DOCKER INSTALLATION

Install packages to allow apt to use a repository over https

```
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Add docker's official GPG key

```
curl –fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

To setup the stable repository

```
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
$ sudo apt update
```

Install Docker Community Edition (CE)

```
$ apt-cache policy docker-ce
$ sudo apt install docker.ce
```

To verify Docker installation status

```
$ sudo systemctl status docker
$ sudo ps -ef | grep [d]ocker
```

To add username to the Docker group, to avoid using sudo to execute the docker command.

```
$ sudo usermod -aG docker ubuntu
```

To view Docker version and information

```
$ docker version
$ docker info
```

WORKING WITH DOCKER CONTAINERS

To search the docker hub for images

```
$ docker search searchWord
```

To run Tomcat Image in docker

```
$ docker container run --publish 80:8080 tomcat:jre8
```

What does the above command do?

1. *Dowloaded the Images form Docker Hub*
2. *Started new Container*
3. *Exposed port 80 on Host Machine*
4. *Routes traffic to the container port 80*
5. *Starts container by using CMD in docker file*

To run MySQL Image in docker

```
$ docker container run --publish 3306:3306 --env MYSQL_RANDOM_ROOT_PASSWORD=yes mysql
```

To stop Container fore-ground process

```
ctrl+c
```

To run container in Back-Ground or detach mode

```
$ docker container run --publish 80:8080 --detach tomcat:jre8
```

To name a docker Container

```
$ docker container run --publish 80:8080 --detach --name <name> tomcat:jre8
```

To list running Container

```
$ docker container ls
```

To stop docker Container

```
$ docker container stop <container_id>
```

To start docker Container

```
$ docker container start <container_id>
```

To bulk stop docker Containers

```
$ docker container stop $(docker container ls -a -q)
```

To list all running and stopped Containers

```
$ docker container ls -a
```

To mount a host volume in a Container

```
$ docker container run --publish 80:8080 -v /opt/myfolder:/hostfolder tomcat:jre8
```

To run a command in a Container in interactive mode (SSH a running container)

```
$ docker container exec -it <container id> bash
```

To commit the changes in the Container

```
$ docker commit -a "<author>" --m "message" <container name> <repository name:tag>
```

To remove Containers

```
$ docker container rm <space separated container ids>
```

To bulk remove docker Containers

```
$ docker container rm $(docker container ls -a -q)
```

To bulk remove all exited containers

```
$ docker container rm $(docker ps -a -q -f status=exited)
```

DEBUGGING THE DOCKER CONTAINERS

To verify Docker installation status

```
$ sudo systemctl status docker
$ sudo ps -ef | grep [d]ocker
$ docker version
$ docker info
```

To see logs of a specific Container

```
$ docker container logs <container_name>/<container_id>
```

To see process running inside the Containers

```
$ docker container top <container_id>
```

To list the details of Container config

```
$ docker container inspect<container_id>
```

To list the performance stats of all Containers

```
$ docker container stats
```

NETWORKING IN CONTAINERS

To start container to allow traffic from Port on Host machine

```
$ docker container run --p <host_port>:<docker_port> -d <image name>
```

To find the traffic and protocol on Container

```
$ docker container port <container_id>
```

To find Docker Container IP

```
$ docker container inspect --f '{{.NetworkSettings.IPAddress}}' <container_id>
```

To show all networks

```
$ docker network ls
```

To filter all bridge network

```
$ docker network ls -f driver=bridge
```

To find all Network IDs and Drivers

```
$ docker network ls --format "{{.ID}}: {{.Driver}}"
```

To inspect any Network (Returns information about one or more networks.)

```
$ docker network inspect
```

To create new Network on Host Machine

```
$ docker network create -d bridge <Network Name>
```

To connect Network with Container

```
$ docker network connect <Network Name> <Container Name>
```

To disconnect Network with Container

```
$ docker network disconnect <Network Name> <Container Name>
```

To remove Network

```
$ docker rm <Network Name>
```