# Docker Commands

**Commands to install docker :**

- fist command : curl -fsSL https://get.docker.com -o get-docker.sh second commad : sh get-docker.sh
- adding User to docker:(sudo will make run it as admin)
    - sudo usermod -aG docker azureuser
- adding a webserver with name nginx1
    - docker run -d --name nginx1 -P nginx
    - docker run -d --name apache1 -P httpd
- list of images:
    - docker images
- dot cloud became docker after releasing docker as open source.

**24/may :**

**Docker Setup :**

- do ssh and connect to linux server
- enter this cmd to download docker to your machine
    - `curl -fsSL https://get.docker.com -o get-docker.sh` click on enter
    - in next line enter this `sh get-docker.sh`
- Now enter below commad to provide the Admin priviliges for the user, this command will add your user to the docker group it will provide full permission for your current user
    - `sudo usermod -aG docker azureuser`
    - by doing above azureuser will get full previliges to docker
- Now enter this command `exit`to exit from your mchine and reconnect to machine using ssh username@vmname, for effective chnages.
- To check just enter `docker info` if it shows both server version and client version then you got full access to it.

```
exit
shh azureuser@ip_address
```

- go to docker hub web site you will find all the feautures that you can run in container ex: mysql, nginx,httpd ..etc
- nginx is **webserver**, httpd is **apache server**
- go to vm again and enter below cmds
    - `docker run -d --name ngnix1 -P nginx`
    - this will download and run the nginx webserver, name of this will be **nginx1**

    - `docker run -d --name apache1 -P httpd`
    - this will download and run the httpd webserver name of this will be **httpd1**

- if you try to install ngnix and apache in one machine(linux machine) by one shot you will get error as "80 port is already in use"

- now enter below command to list out the containers

```
docker ps
docker container ls -a
```

- now copy your vm public ip and check the nginx webserver port number, and httpd port number and enter in browser you will get nginx and httpd are running
    - `public_ip:32768`
    - you can run as many containers you want do below for awareness
    -

```
docker run -d --name apache2 -P httpd
docker run -d --name apache3 -P httpd
```

- to check status of containers usage
    - `docker stats`
- any os to run on any machine minimun of 4gb+ will be occupied, but in container it is very less
    - `docker pull alpine`
    - alpine is linux operating system
    - to see all images we have `docker images`

**25/may :**

- on any machine when you install docker you will get two components
    - docker engine or docker demon
    - docker clinet
- to speak with docker engine you would need docker client(cli for docker engine)
- when you open clint and write "docker run nginx" indrectly you are saying i want nginx container, when you did this request will go to the docker demon
- docker demon will have local repository, it will try to ask the repository "do we have nginx image?" if no it will check in online repository
- over the internet we have "Docker hub in technical we call Registry"
- if ngnix image is not present it will pull(download the image from registry) and store to local repository
- now it will create container form nginx image

-

```
* To create container we need image
* using the same image you can create n number of conatiners
```

```
  * images are stored and made available using registries and docker is a
default registry.
```

- container is a runtime in an isolated area which gets
  - virtual cpu
  - virtual ram
  - v disk(contents will be filled from image)
  - v nic
- **scenario :**

```
when you try to install any OS in your laptop it will take approx 1 hr, where as
when you go to any company laptop the repair team will take 10 min to install OS
how ??

A . At first time they will also install manually taking 1hr time later, now we
know everything wil be stored in disk, these people will take **Ghost
image(compressed form of disk) of that disk**. whenever a new person comes they
will use this image to install OS becasuse it has all applications pre installed
init.

image means - compressed form of disk
```

- **get docker cheetsheet to learn commands**
- **Install Doker :**
- docker can be installed on :
  - windows
  - Mac
  - linux
- we will install in linux most of the time
- Follow above given container creation steps and create a container
- if you don't know commads
  - just type `docker run --help` it will suggest you the all usable commands for run
- else browse for cmd based on your requirement
- create a a vm and elivate the priviliges for your user and create a container
- enter this cmd `docker container run hello-world`
  - when you hit on enter it will show "unable to find image locally and pulling from registry"
- run the same command again to confirm whether the image is downloaded or not."this time it won't show any not available message as above"
- download ngnix `docker container run -d nginx`
- to list out images use this cmd `docker image ls or docker ps -all`

```
how to give a name to container? what happens if you don't give name to a
container?
A. if you don't give name to a container docker will provide a random name to it ,
if you want to provide name ``` --name <name of container>
```

**26/may :**

- every docker image has a tag,if you are not passing tag name it will take latest `<repository>:tag`

```
nginx  => nginx:<tag>
       => ngnix:perl
jenkins => jenkins/jenkins:jdk17

    jenkins => jenins:latest
```

**Container life cycle**

- go to docker play ground create a new vm and connect to it from powershell
- go to vm and do below

  ```
  `docker image ls` check for any images present already
  `docker image pull --help`
  ```

- execute this `docker pull ngnix`
  - observe the output you will identify that it is stating that it is using latest tag
- now do this `docker pull nginx:perl`
- now do `docker image ls` to check images, find out `Tags` and if you try to observe the image id
  - if two images are said to be same when their image id's are same
- now we will ceate a container :
  - do this `docker container create --name nginx1 -P nginx`
  - this will create a container for you check by using cmd `docker container ls --all`
  - To run container:
    - `docker container start nginx1`
    - do this to check whether container is runnig `docker container ls --all`
    - obseve the status
  - To pause container:
    - `docker container pause nginx1`
    - check `docker container ls --all`
    - check the status
  - To stop the container:
    - `docker conatainer stop nginx1`
    - check `docker container ls --all`
    - check the status

    - To start back again:
        - `docker container start nginx1`
        - check `docker container ls --all`
        - check the status
    - To remove container:
        - first stop the container and remove it
        - `docker container stop nginx1`
        - `docker container rm nginx1`

- creating container and starting is not recomended we will use below command to create and run the container in one go

    - `docker container run -d -P --name nginx2 nginx:perl`
    - stop using `docker conatainer stop nginx2`
    - start back `docker container start nginx2`
    - try removing container when it is runnig `docker container rm nginx2`

- Now create a new container

    - `docker container run --name test1 hello-world`
    - check state `docker container ls -a` check the status `not runnig`
    - we need to understand the thing whenever you try to run the container it doesn't mean it will run container it may or might not be runnig

- create a new vm from azure/aws

    - ssh to vm
    - do `ifconfig` to know about nic card
    - do `sudo apt update && sudo apt install net-tools -y`
    - do `ifconfig`
    - you will see two network interfaces
        - eth0 - private ip
        - lo(loop back) : "local host" - 127.0.0.1
    - do `htop` for task manager in linux
    - in linux for every process there will be process id , if process id is 1 then that is the process which started at first.
    - check the users list `cat /etc/passwd
    - we will focus on last two users `ubuntu and lxd`
    - check groups `cat /etc/group` take last two groups `_chrony and ubuntu`
    - check storage `sudo lsblk` find an xvda:8gb disk

- now we will try to install docker

    - use `curl https://.....`
    - after installing docker do `if config` and check the network interfaces **`one more wil be added`**
    - check for the users you find no imapct, check for group `you will find a new group` **docker**

- we will run a container now

    - sudo usermod -aG docker

    - now do this `docker container run -d --name nginx1 nginx`
    - `docker container ls -a` find container is runnig
    - now we go inside of container:
        - `docker container exec -it nginx1 /bin/bash`
        - now your path will change to `root@somenumber:/#`
        - do `apt update && apt install net-tools`
        - do `ifconfig`
        - you will see two network interfaces `lo and eth0`
        - check for runnig process `apt install htop` and do `htop`
        - do `lsblk` you can find same 8gb storage it is usig from root machine
        - do `cat /etc/group` you will find different groups here when compared to root machine
        - check for users `cat /etc/passwd` you will find `nobody`
        - go to root folder `cat /` and do `ls`
        - do `apt install tree -y`
        - execute `tree` you will find many things here, we will navigate to one folder `cd /etc/` and do `ls`
        - do `exit`
        - it will come to root machine execute `tree` you won't find anything because whatever you do inside contair there won't be any impact on container
        - install tree on root machine `sudo apt install tree -y`
        - go to `ls/etc` you will find the same

- now create one more container:

    - `docker container run -d -P --name apache httpd`
    - go inside container and create bash `docker container exec -it apache /bin/bash`
    - do `apt update && apt install htop -y (you shouldn't install softwares inside containers for understanding you had to do)
    - do `htop` and observe all the process running about one thing `httpd` unlike root machine

**28/may :**

```
* consider a monolith architechture of any application running in server,it can
accept at a limit of 5000 users, if you want to extend you need to run the
application in one more server which obiviously called as scaling, now what if
users are incresing more and more..
* consider i am runnig an e commerse application on server which do so many things
at one runtime like payment, adding products to carts there will so many services
running at a time.
* here is our question  who ever the user loged will use all the services? some
will just browse and leave some will add products to cart and leave...etc
* no user will use all the services. to solve this we diveded our application into
small serivices
    * catalog service
    * identity sevice
    * cart service
    * payment service
    * notification service .......etc
```

```
* these all component services we will run in each containers when ever the need comes to access any serivce the container will run

* one more advantage is we can create differnt serives in different languages not by sticking to one language like catalog service in C#,identity sevice in java,cart service in python ..etc
* as definition of microservices, if i found catalog service is written in python is working more efficiently than catalog service in C#, i will be able to remove it and add this to my container without taking more time
* there will be so many sevices like this more than 100, in that case to manage these we need a orchestrator such use kuberneties
```

**We will containarize applications now :**

- Ways to containarize :
  - figure the below manually
    - what is required to run the application
    - how to deploy application
    - what command to execute to start the application
    - on which port is appliation accesible
  - Try yo configuring application manually once *virtual machine
    - conatainer

**host spring pet-clinic app**

- requirees: *java 17
  - application
  - command to run the application `java -jar <path to spring petclinic.jar>`
  - it runs on port 8080

**setup virtual machine**

- create a linux vm in azure/aws
- this application runs on 8080 port in security
- setup java

```
sudo apt update
sudo apt install openjdk-17-jdk -y
```

- download spring petclinic file using `wget <link>` and do `ls` to check whether the file downloaded or not
- To run this application use this cmd `java -jar <path of file.jar>` if you are in current directory no need to mention path
- it will take a bit time and shows started, once it is started take public ip of vm and check the port and browse theough `ip:port` it should work
- go to docker playgroud and take a vm and open in powershell

**Run the app in container manually**

- go to docker hub and search for open jdk 17
- and check the image with jdk17 and download it `docker container run --name manualspc -p 32767:8080 amazoncorretto:17 /bin/bash`

```
docker container run --name myjavaapp -p 2000:8080 -it amazoncorretto:17 /bin/bash

create :
    docker container create -p 1000:8080 --name xyz -it amazoncorretto:17
Run:
    docker start xyz
get inside :
    docker exec -it xyz bash
after running the application to back back by still keeping the app run:
    ctrl + p + q
```

- this command will download the image and take you to the bash termial inside container
- to get java file download use this `curl -O <link>`
- start the app `java -jar <path of file.jar>`

**Creating image from container**

- creating image form container use cmd `docker container commit manualspc spc:frommanual`
- check `docker image ls`

**Docker file :** looks like

```
From amazoncorretto:17
RUN curl -O <link>
SNAPSHOT.jar
EXPOSE 8080
CMD ["java", "-jar","<file.jar>"]
```

- go to server and create a folder `mkdir test`, create a file inside it `vi Dockerfile` paste the above containarization code.
- create a docker image using above code `docker image -t spc:automated .`
- now docker will read the given instructiins and start building th image on its own
- check by `docker image ls`
- run the container using the image `docker container run --name myspc -p 32770:8080 -d spc:automated`
- if you want to run one more instance of image `docker container run --name myspc1 -p 32771:8080 -d spc:automated`
- check `docker stats` how much memory the conatiners are using you will find very less.

**Little Linux on managing files (using VIM) :**

```
* Creating files and saving them:
    * to create a newfile and open it `vim test.txt`
    * just click `i` to enter into edit mode
    * after writing the notes just click on `esc` and write `:wq` for saving the
file and back to the directory
    * dont want to save the file `:q!` without saving you will be pulled back to
directory
    * just want save and be there in file editor `:w`
```

**29/may**

- dockerile is an instruction based approach to create docker images.
- in this approach we create a file with name `Dockerfile`
- Docker file contains series of instructions

```
<instruction>:<value>
```

**Most widely used instructions**

- FROM : this instruction specifies the base image
- RUN : this instruction executes commands as part if base image building
- EXPOSE : this instruction specifies the ports to be exposed
- CMD : This instruction will have command that is used when container is started (it helps nothing in image building)
- LABEL : This instruction is used to add metadata

**Create EC2 vm in azure and do ssh from powershell**

- install docker `curl -fsSL https://get.docker.com -o install-docker.sh`
- in next line `sh install-docker.sh`
- add user `sudo usermod -aG docker ubuntu`
- exit `exit` and log back in
- try `dicker info` and check the client and server versions are able to view
- do `mkdir spring-petclinic`
- do `vi Dockerfile`
- inside the file add only `FROM : amazoncorretto:17` and save it.
- go inside the folder `cd /spring-petclinic` and do `docker image build -t trail:1.0 .`

```
docker image build -t <image-name>:<tag> .
* `.` represents current directory
```

- do `docker image ls` and check the message you will see `created = 5 weeks ago` which is a lie, we just created a image
- when you write `FROM : amazoncorretto:17` it means you are not building image but it means you are downloading and using it.
- do `docker image pull amazoncorretto:17` and check `docker image ls` you can see the same as `created 5 weeks ago`.

```
add below to our Dockerfile

`LABEL : author ="Aravindh"` # take amazoncorretto 17 as a base image
`LABEL : project = "it-learning"` # add metadata
`RUN curl -O https://khajareferenceapps.s3.ap-south-1.amazonaws.com/spring-petclinic-3.2.0-SNAPSHOT.jar` # download pet clinic application
`EXPOSE 8080` # expose 8080 port, as spring petclinic needs 8080
`CMD ["java", "-jar","spring petclicnic-3.2.0-snapshot.jar"] # command to start the application
```

- do `vi dockerfile` and write all the above lines and run it.
- go inside the folde by doing `cd` and do `docker image build -t spc:1.0 .`
- now do `docker image ls` and check `created : n secons ago` which shows it is created few sec back.
- lets try cresting the container with image

```
docker container run -d --name spc1 -P spc:1.0
```

**passing values while building the image**

- here comes the `ARG` keyword comes **1hr**