$ mkdir registry_backup
$ cd registry_backup
```

and run the registry image along with mounting.

```
$   docker run -d -p  9999:5000 -v  $(pwd)/:/var/lib/registry registry:2
```

Let’s do a ‘docker ps’ to see that the registry container is indeed running
Now your private registry setup is done.

(iii)Now let’s tag one of our existing images (let’s take centos) so that we can push it to our local repository.

```
$ docker tag centos  localhost:9999/centos
```

(iv)Now let’s use the Docker push command to push the repository to our private Repository.

```
$ docker push localhost:9999/centos
```

(v)You can share your private registry image to others also.To do that go to the file /etc/docker/daemon.json and add this line.

```
{
        "insecure-registries" : [ "<your_IP_address>:9999" ]
}
```

And in the system from where you need to pull this private image also add the above line in /etc/docker/daemon.json in its system.(The ipaddress which is mentioning here should be the image holder’s ipaddress.)

(vi) After that you can pull that private image with the following command.

```
$ docker pull <ipaddressofimageholder>:9999/centos
```

(c) Share image via .tar file

(i) Sometimes I want to save a docker image and then use it on another computer without going through the hassle of uploading it to dockerhub or private registry

(ii) Docker images aren’t really stored as files that you can just grab and move to another computer .But you can save and then load the images like this:

If you want to save the image as a tar archive, using docker save -o:

```
$ docker save -o  abcd.tar  <image_name:tag>
```

(iii) Then copy the adcd.tar file into another computer and load there :

```
$ docker load abcd.tar
```

(iv)Run “ docker images ” . It will show up in your docker images list .