

**Installation Steps:**

1. Setup an EC2 instance using Ubuntu 18

* select T2.micro and use all defaults
* Key = Name and Value = Docker-Engine
* Create a Security group
* Security group name = Docker-SG
* SSH / TCP / 22 / MyIP
* click on next and create a key pair
* Key pair name = docker-key
* download the .csv and click on Launch

1. Follow the Documentation on (https://docs.docker.com/engine/install/ubuntu/),

* Select Ubuntu
* Go to GitBash and SSH into the Docker EC2
* ssh –i Downloads/docker-key.pem ubuntu@public IP of the EC2
* sudo –i
* Updating the apt package index and installing packages to allow apt to use a repository:
* sudo apt-get update
* sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common -y

* Adding Docker’s official GPG key:

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

* Setup the Repository :

echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

* Install Docker Engine

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

* Verify that the Docker Engine installation is successful by running the hello-world image:

systemctl status docker

sudo docker run hello-world (docker ps –a would show you the dummy container)

* If you want to Add a User to the list of Docker users

sudo usermod –aG docker (username .e.g. Ubuntu)

* To check if the User exist in the group

sudo vim /etc/group

1. Commands:

(for Docker Images, visit: hub.docker.com)

For a List of Docker commands: <https://docs.docker.com/engine/reference/commandline/cli/>

* Open hub.docker.com to select an Image.
* SSH into your EC2 instance
* sudo -i
* docker pull nginx (Without Tag)
* docker pull nginx:mainline-alpine-perl (With Tag)
* docker images
* docker run - -name myweb –p 7090:80 –d nginx
* docker ps (shows Infos on the container like the I.D and port)
* Go to the Security group of your EC2 instance above and add inbound rule:

All traffic / All / All / My IP

click on save

* open your browser: (public IP of your EC2 instance) : 7090
* Nginx service will open, running from a container
* docker stop (container ID) or (container name)
* docker ps –a (to confirm if it is stopped)
* docker start (container ID) or (container name)
* cd (To leave root)
* ps –ef (To proof container is a process)
* cd /var/lib/docker/ (To proof container is a process running from a directory)
* ls
* cd containers/
* ls
* cd container ID/ (shows you data of your container)
* cd ..
* du –sh (container ID) (To prove it is an Image, check the size of the container)
* logout
* sudo –i
* docker ps
* docker exec (container ID) or (container name) (To attach to the process like a form of SSH)
* docker exec -it (container ID) or (container name) /bin/bash (Meaning run /bin/bash in the container and attach it …. You will become the Root user of the container)
* apt update
* apt install procps –y (Installs the process command)
* ps –ef (shows the process of the container only)
* exit (To leave the container process)
* docker stop nginx:mainline-alpine-perl
* docker rmi nginx:mainline-alpine-perl (Remove a Docker Image)
* Working with another Image
* docker pull ubuntu
* docker run –it ubuntu /bin/bash
* ps –ef
* exit (these kills the container instantly)
* docker ps –a
* docker rm (container ID) or (container name)

1. DOCKER LOGS:

In Docker, logs refer to the output generated by a container's processes. By default, Docker captures the standard output (stdout) and standard error (stderr) streams from the container's processes and stores them in a file.

These helps to troubleshoot the reason why a container wouldn’t start

* docker pull nginx
* docker run –P nginx (Shows you live logs… Control C ….. kills the process)
* docker run –d –P nginx (-d ….. hides the logs, -P …… port mapping)
* docker logs (container ID) or (container name)
* exit
* docker run –d –P mysql:5.7
* docker ps –a (mysql didn’t start)
* docker logs (container ID of mysql) or (container name) (To see a full report)
* To correct the MySQL setup, using pullig according to the Logs
* docker run –d –P –e MYSQL\_ROOT\_PASSWORD=mypass mysql:5.7 (-e is used to set a variable)

1. CONTAINER VOLUMES:

In Docker, a volume is a persistent data storage mechanism that allows data to be shared between containers and between the host system and containers. Volumes provide a way to store and manage data separately from the container's filesystem, making it easier to manage and backup data and making it possible to share data between containers or between the host system and containers.

STEPS TO CREATE DOCKER VOLUMES:

1. Using a Database framework, search for mysql in hub.docker.com
2. sudo -i
3. docker pull mysql:5.7
4. docker images (to findout the size of the directory)
5. docker inspect mysql:5.7 (To findout the port mapping, ID, name and many more in json format)
6. Binding with host directory volume
7. mkdir vprodbdata
8. docker run - -name vprodb –d –e MYSQL\_ROOT\_PASSWORD=secretpass –p 3030:3306 –v /home/ubuntu/vprodbdata:/var/lib/mysql mysql:5.7
9. docker ps (to check running status)
10. ls vprodbdata/
11. docker exec –it (container name ….. vprodb) /bin/bash (to login)
12. cd /var/lib/mysql
13. ls
14. you should find same data
15. I want to stop & delete the container to see if it remains in the system directory
16. docker stop vprodb
17. docker rm vprodb
18. ls
19. ls vprodbdata/ (the data should still be there)
20. docker volume (to show a list of commands for volume)
21. E.g. ….. docker volume create mydbdata
22. docker volume ls
23. docker run - -name vprodb –d –e MYSQL\_ROOT\_PASSWORD=secretpass –p 3030:3306 –v /mydbdata:/var/lib/mysql mysql:5.7
24. ls /var/lib/docker/volumes/mydbdata/\_data/
25. docker inspect (container name ….. vprodb) (To get info for the command below)
26. mysql -h (IP of the container) -u root -p(password)
27. you should be logged in
28. show databases;
29. Bye

CLEAN UP:

1. docker ps –a
2. docker stop (container name)
3. docker rm (container name)
4. docker rmi mysql:5.7