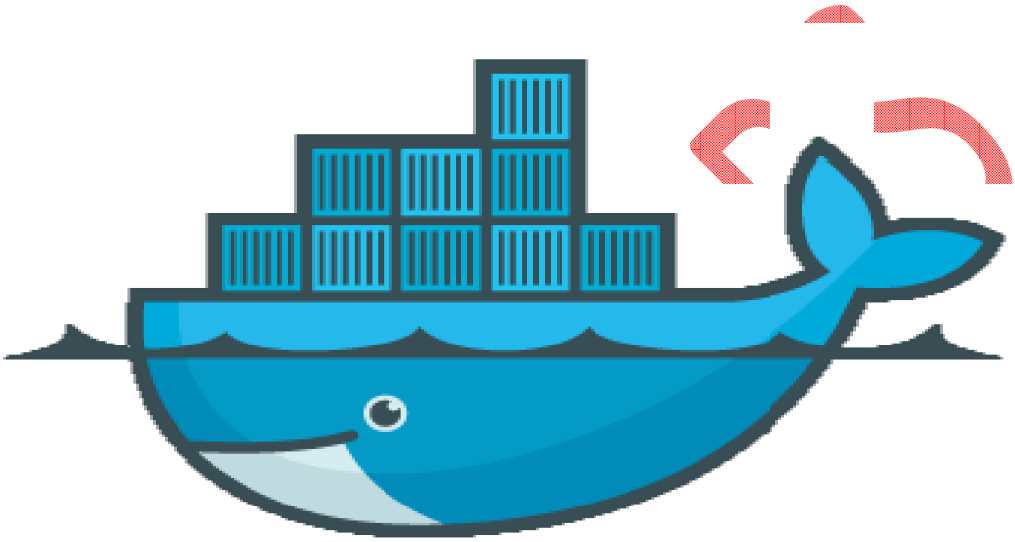


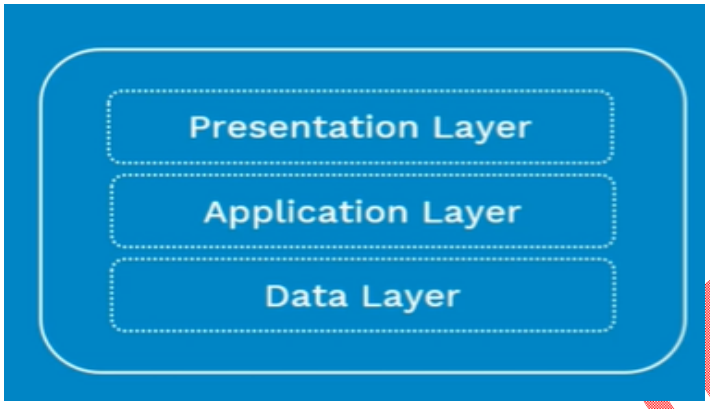
docker

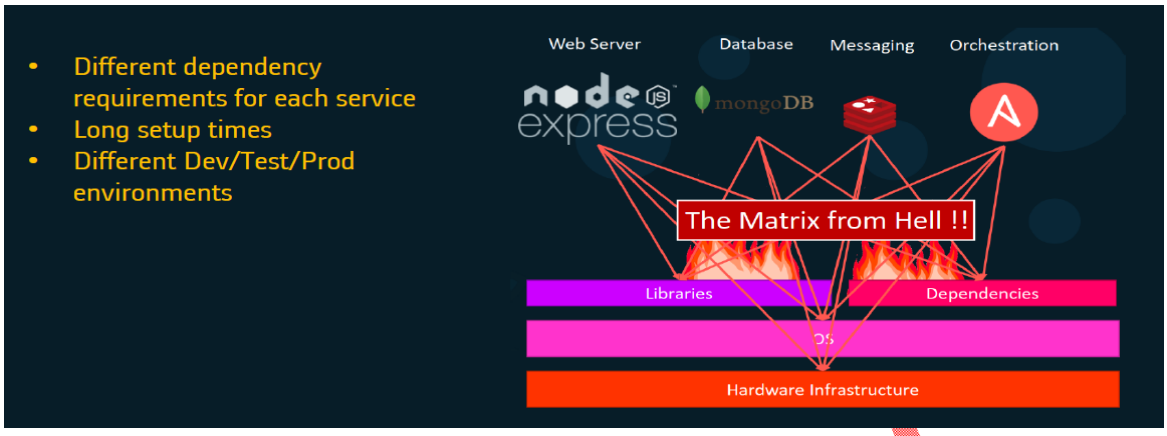
……. Prepared by PPNREDDY

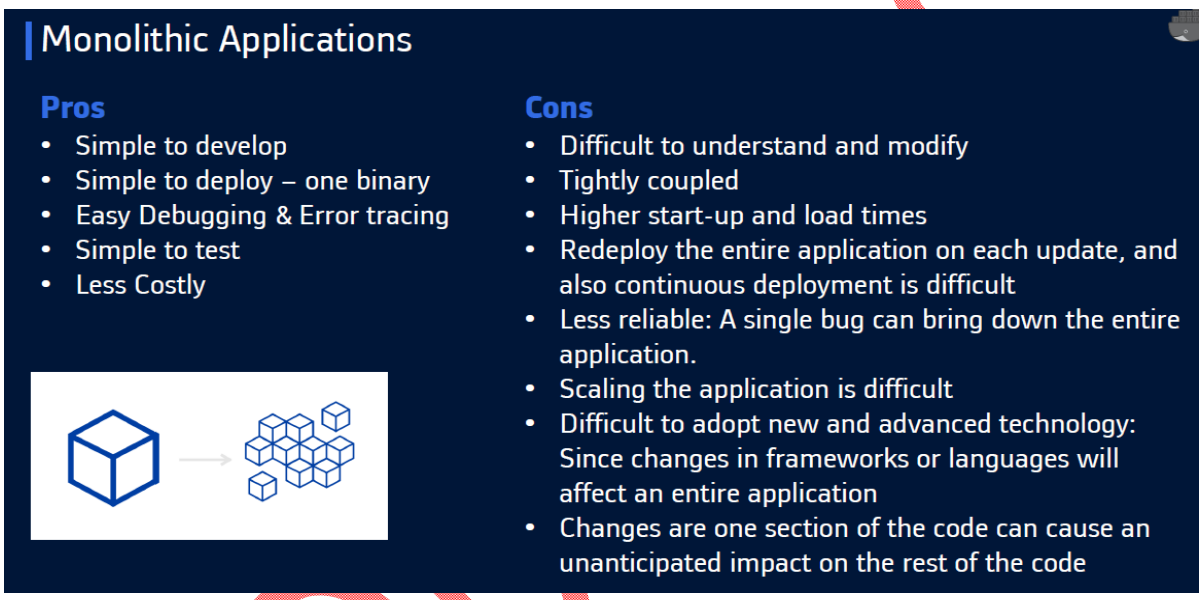
DOCKER

Monolithic application

The application divided into specific code OR technology based called Monolithic application.







Advantages

- Single unit of deployment.
- IDE Support.
- Simple to develop.
- Simple to test. For example you can implement end-to-end testing by simply launching the application and testing the UI with Selenium.
- Simple to deploy. You just have to copy the packaged application to a server.
- Simple to scale horizontally by running multiple copies behind a load balancer.
- Easy Debugging & Error tracing.
- Less Costly.

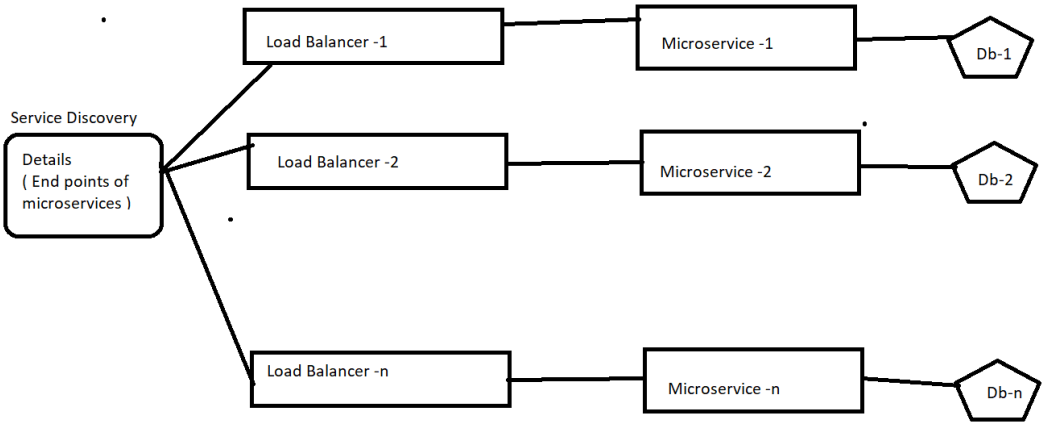
Disadvantages

- If application is large, we can't understand and it will take time to develop.
- Difficult to understand and modify.
- Tightly coupled.

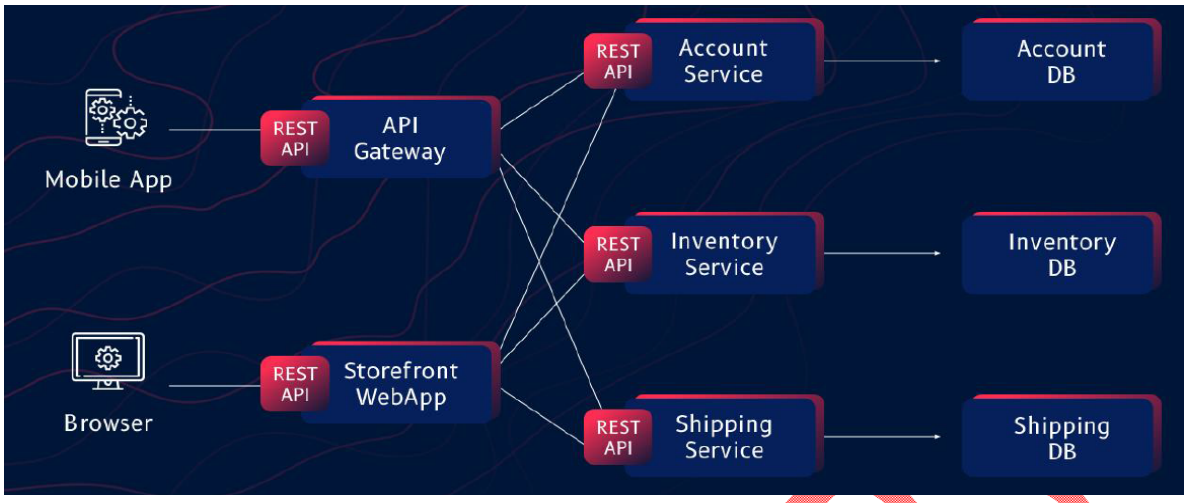
- Higher start-up and load times.
- Redeploy the entire application on each update, and also continuous deployment is difficult.
- Less reliable: A single bug can bring down the entire application.
- Scaling the application is difficult.
- Difficult to adopt new and advanced technology: Since changes in frameworks or languages will affect an entire application.
- Changes are one section of the code can cause an unanticipated impact on the rest of the code.
- Technology dependency on initial decision.
- Implemented using single development stack
- Frequent deployments are not practical.
- Need to scale entire application stack.
- This simple approach has a limitation in size and complexity.
- The size of the application can slow down the start-up time.
- You must redeploy the entire application on each update.

Microservices

Microservices is nothing but a sub application, it's acts as a standalone application and own database.



Service discovery is a service(registry) which details about all microservices.



Characteristics

Autonomous:

It's individual micro service it will not effect on other services if it's changed.

Specialized:

Each of the microservices are specialized and design with specific capabilities and specific business logic.

Advantages

- It improves developer productivity.
- Decoupled.
- It drastically improves fault tolerance of the applications.If one OR more microservices crash still customers can use the application.
- Microservices application has two developers, it improves the developer productivity.
- We can scale a specific microservice.
- We can choose different technology stack to different microservices.
- Easy deployment means it's small.
- Reusability code one more microservices.
- Ensures continuous delivery and deployment of large, complex applications.
- Better testing — since services are smaller and faster to test.
- Better deployments — each service can be deployed independently.
- No long-term commitment to technology – when developing a new service, you can start with a new technology stack.

Disadvantages

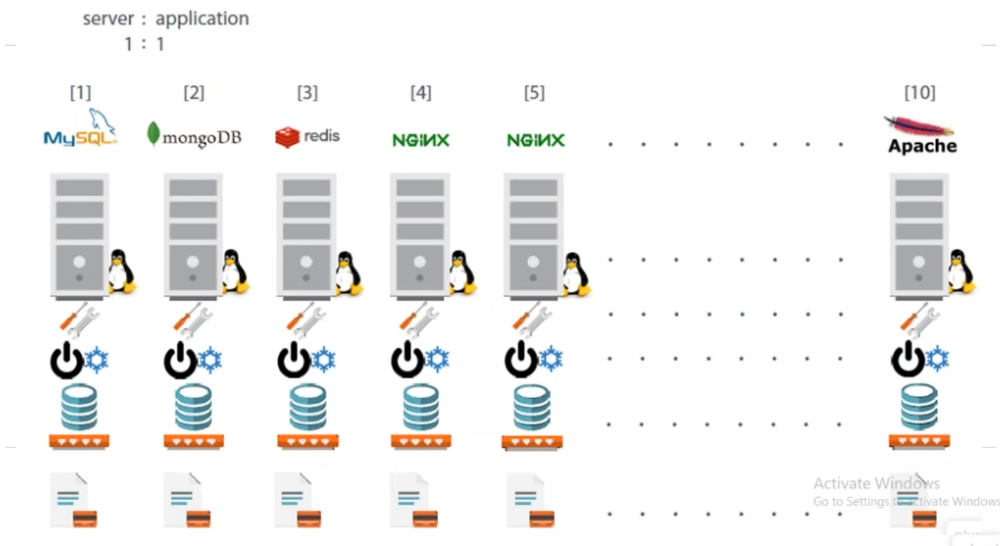
- Spawning (Microservices are keep on increasing day by day).
- Maintenance (Maintaining multiple microservices are difficult).
- Duplicate code.
- Network Performance.
- Slow bootup times of VMs
- Increased memory consumption
- Large OS footprint
- Initial Costs are very High and this type of architecture demands for proficiency in the skills of the developers.
- Testing is difficult and time-consuming because there is an additional complexity involved because of the distributed system.
- Deployment Complexity – there is added operational complexity of deploying and managing a system that contains various service types.

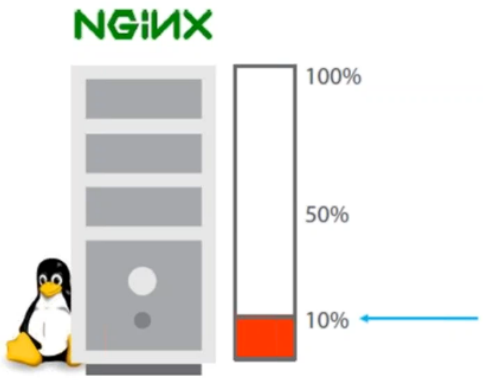
Note:

All microservices will run on Physical Servers, Virtual Machines and Containers.

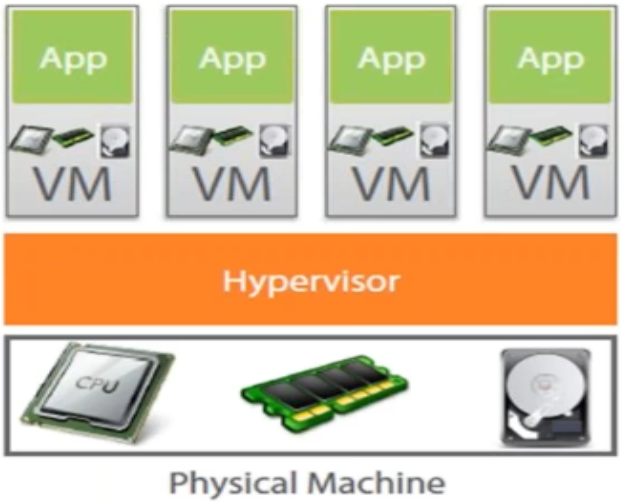
15 years back

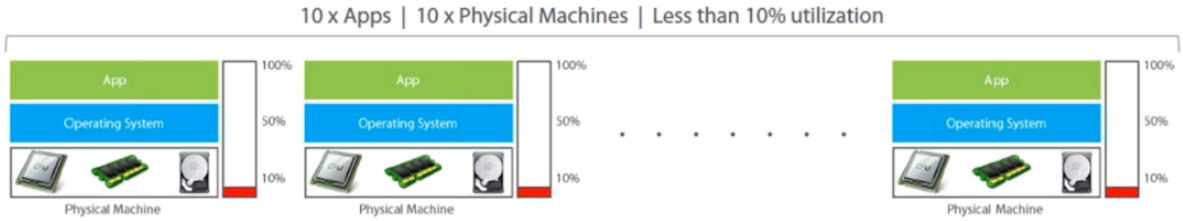
One Server = One Application

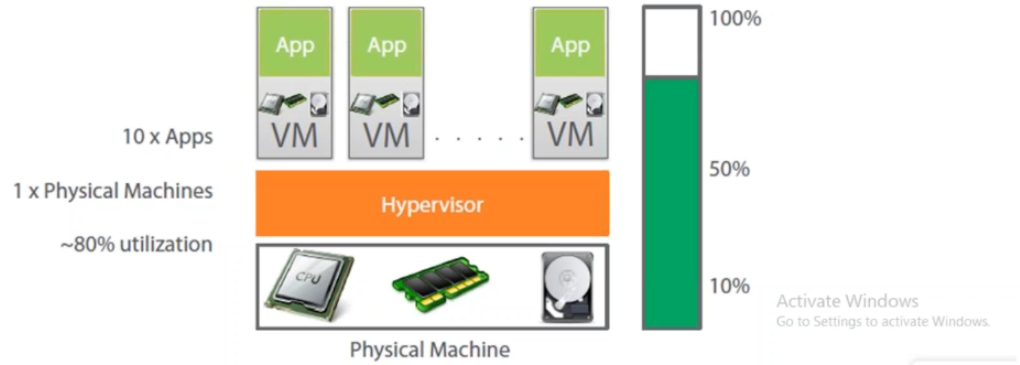


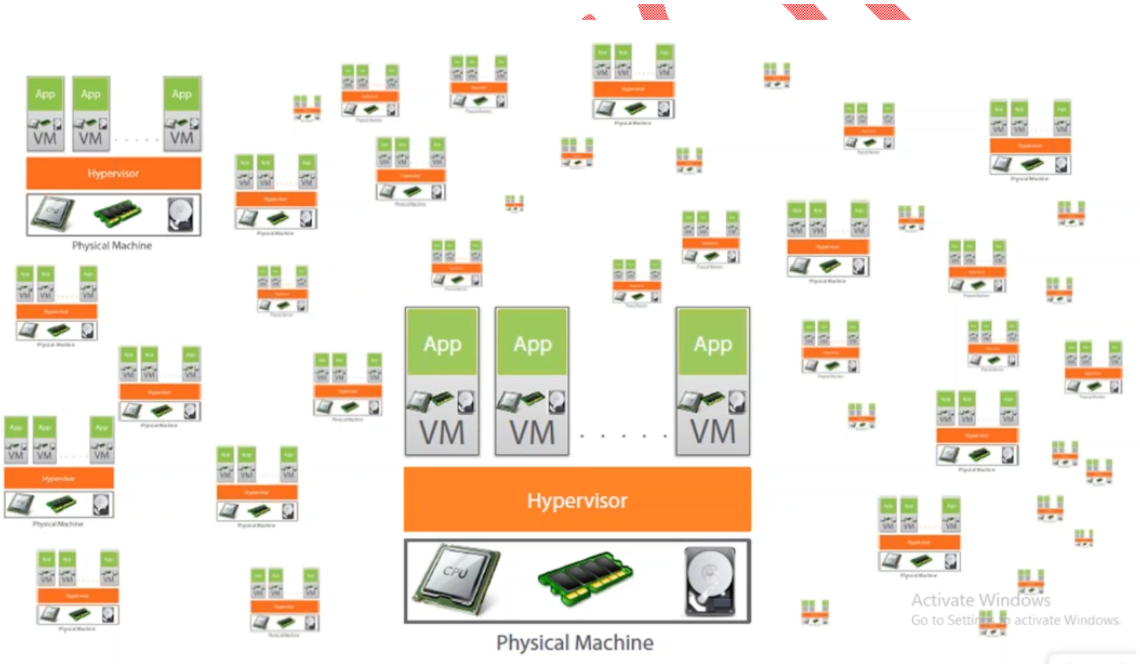


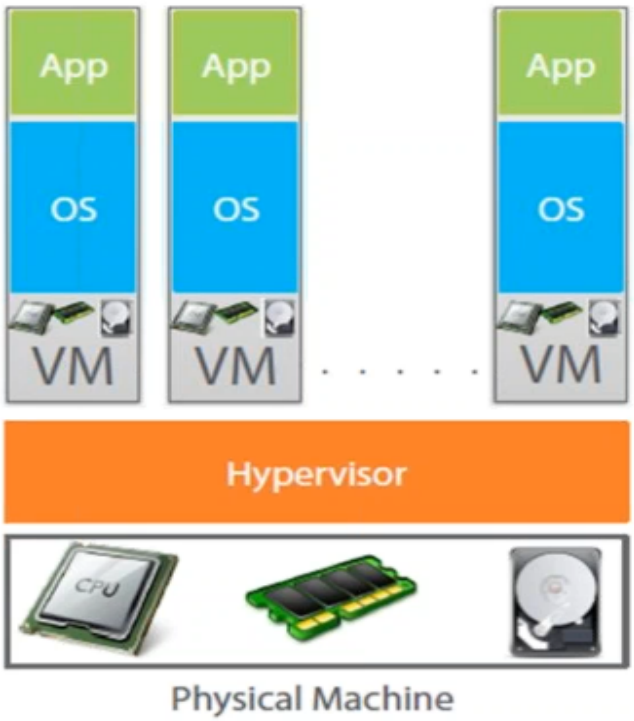
Here utilization 10% and wastage is 90%

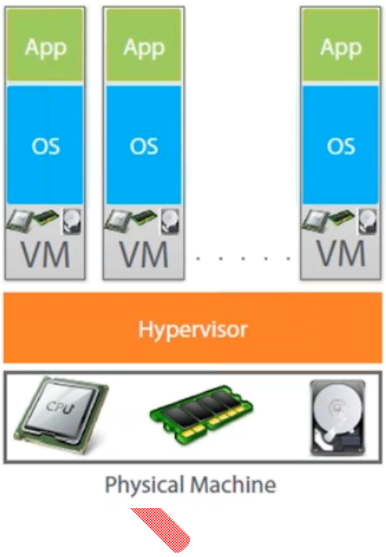



10 x Apps | 10 x Physical Machines | Less than 10% utilization
App
Operating System
100%
50%
10%
Physical Machine

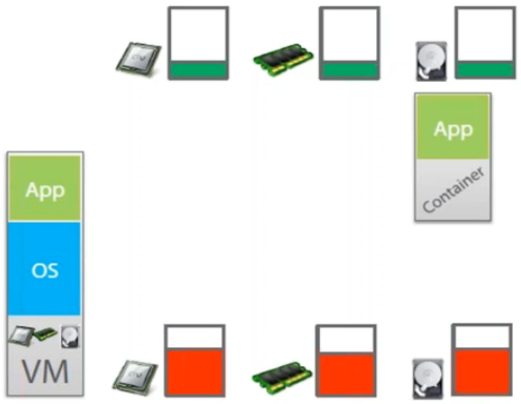

10 x Apps
1 x Physical Machines
~80% utilization
App
VM
Hypervisor
100%
50%
10%
Physical Machine
Activate Windows
Go to Settings to activate Windows.


App
VM
Hypervisor
Physical Machine
Activate Windows

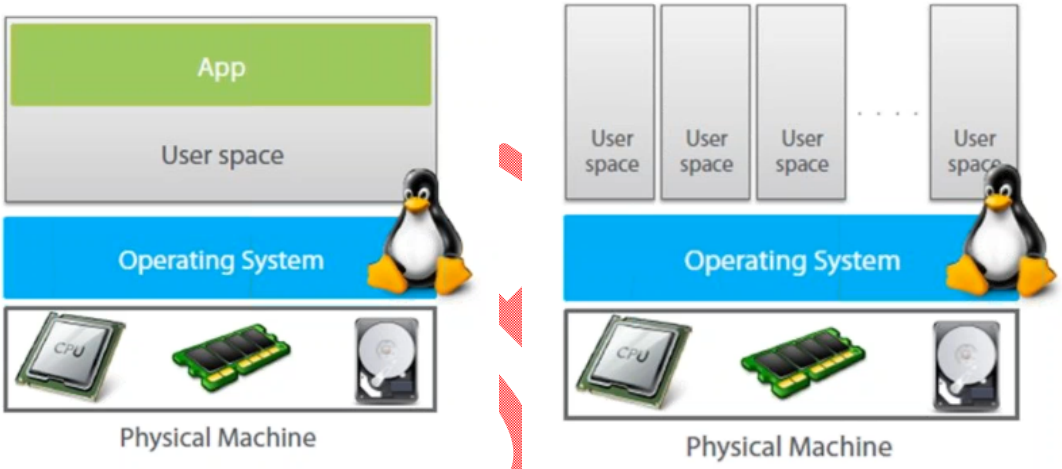




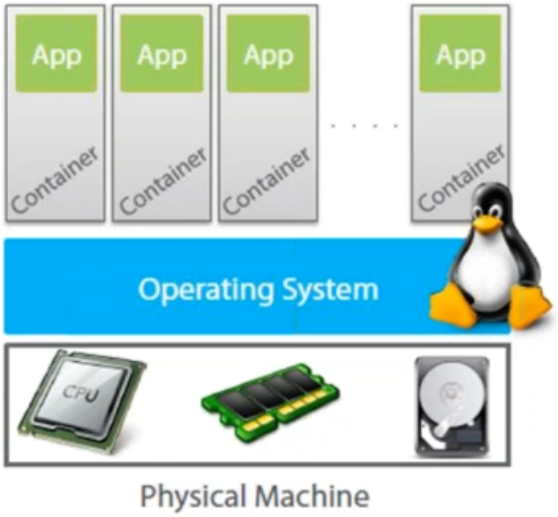
> OS != Business Value

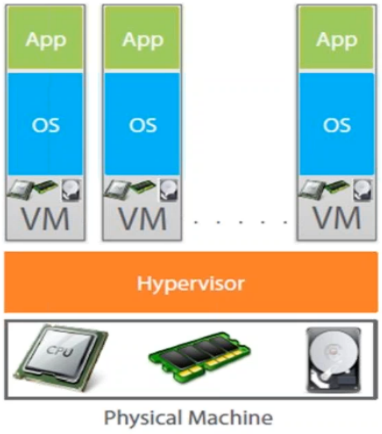


Containers are more lightweight than Virtual Machines



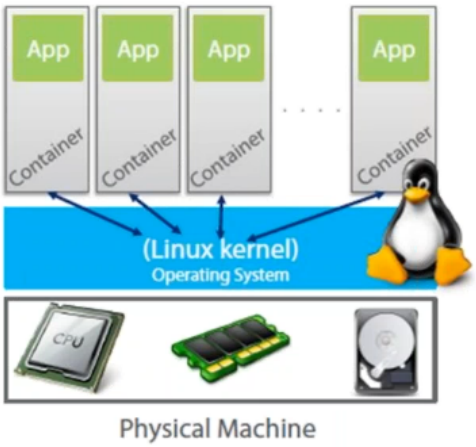
Here user space is nothing but users in OS.



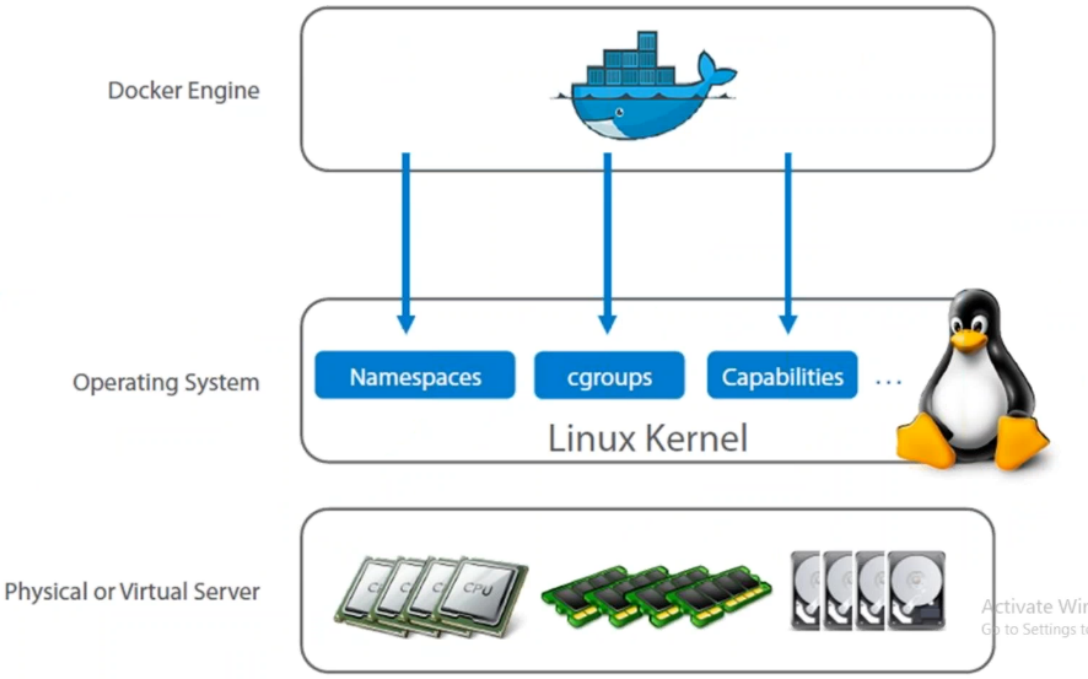


Each OS have:

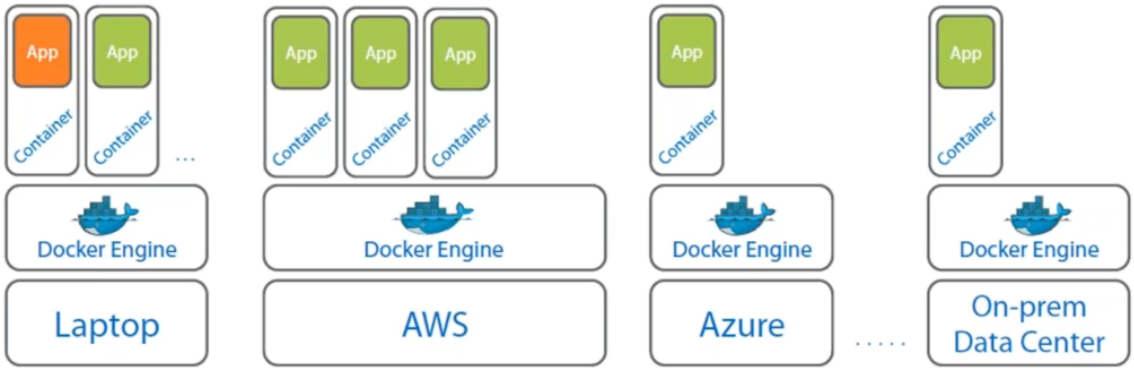
- Own Network
- Own Mounts
- Own Permissions
- Own Storage
- Own Process map



Containers consume less CPU, RAM and disk resource than Virtual Machines



……. Prepared by PPNREDDY



How application in docker is isolated ?

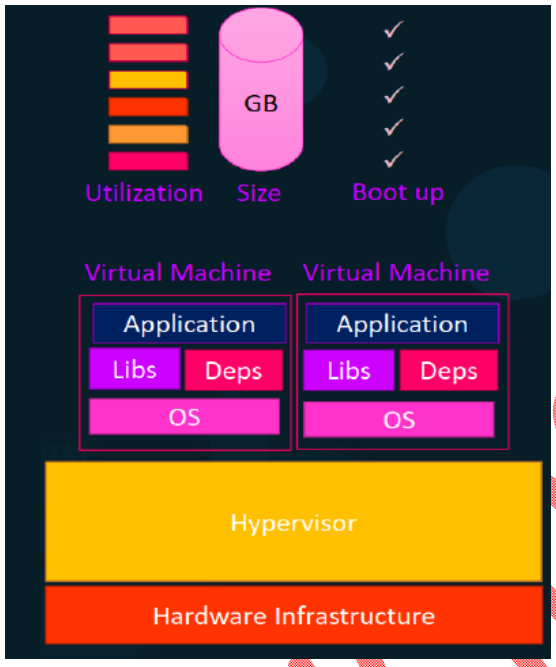
- Every app running on os will get process id
- Application will be running in a os which will have some ip address
- Application running in a os which will have storage
- Application running in an os which will have cpu& RAM
- Application running inside docker container also will get all of the above point
- Container is an isolated area of execution with
  - Storage
  - CPU
  - RAM
  - Process tree
  - network interface (ip address)

Hypervisor

- A hypervisor is software that creates and runs virtual machines (VMs) also known as guests.
- It isolates the hypervisor operating system and resources from the virtual machines and enables the creation and management of those VMs.
- The hypervisor treats host resources—like CPU, memory, and storage—as a pool that can be easily reallocated between existing guests or to new virtual machines.
- Generally, there are two types of hypervisors.

  Type-1: hypervisors, called "bare metal," run directly on the host's hardware. Ex: Microsoft Hyper-V or VMware ESXi hypervisor.

  Type-2: hypervisors, called "hosted," run as a software layer on an operating system. Ex: VirtualBox, VMware Player



Problems with hypervisor architecture

Now we know that every VM has its own OS, which is a problem. OS needs fair amount of resources like CPU, Memory, Storage etc. We also maintain OS licences and nurse them regularly like patching, upgrades, config changes. We wanted to host an application but collected good amount of fats over our infra, we are wasting OpEx and CapEx here. Think about shipping a vm from one place to other place, this sounds a great idea that if we could bundle everything in a vm image and ship it so the other person doesn't need to setup vm from scratch can directly run the vm from image. We did it in Vagrant chapter where we download preinstalled vm and have just run it.

But these images are heavy and bulky as they contain OS with the app. Booting them is a slow process. So being portable it's not convenient to ship the vm every time. Shipping an application bundled with all the dependencies/libraries in an image without OS. Hmm, sounds like we solved a big problem there. That's what containers are.

Think about setting up an application in a vm or physical machine. We need OS setup, dependencies, application deployed and some config changes in the OS. We follow a series of steps to setup all these like setting up a LAMP stack. If we could bundle all these into one container and ship it, then admins don't need to do any setup on the target, all we need to do is pull a container image and run it.

Docker

Docker is an OS-level virtualization tool, By using docker we can build, run and ship the applications.

- Build our own docker image with required dependences.
- Run the docker images any environment.
- Ship the docker image from one place to another place.
- Docker images are immutable, We can't change the existing layers.
- During run time docker containers use the host OS kernel.

Why docker?

Docker is designed to benefit both the Developer and System Administrator.

There are the following reasons to use docker.

- Docker allows us to easily install and run software without worrying about setup or dependencies.
- Scalable-light weight not like other VM's.
- Portable- Docker applications run anywhere.
- Build any application any language.
- Developers use Docker to eliminate machine problems, i.e. "***but code is worked on my laptop***." when working on code together with co-workers.
- Operators use Docker to run and manage apps in isolated containers for better compute density.
- Enterprises use Docker to securely built agile software delivery pipelines to ship new application features faster and more securely.
- Since docker is not only used for the deployment, but it is also a great platform for development, that's why we can efficiently increase our customer's satisfaction.
- We don't have to pre allocate RAM.

Advantages

- Application running on the docker performs better than running on virtual machines.
- Docker applications are highly portable. It solves the problems like,if application working on one environment and not working on another environment.
- Container uptime is 1 to 2 seconds, where as Virtual machine uptime is 1 to 2 minutes.

- One sever can host multiple applications.
- We can configure container with specific compute resources.
- Docker uses less memory and it provides lightweight virtualization.
- It does not a require full operating system to run applications.
- It uses application dependencies to reduce the risk.
- Docker allows you to use a remote repository to share your container with others.
- It provides continuous deployment and testing environment.

Disadvantages

- It increases complexity due to an additional layer.
- Docker is not a good solution for applications that require rich graphical user interface.
- Docker doesn't provides cross-platform compatibility means if an application is designed to run in a Docker container on Windows, then it can't run on Linux or vice versa.

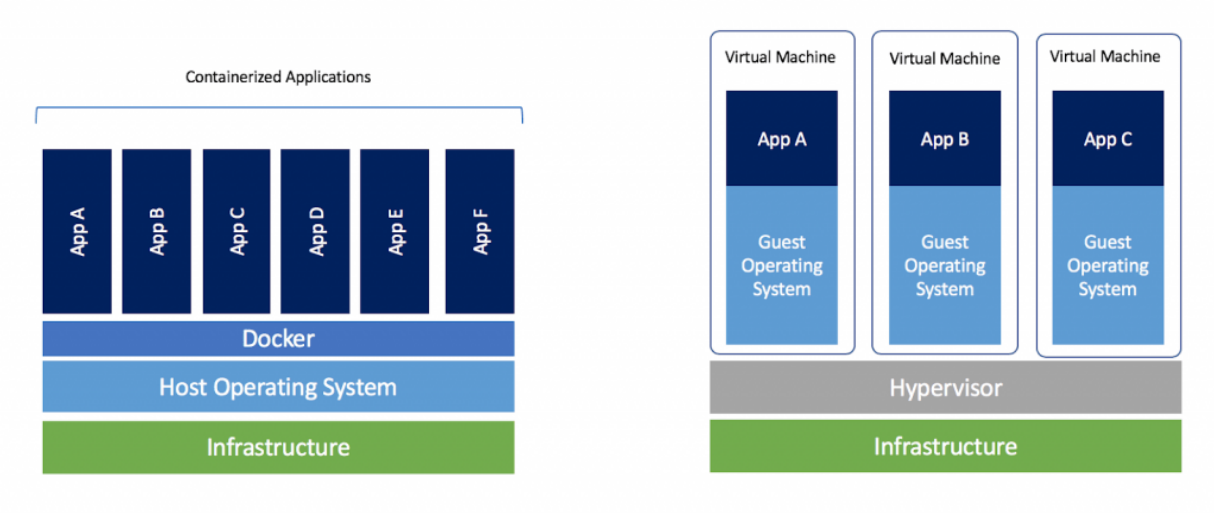
Difference between Docker and Hypervisor?

We can simply find out the difference between docker and hypervisor from below images, Hypervisor enables hardware level virtualization whereas docker enables OS level Virtualization.

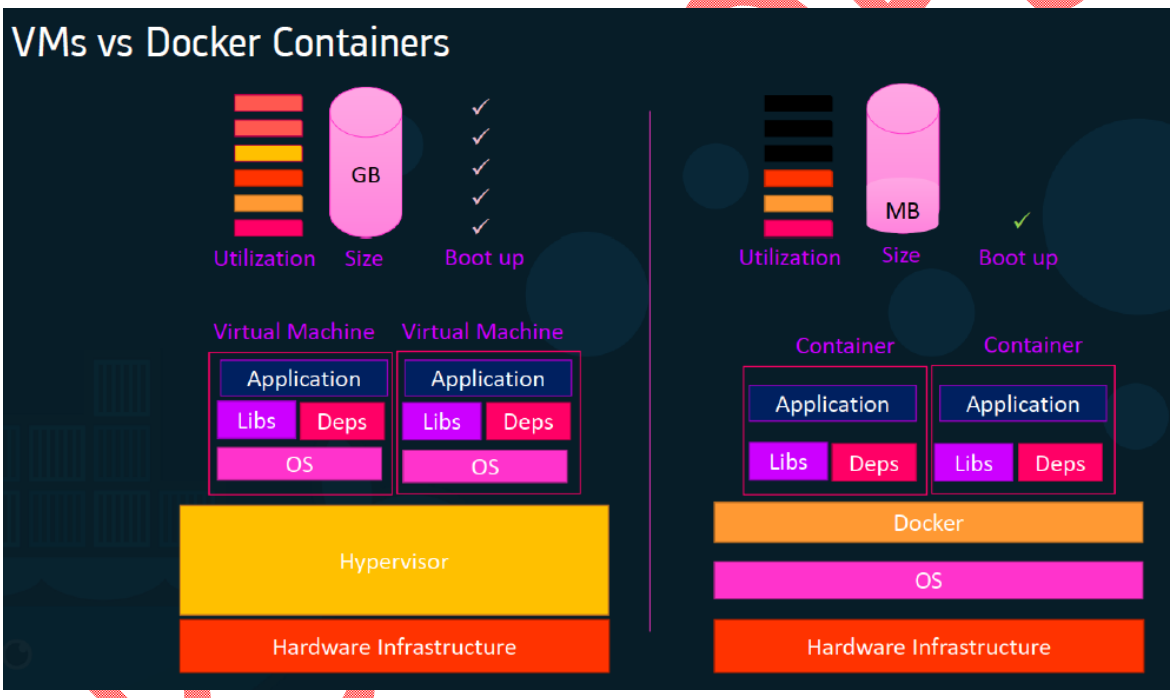
Docker uses same kernel of OS on top of which it is installed whereas in Hypervisor, every VM (Virtual Machine) will have its own kernel, this is also the reason why docker container takes less than a sec to come up whereas it takes more than 20 minutes for a Virtual machine in hypervisor to come up.

In docker we can easily scale up the number of containers which we require in a minute by running a simple command whereas in Hypervisor installed VM it takes minimum 4 hrs to scale around 10-20 VM.

Autoscaling feature is available with us in docker means docker can auto scale its container when a heave load will be there on a particular container while in case of hypervisor Our admin has to increase the resource manually on request of client, which is a big drawback as no companies knows when their traffic can increase suddenly to a high level and they would require high resource availability to maintain the huge traffic.

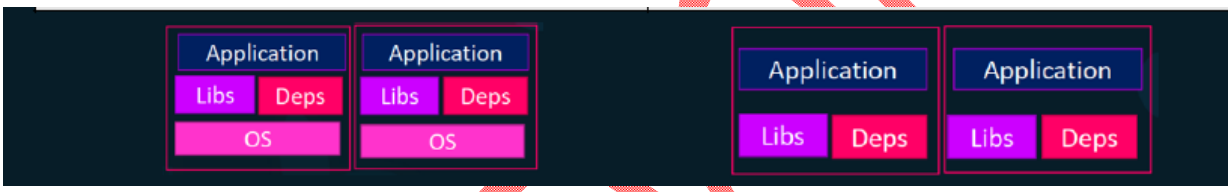


VM Vs Docker Containers



VMs vs Docker Containers

Virtual Machine	Docker Container
Hardware-level process isolation	OS level process isolation
Each VM has a separate OS	Each container can share OS
Boots in minutes	Boots in seconds
VMs are of few GBs	Containers are lightweight (KBs/MBs)
Ready-made VMs are difficult to find	Pre-built docker containers are easily available
VMs can move to new host easily	Containers are destroyed and re-created rather than moving
Creating VM takes a relatively longer time	Containers can be created in seconds
More resource usage	Less resource usage

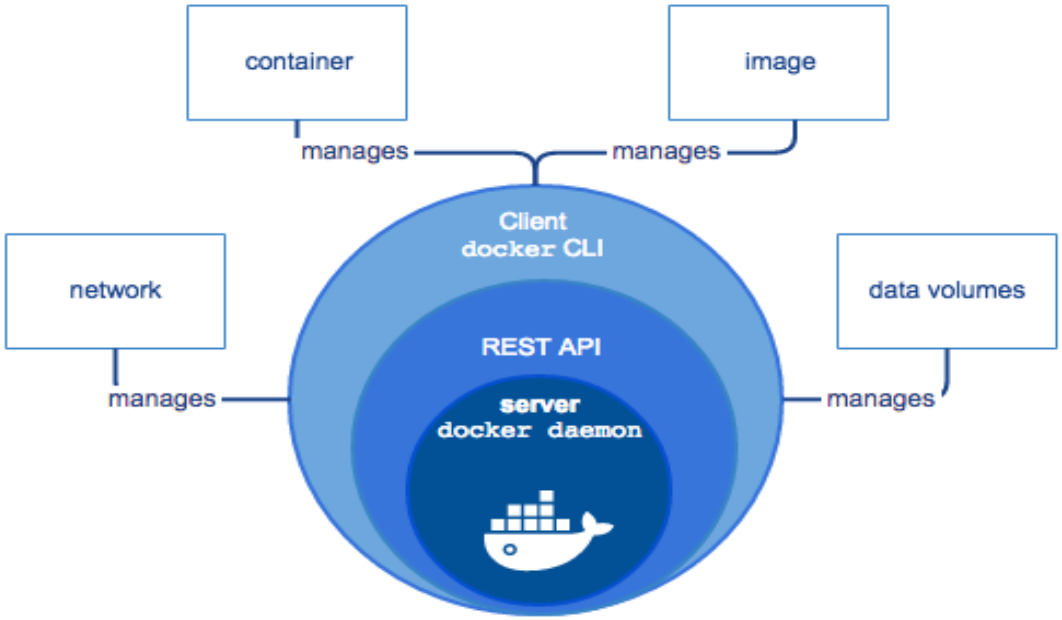


Docker Engine

It is a client server application that contains the following major components.

- A server which is a type of long-running program called a daemon process.
- The REST API is used to specify interfaces that programs can use to talk to the docker daemon and instruct it what to do.
- A command line interface client

At a high level, a docker server is, where the long running processes are managed, that is the daemon processes. To allow communication between the client machine and the docker server, REST API's made instructions are used to perform functions, what we normally call commands. Client instructs the docker daemon using the command line instructions.



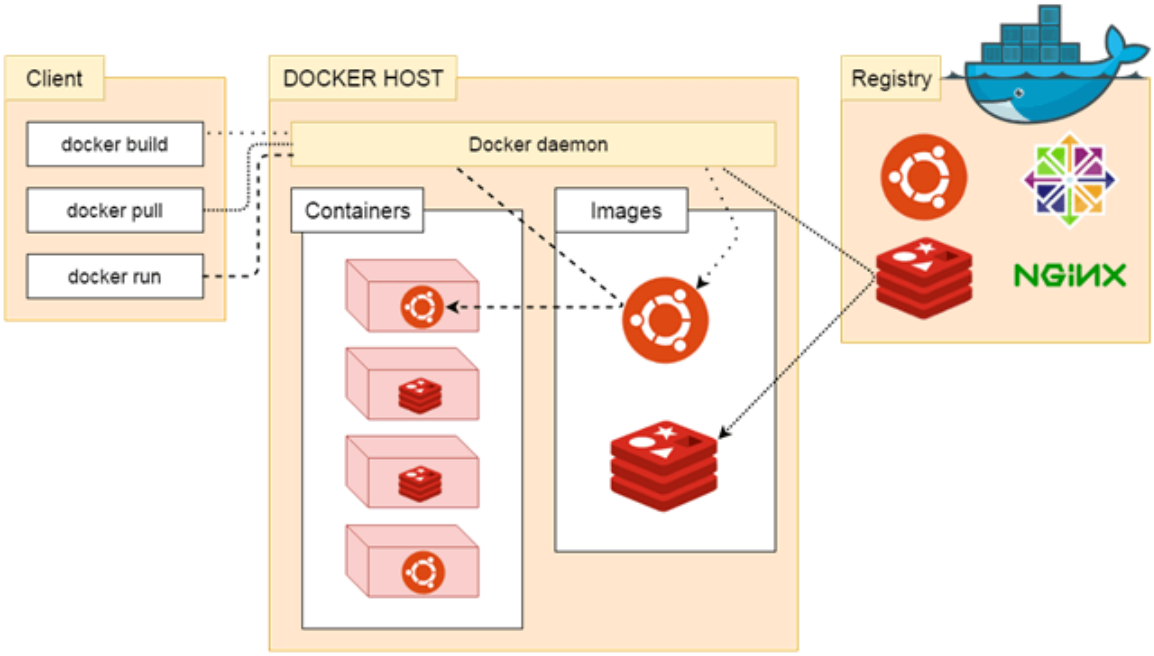
As shown in the above picture; Using the cli(command line interface), one can communicate with docker server or daemon and pass on the instruction about its usage. The daemon or the docker server creates and manages docker objects like images, containers, volumes and networking of the architecture.

What is Docker daemon?

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage.

Docker architecture

Docker follows Client-Server architecture, which includes the three main components that are Docker Client, Docker Host, and Docker Registry.



Docker Client:

Docker client uses commands and REST APIs to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

Docker Client uses Command Line Interface (CLI) to run the following commands.

- docker build
- docker pull
- docker run

Note: Docker Client has an ability to communicate with more than one docker daemon.

Docker Host

Docker Host is used to provide an environment to execute and run applications and listens for requests coming from the client and manages the docker objects. It contains the docker objects such as docker daemon, images, containers, networks, and storage.

Docker Registry

Docker registry is the place from where you can download and upload the docker images. It manages and stores the Docker images. In other words, Docker registry contains the docker repositories from where you can download the official and self made docker images to make a suitable environment for your application. Docker has its own public registry named docker hub and docker cloud from where you can download the official images. Apart from public repositories you can make your own private repositories to manage the private self designed images.

There are two types of registries in the Docker

Pubic Registry - Public Registry is also called as Docker hub.

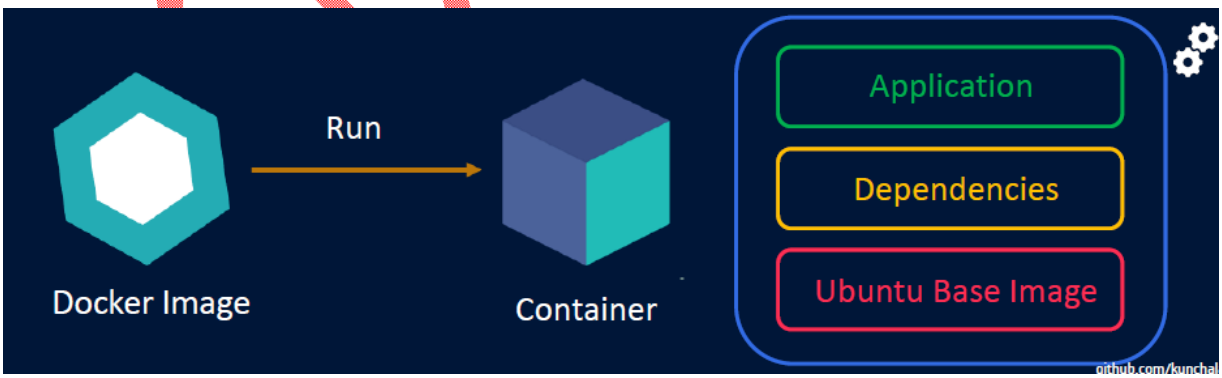
Private Registry - It is used to share images within the enterprise.

Docker Image: (Docker image is nothing but operating system without kernel)

Docker image can be compared to a template that is used to create Docker containers. These are read-only templates that contain application binaries and dependencies. Docker images are stored in the Docker Registry. If you want to run your application you have to create the docker containers with docker images. By creating the docker containers from the docker image we can run our applications.

Docker Container:

Docker container is a running instance of a Docker image as they hold the entire package needed to run the application.



Explain Docker components?

There are two categories of templates in docker.

1. Core components
2. Workflow components

1. Core components

Install and handle everything.

Docker daemon

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage.

Docker client

Docker client uses commands and REST APIs to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

2. Workflow components

Activity and perform some tasks

Docker image

Docker image can be compared to a template that is used to create Docker containers. These are read-only templates that contain application binaries and dependencies. Docker images are stored in the Docker Registry. If you want to run your application you have to create the docker containers with docker images. By creating the docker containers from the docker image we can run our applications.

Docker container

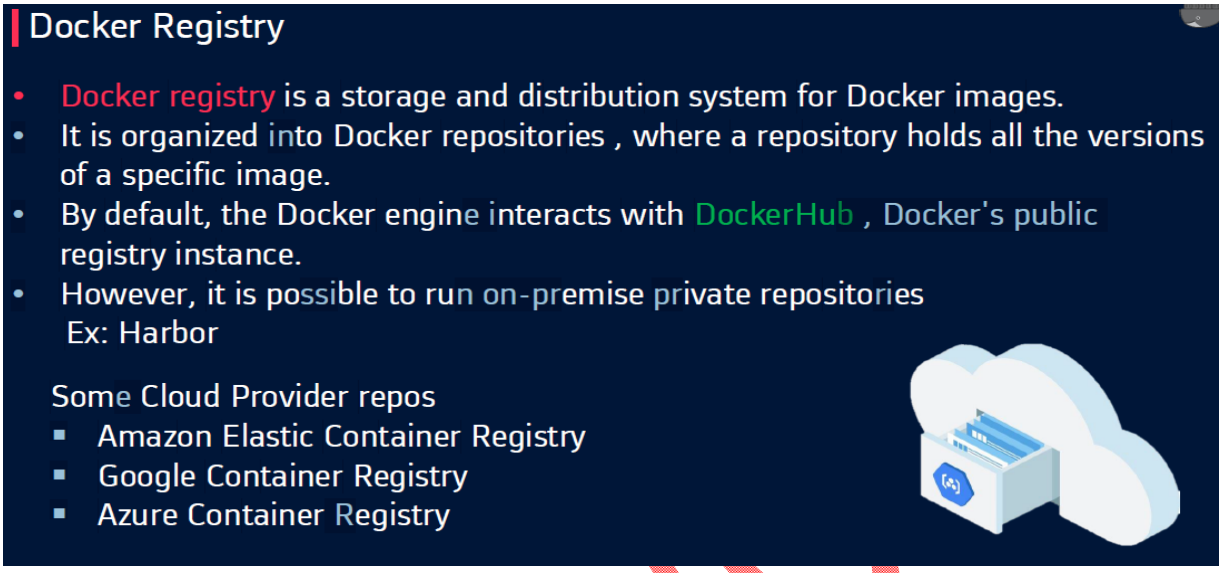
Docker container is a running instance of a Docker image as they hold the entire package needed to run the application.

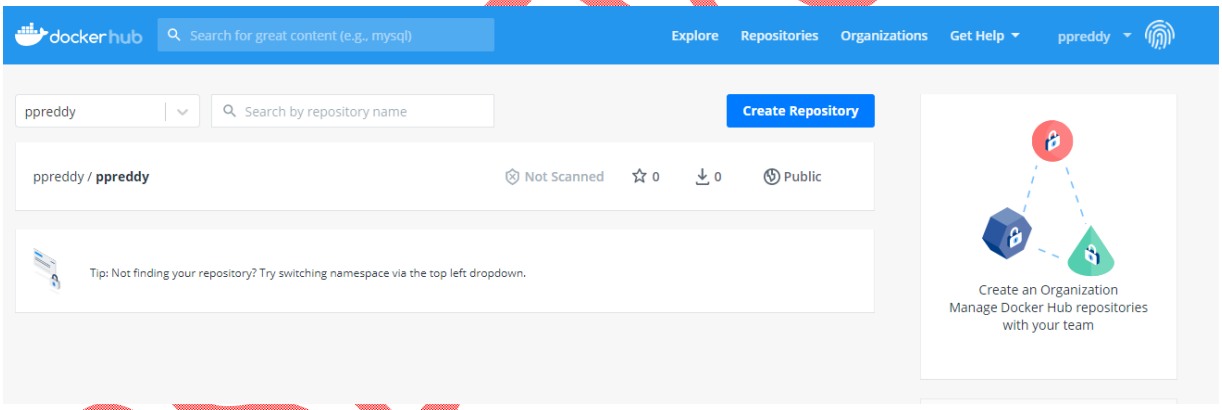
Docker registry

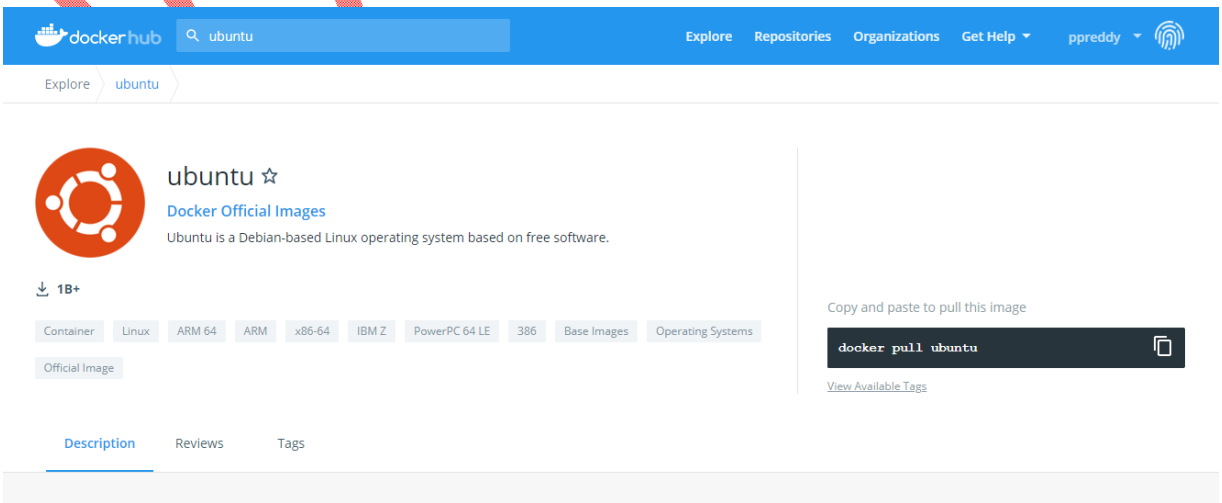
Docker registry is the place from where you can download and upload the docker images. It manages and stores the Docker images. In other words, Docker registry contains the docker repositories from where you can download the official and self made docker images to make a suitable environment for your application. Docker has its own public registry named docker hub and docker cloud from where you can download the official images. Apart from public repositories you can make your own private repositories to manage the private self designed images.

Docker file

By using docker file we can build the custom images from base image.







Docker installation on ubuntu server

Docker Engine runs natively on Linux distributions. Here, we are providing step by step process to install docker engine for Ubuntu.

Prerequisites:

Docker needs two important installation requirements:

- It only works on a 64-bit Linux installation.
- It requires Linux kernel version 3.10 or higher.

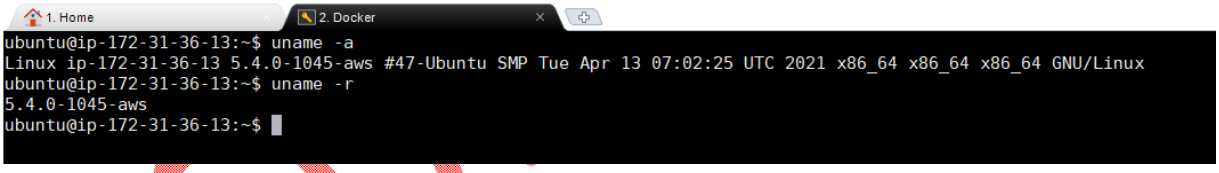
To check your current kernel version, open a terminal and type *uname -r* command to display your kernel version

Step-1: Launch ubuntu server in aws console.

Step-2: Verify the username and kernel version.

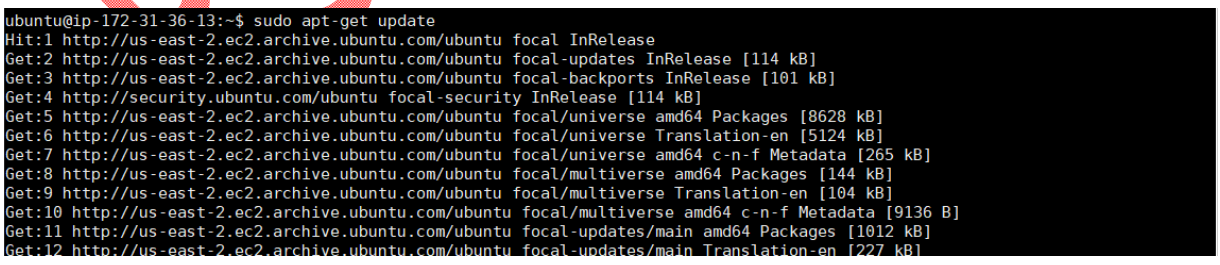
uname -a

uname -r



Step-3: Setup repository

sudo apt-get update



sudo apt-get install apt-transport-https ca-certificates curl software-properties-common -y

```
ubuntu@ip-172-31-36-13:~$ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20210119~20.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.68.0-1ubuntu2.5).
curl set to manually installed.
The following additional packages will be installed:
  python3-software-properties
The following NEW packages will be installed:
  apt-transport-https
```

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

```
ubuntu@ip-172-31-36-13:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK
ubuntu@ip-172-31-36-13:~$
```

sudo add-apt-repository "deb [arch=amd64]

https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
ubuntu@ip-172-31-36-13:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs
) stable"
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:5 https://download.docker.com/linux/ubuntu focal InRelease [41.0 kB]
Get:6 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [9960 B]
Fetched 165 kB in 0s (343 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-36-13:~$
```

sudo apt-get update

```
ubuntu@ip-172-31-36-13:~$ sudo apt-get update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:5 https://download.docker.com/linux/ubuntu focal InRelease
Fetched 114 kB in 0s (245 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-36-13:~$
```

Step-4: Verify the list of docker versions

apt-cache madison docker-ce

```
ubuntu@ip-172-31-36-13:~$ apt-cache madison docker-ce
 docker-ce | 5:20.10.7~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:20.10.6~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:20.10.5~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:20.10.4~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:20.10.3~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:20.10.2~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:20.10.1~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:20.10.0~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.15~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.14~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.13~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.12~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.11~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.10~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.9~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
ubuntu@ip-172-31-36-13:~$
```

Step-5: Install the latest docker version

sudo apt-get install docker-ce -y

```
ubuntu@ip-172-31-36-13:~$ sudo apt-get install docker-ce -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-ce-cli docker-ce-rootless-extras docker-scan-plugin pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras docker-scan-plugin pigz slirp4netns
0 upgraded, 7 newly installed, 0 to remove and 51 not upgraded.
Need to get 108 MB of archives.
After this operation, 466 MB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57.4 kB]
```

Checking the docker version

docker --version

```
ubuntu@ip-172-31-36-13:~$ docker --version
Docker version 20.10.7, build f0df350
ubuntu@ip-172-31-36-13:~$
```

Use Docker With Non-Root User:

If you want to use docker with non root user you must add that user to docker group. You can see in the below image i have executed docker command with devops user but it is showing permission denied. For that we have to add devops user to docker group.

```
ubuntu@ip-172-31-36-13:~$ docker image ls
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get http://%2Fvar%2F
run%2Fdocker.sock/v1.24/images/json: dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-36-13:~$
```

sudo –i

usermod -a -G docker ubuntu

```
ubuntu@ip-172-31-36-13:~$ sudo -i
root@ip-172-31-36-13:~# usermod -a -G docker ubuntu
root@ip-172-31-36-13:~#
```

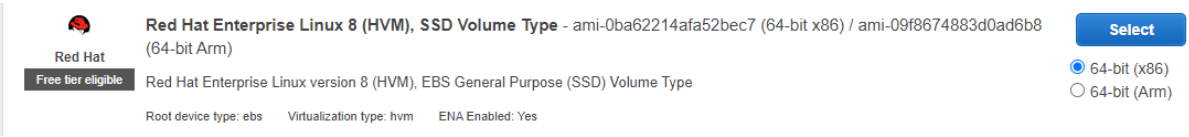
su ubuntu

docker image ls

```
root@ip-172-31-36-13:~# su ubuntu
ubuntu@ip-172-31-36-13:/root$ cd
ubuntu@ip-172-31-36-13:~$ docker image ls
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
ubuntu@ip-172-31-36-13:~$
```

Docker installation on RHEL machine

Step-1: Launch the RHEL in aws console



Step-2: Verify the user and kernel version

uname -a

uname -r

```
[ec2-user@ip-172-31-0-141 ~]$ uname -a
Linux ip-172-31-0-141.us-east-2.compute.internal 4.18.0-305.el8.x86_64 #1 SMP Thu Apr 29 08:54:30 EDT 2021 x86_64 x86_64 x86_6
4 GNU/Linux
[ec2-user@ip-172-31-0-141 ~]$ uname -r
4.18.0-305.el8.x86_64
[ec2-user@ip-172-31-0-141 ~]$
```

Step-3: Update the packages

sudo yum update -y

```
[ec2-user@ip-172-31-0-141 ~]$ sudo yum update -y
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

Red Hat Update Infrastructure 3 Client Configuration Server 8                15 kB/s | 3.8 kB     00:00
Red Hat Enterprise Linux 8 for x86_64 - AppStream from RHUI (RPMs)           35 MB/s |  30 MB     00:00
Red Hat Enterprise Linux 8 for x86_64 - BaseOS from RHUI (RPMs)              37 MB/s |  33 MB     00:00
Dependencies resolved.
```

Step-4: Install the dependencies

sudo yum install lvm2 device-mapper device-mapper-event-libs device-mapper-event device-mapper-libs device-mapper-persistent-data -y

```
[ec2-user@ip-172-31-0-141 ~]$ sudo yum install lvm2 device-mapper device-mapper-event-libs device-mapper-event device-mapper-l
ibs device-mapper-persistent-data -y
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

Last metadata expiration check: 0:02:53 ago on Fri 04 Jun 2021 03:20:28 PM UTC.
Package device-mapper-8:1.02.175-5.el8.x86_64 is already installed.
Package device-mapper-libs-8:1.02.175-5.el8.x86_64 is already installed.
Dependencies resolved.
================================================================================================================================
 Package                              Architecture     Version                  Repository                            Size
================================================================================================================================
```

Step-5: Add the docker yum repository

sudo yum-config-manager --add-repo

https://download.docker.com/linux/centos/docker-ce.repo

```
[ec2-user@ip-172-31-0-141 ~]$ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

Adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
[ec2-user@ip-172-31-0-141 ~]$
```

Step-6: Install the docker

sudo yum install docker-ce -y

```
[ec2-user@ip-172-31-0-141 ~]$ sudo yum install docker-ce -y
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

Last metadata expiration check: 0:01:09 ago on Fri 04 Jun 2021 03:28:35 PM UTC.
Dependencies resolved.
================================================================================
 Package                 Arch      Version                 Repository      Size
================================================================================
Installing:
```

Step-7: Verify the docker version

sudo docker --version

```
[ec2-user@ip-172-31-0-141 ~]$ sudo docker --version
Docker version 20.10.7, build f0df350
[ec2-user@ip-172-31-0-141 ~]$
```

Step-8: Access to normal user

sudo usermod -aG docker ec2-user

Docker Commands

The Docker commands are how you interact with Docker. You use Docker commands to build a Docker image, run a Docker container, send a Docker image to a remote/docker registry etc.

Starting/Stopping docker service

Using below commands we can start/stop/status the docker.

sudo systemctl status docker

sudo systemctl start docker

sudo systemctl stop docker

Verify the docker status

sudo systemctl status docker

```
[ec2-user@ip-172-31-42-105 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: inactive (dead) since Mon 2021-06-28 07:07:44 UTC; 18s ago
     Docs: https://docs.docker.com
  Process: 3321 ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock $OPTIONS $DOCKER_STORAGE_OPTIONS
$DOCKER_ADD_RUNTIMES (code=exited, status=0/SUCCESS)
  Process: 3313 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Process: 3303 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
 Main PID: 3321 (code=exited, status=0/SUCCESS)

Jun 28 06:42:32 ip-172-31-42-105.us-east-2.compute.internal dockerd[3321]: time="2021-06-28T06:42:32.468559824Z" level=info ...te"
Jun 28 06:42:32 ip-172-31-42-105.us-east-2.compute.internal dockerd[3321]: time="2021-06-28T06:42:32.562950149Z" level=info ...te"
Jun 28 06:42:32 ip-172-31-42-105.us-east-2.compute.internal dockerd[3321]: time="2021-06-28T06:42:32.563283808Z" level=info ...te"
Jun 28 06:42:32 ip-172-31-42-105.us-east-2.compute.internal dockerd[3321]: time="2021-06-28T06:42:32.577701218Z" level=info ...te"
Jun 28 06:47:02 ip-172-31-42-105.us-east-2.compute.internal dockerd[3321]: time="2021-06-28T06:47:02.468953579Z" level=error...wn"
Jun 28 07:07:44 ip-172-31-42-105.us-east-2.compute.internal systemd[1]: Stopping Docker Application Container Engine...
Jun 28 07:07:44 ip-172-31-42-105.us-east-2.compute.internal dockerd[3321]: time="2021-06-28T07:07:44.014671958Z" level=info ...d'"
Jun 28 07:07:44 ip-172-31-42-105.us-east-2.compute.internal dockerd[3321]: time="2021-06-28T07:07:44.049832110Z" level=info ...te"
Jun 28 07:07:44 ip-172-31-42-105.us-east-2.compute.internal dockerd[3321]: time="2021-06-28T07:07:44.115619975Z" level=info ...te"
Jun 28 07:07:44 ip-172-31-42-105.us-east-2.compute.internal systemd[1]: Stopped Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-42-105 ~]$
```

```
[ec2-user@ip-172-31-42-105 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2021-06-28 07:11:10 UTC; 6s ago
     Docs: https://docs.docker.com
  Process: 8812 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Process: 8802 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
 Main PID: 8828 (dockerd)
    Tasks: 7
   Memory: 37.4M
   CGroup: /system.slice/docker.service
           └─8828 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=1024:4096

Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.277914600Z" level=info ...rpc
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.278171817Z" level=info ...rpc
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.289140198Z" level=info ...y2"
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.312587747Z" level=info ...t."
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.404920800Z" level=info ...ss"
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.441135546Z" level=info ...e."
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.454496612Z" level=info ...0.4
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.454942776Z" level=info ...on"
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal systemd[1]: Started Docker Application Container Engine.
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.482482815Z" level=info ...ck"
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-42-105 ~]$
```

Start the docker service

sudo systemctl start docker

```
[ec2-user@ip-172-31-42-105 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-42-105 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2021-06-28 07:11:10 UTC; 6s ago
     Docs: https://docs.docker.com
  Process: 8812 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Process: 8802 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
 Main PID: 8828 (dockerd)
    Tasks: 7
   Memory: 37.4M
   CGroup: /system.slice/docker.service
           └─8828 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=1024:4096

Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.277914600Z" level=info ...rpc
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.278171817Z" level=info ...rpc
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.289140198Z" level=info ...y2"
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.312587747Z" level=info ...t."
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.404920800Z" level=info ...ss"
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.441135546Z" level=info ...e."
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.454496612Z" level=info ...0.4
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.454942776Z" level=info ...on"
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal systemd[1]: Started Docker Application Container Engine.
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.482482815Z" level=info ...ck"
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-42-105 ~]$
```

Stop the docker service

sudo systemctl stop docker

```
[ec2-user@ip-172-31-42-105 ~]$ sudo systemctl stop docker
Warning: Stopping docker.service, but it can still be activated by:
  docker.socket
[ec2-user@ip-172-31-42-105 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: inactive (dead) since Mon 2021-06-28 07:13:35 UTC; 4s ago
     Docs: https://docs.docker.com
  Process: 8828 ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock $OPTIONS $DOCKER_STORAGE_OPTIONS
$DOCKER_ADD_RUNTIMES (code=exited, status=0/SUCCESS)
  Process: 8812 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Process: 8802 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
 Main PID: 8828 (code=exited, status=0/SUCCESS)

Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.404920800Z" level=info ...ss"
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.441135546Z" level=info ...e."
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.454496612Z" level=info ...0.4
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.454942776Z" level=info ...on"
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal systemd[1]: Started Docker Application Container Engine.
Jun 28 07:11:10 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:11:10.482482815Z" level=info ...ck"
Jun 28 07:13:35 ip-172-31-42-105.us-east-2.compute.internal systemd[1]: Stopping Docker Application Container Engine...
Jun 28 07:13:35 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:13:35.189733981Z" level=info ...d'"
Jun 28 07:13:35 ip-172-31-42-105.us-east-2.compute.internal dockerd[8828]: time="2021-06-28T07:13:35.190399860Z" level=info ...te"
Jun 28 07:13:35 ip-172-31-42-105.us-east-2.compute.internal systemd[1]: Stopped Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-42-105 ~]$
```

docker system information

This command displays system wide information regarding the Docker installation. Information displayed includes the kernel version, number of containers and images. The number of images shown is the number of unique images. The same image tagged under different names is counted only once.

docker info

```
[ec2-user@ip-172-31-42-105 ~]$ docker info
Client:
 Context:    default
 Debug Mode: false

Server:
 Containers: 1
  Running: 0
  Paused: 0
  Stopped: 1
 Images: 1
 Server Version: 20.10.4
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runtime.v1.linux runc io.containerd.runc.v2
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 05f951a3781f4f2c1911b05e61c160e9c30eaa8e
 runc version: 12644e614e25b05da6fd08a38ffa0cfe1903fdec
 init version: de40ad0
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 4.14.232-177.418.amzn2.x86_64
 Operating System: Amazon Linux 2
 OSType: linux
 Architecture: x86_64
 CPUs: 1
 Total Memory: 983.3MiB
 Name: ip-172-31-42-105.us-east-2.compute.internal
 ID: OBWL:ALZ6:Y7HN:CN2N:VDUV:XGNE:MTRV:KV5Z:ZQ3P:IITX:TR4W:T5BA
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Username: ppreddy
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

[ec2-user@ip-172-31-42-105 ~]$
```

Verify the docker version

docker --version

```
[ec2-user@ip-172-31-42-105 ~]$ docker --version
Docker version 20.10.4, build d3cb89e
[ec2-user@ip-172-31-42-105 ~]$
```

Pull The Docker Images

The docker images will be available or located in docker registry or dockerhub. To get these images to your system you have to pull the images from dockerhub, for this we use docker pull command. If we want to pull the docker image we use docker pull command

Syntax:

docker pull <docker-image>

docker pull ubuntu

```
[ec2-user@ip-172-31-42-105 ~]$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    9873176a8ff5   7 days ago   72.7MB
[ec2-user@ip-172-31-42-105 ~]$
```

Here i have pulled the ubuntu docker image, by default it will pull the latest tag.

Verify the images in host

By using below commands we will verify list of images

docker images

docker image ls

Image storage path in host machine is /var/lib/docker/image/

```
[root@ip-172-31-42-105 ec2-user]# cd /var/lib/docker/image/
[root@ip-172-31-42-105 image]# pwd
/var/lib/docker/image
[root@ip-172-31-42-105 image]# ls
overlay2
[root@ip-172-31-42-105 image]# cd overlay2/
[root@ip-172-31-42-105 overlay2]# ll
total 4
drwx------ 4 root root  58 Jun 15 02:57 distribution
drwx------ 4 root root  37 Jun 15 02:48 imagedb
drwx------ 5 root root  45 Jun 17 02:13 layerdb
-rw------- 1 root root 274 Jun 25 05:25 repositories.json
[root@ip-172-31-42-105 overlay2]# cd distribution/
[root@ip-172-31-42-105 distribution]# ll
total 0
drwxr-xr-x 3 root root 20 Jun 15 02:57 diffid-by-digest
drwxr-xr-x 3 root root 20 Jun 15 02:57 v2metadata-by-diffid
[root@ip-172-31-42-105 distribution]# cd ..
[root@ip-172-31-42-105 overlay2]# cd imagedb/
[root@ip-172-31-42-105 imagedb]# ll
total 0
drwx------ 3 root root 20 Jun 15 02:48 content
drwx------ 3 root root 20 Jun 15 02:48 metadata
[root@ip-172-31-42-105 imagedb]#
```

Creating containers

If we want to create docker container from the pulled docker images, we use docker run command.

Syntax

docker run -i -t <image>:<tag> <entrypoint>

docker run -d <image>:<tag>

-i – interactive

-t – terminal

-d – demonized

docker run -i -t ubuntu:latest /bin/bash

Creates the container and enter to the terminal

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -i -t ubuntu:latest /bin/bash
root@a612ece01521:/#
```

exit → container stop and came out

```
root@a612ece01521:/# exit
exit
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE           COMMAND       CREATED              STATUS                          PORTS     NAMES
a612ece01521   ubuntu:latest   "/bin/bash"   About a minute ago   Exited (127) 11 seconds ago               heuristic_chandrasekhar
3024ae899aaf   ubuntu:latest   "bash"        50 minutes ago       Exited (0) 48 minutes ago                 c1
[ec2-user@ip-172-31-42-105 ~]$
```

docker run -d nginx:latest

Creates the container and running in backend side and didn't enter to the terminal.

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -d nginx:latest
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
b4d181a07f80: Pull complete
edb81c9bc1f5: Pull complete
b21fed559b9f: Pull complete
03e6a2452751: Pull complete
b82f7f888feb: Pull complete
5430e98eba64: Pull complete
Digest: sha256:47ae43cdfc7064d28800bc42e79a429540c7c80168e8c8952778c0d5af1c09db
Status: Downloaded newer image for nginx:latest
f285a98ec26ad08c9b6e7ed3a76e12990486182ecb3fae2c5b838f6ee99021f7
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   7 seconds ago   Up 6 seconds   80/tcp    vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$
```

Creating containers with name

If we want to create the names with use below command.

docker run -it --name myubuntucontainer ubuntu:latest /bin/bash

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -it --name myubuntucontainer ubuntu:latest /bin/bash
root@d74c42026f2e:/# [ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              10 seconds ago   Up 9 seconds                             myubuntucontainer
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   15 minutes ago   Up 15 minutes   0.0.0.0:49154->80/tcp    infallible_brahma
gupta
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   15 minutes ago   Up 15 minutes   0.0.0.0:49153->80/tcp    zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   18 minutes ago   Up 18 minutes   80/tcp                   vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$
```

Here *myubuntucontainer* is the container name.

Renaming the container name

By using docker rename we can change the container name but not container id

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              10 minutes ago   Up 10 minutes                            myubuntucontainer
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   25 minutes ago   Up 25 minutes   0.0.0.0:49154->80/tcp    infallible_brahma
gupta
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   25 minutes ago   Up 25 minutes   0.0.0.0:49153->80/tcp    zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   80/tcp                   vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$ docker rename myubuntucontainer mycontainer
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              10 minutes ago   Up 10 minutes                            mycontainer
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   25 minutes ago   Up 25 minutes   0.0.0.0:49154->80/tcp    infallible_brahma
gupta
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   25 minutes ago   Up 25 minutes   0.0.0.0:49153->80/tcp    zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   80/tcp                   vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$
```

docker rename d74c42026f2e d74c42026as3

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              12 minutes ago   Up 12 minutes                            mycontainer
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   0.0.0.0:49154->80/tcp    infallible_brahma
gupta
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   0.0.0.0:49153->80/tcp    zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   80/tcp                   vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$ docker rename d74c42026f2e d74c42026as3
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              13 minutes ago   Up 13 minutes                            d74c42026as3
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   0.0.0.0:49154->80/tcp    infallible_brahma
gupta
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   0.0.0.0:49153->80/tcp    zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   80/tcp                   vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$
```

Listing all containers

By using "docker ps –a" command we can list out all containers (Including stopped conatainers).

docker ps -a

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS                      PORTS                    NAMES
4357e0646ebf   nginx:latest   "/docker-entrypoint.…"   6 minutes ago       Exited (0) 5 minutes ago                             frosty_almeida
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   6 minutes ago       Up 6 minutes                0.0.0.0:49154->80/tcp    infallible_brahm
agupta
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   6 minutes ago       Up 6 minutes                0.0.0.0:49153->80/tcp    zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   9 minutes ago       Up 9 minutes                80/tcp                   vigilant_knuth
a612ece01521   ubuntu:latest  "/bin/bash"              12 minutes ago      Exited (127) 11 minutes ago                          heuristic_chandr
asekhar
3024ae899aaf   ubuntu:latest  "bash"                   About an hour ago   Exited (0) 59 minutes ago                            c1
[ec2-user@ip-172-31-42-105 ~]$
```

docker ps –aq (All container id’s)

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps -aq
4357e0646ebf
1a85c10bb698
20e2c4cf55ad
f285a98ec26a
a612ece01521
3024ae899aaf
[ec2-user@ip-172-31-42-105 ~]$
```

Listing running containers

By using “docker ps ” command we can list out all containers.

docker ps

docker ps –q (Running container id’s)

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                     NAMES
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   7 minutes ago    Up 7 minutes    0.0.0.0:49154->80/tcp     infallible_brahmagupta
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   7 minutes ago    Up 7 minutes    0.0.0.0:49153->80/tcp     zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   9 minutes ago    Up 9 minutes    80/tcp                    vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$ docker ps -q
1a85c10bb698
20e2c4cf55ad
f285a98ec26a
[ec2-user@ip-172-31-42-105 ~]$
```

Containers Start/Stop

By using docker stop and start commands we can achive.

docker ps

docker stop mycontainer

docker stop 1a85c10bb698

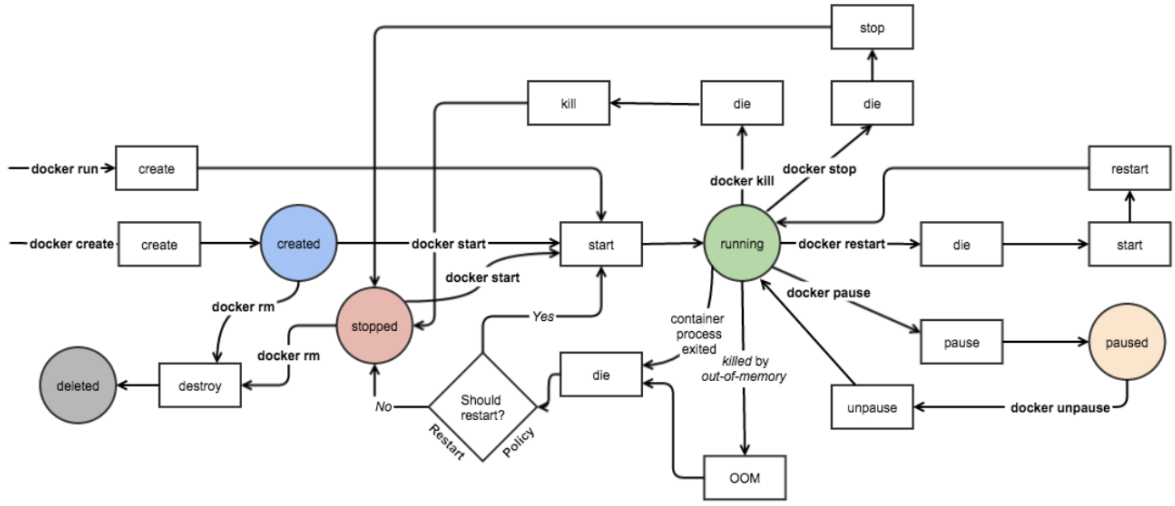
```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                     NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              18 minutes ago   Up 18 minutes                             mycontainer
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   33 minutes ago   Up 33 minutes   0.0.0.0:49154->80/tcp     mynginxcontainer
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   33 minutes ago   Up 33 minutes   0.0.0.0:49153->80/tcp     zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   36 minutes ago   Up 36 minutes   80/tcp                    vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$ docker stop mycontainer
mycontainer
[ec2-user@ip-172-31-42-105 ~]$ docker stop 1a85c10bb698
1a85c10bb698
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                     NAMES
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   34 minutes ago   Up 34 minutes   0.0.0.0:49153->80/tcp     zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   37 minutes ago   Up 37 minutes   80/tcp                    vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS                       PORTS
 NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              19 minutes ago      Exited (0) 24 seconds ago
 mycontainer
4357e0646ebf   nginx:latest   "/docker-entrypoint.…"   34 minutes ago      Exited (0) 34 minutes ago
 frosty_almeida
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   34 minutes ago      Exited (0) 12 seconds ago
 mynginxcontainer
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   34 minutes ago      Up 34 minutes                0.0.0.0:49153->80/tcp
 zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   37 minutes ago      Up 37 minutes                80/tcp
 vigilant_knuth
a612ece01521   ubuntu:latest  "/bin/bash"              40 minutes ago      Exited (127) 39 minutes ago
 heuristic_chandrasekhar
3024ae899aaf   ubuntu:latest  "bash"                   About an hour ago   Exited (0) About an hour ago
 c1
[ec2-user@ip-172-31-42-105 ~]$
```

docker start mycontainer

docker stop 1a85c10bb698

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                      PORTS                   NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              22 minutes ago   Exited (0) 3 minutes ago                            mycontainer
4357e0646ebf   nginx:latest   "/docker-entrypoint.…"   37 minutes ago   Exited (0) 37 minutes ago                           frosty_almeida
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   38 minutes ago   Exited (0) 3 minutes ago                            mynginxcontainer
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   38 minutes ago   Up 38 minutes               0.0.0.0:49153->80/tcp   zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   40 minutes ago   Up 40 minutes               80/tcp                  vigilant_knuth
a612ece01521   ubuntu:latest  "/bin/bash"              44 minutes ago   Exited (127) 42 minutes ago                         heuristic_chandrasekhar
3024ae899aaf   ubuntu:latest  "bash"                   2 hours ago      Exited (0) 2 hours ago                              c1
[ec2-user@ip-172-31-42-105 ~]$
[ec2-user@ip-172-31-42-105 ~]$ docker start mycontainer
mycontainer
[ec2-user@ip-172-31-42-105 ~]$ docker stop 1a85c10bb698
1a85c10bb698
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                   NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              23 minutes ago   Up 16 seconds                           mycontainer
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   38 minutes ago   Up 38 minutes   0.0.0.0:49153->80/tcp   zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   41 minutes ago   Up 41 minutes   80/tcp                  vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$ docker start 1a85c10bb698
1a85c10bb698
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                   NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              25 minutes ago   Up 2 minutes                            mycontainer
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   40 minutes ago   Up 4 seconds    0.0.0.0:49156->80/tcp   mynginxcontainer
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   40 minutes ago   Up 40 minutes   0.0.0.0:49153->80/tcp   zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   43 minutes ago   Up 43 minutes   80/tcp                  vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$
```

Docker container life-cycle



Create container

Create a container to run it later on with required image.

docker create --name <container-name> <image-name>

docker create --name myubuntucontainer ubuntu:latest

```
[ec2-user@ip-172-31-42-105 ~]$ docker create --name myubuntucontainer ubuntu:latest
53586844ba3baf5016b4d7b96db67da989866f1e7e6f08789aa23fa14448c2f2
[ec2-user@ip-172-31-42-105 ~]$
```

Run docker container

Run the docker container with the required image and specified command / process. '-d' flag is used for running the container in background.

docker run -it -d --name <container-name> <image-name> bash

docker run -d -it --name myubuntucontainer1 ubuntu:latest

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -d -it  --name myubuntucontainer1 ubuntu:latest
e3aebdca77757b014cbf0fdb0206c6022e7b22d3a39a0ee61680fef1c103e237
[ec2-user@ip-172-31-42-105 ~]$
```

Pause container

Used to pause the processes running inside the container.

docker pause <container-id/name>

docker pause myubuntucontainer1

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE           COMMAND   CREATED             STATUS             PORTS     NAMES
e3aebdca7775   ubuntu:latest   "bash"    About a minute ago   Up About a minute            myubuntucontainer1
[ec2-user@ip-172-31-42-105 ~]$ docker pause myubuntucontainer1
myubuntucontainer1
[ec2-user@ip-172-31-42-105 ~]$
```

Unpause container

Used to unpause the processes inside the container.

docker unpause <container-id/name>

docker unpause myubuntucontainer1

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE           COMMAND   CREATED         STATUS                  PORTS     NAMES
e3aebdca7775   ubuntu:latest   "bash"    3 minutes ago   Up 3 minutes (Paused)             myubuntucontainer1
[ec2-user@ip-172-31-42-105 ~]$ docker unpause myubuntucontainer1
myubuntucontainer1
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE           COMMAND   CREATED         STATUS         PORTS     NAMES
e3aebdca7775   ubuntu:latest   "bash"    4 minutes ago   Up 4 minutes             myubuntucontainer1
[ec2-user@ip-172-31-42-105 ~]$
```

Stop container

To stop the container and processes running inside the container:

docker stop <container-id/name>

To stop all the running docker containers

docker stop $(docker ps -a -q)

docker stop e3aebdca7775

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE           COMMAND   CREATED         STATUS         PORTS     NAMES
e3aebdca7775   ubuntu:latest   "bash"    4 minutes ago   Up 4 minutes             myubuntucontainer1
[ec2-user@ip-172-31-42-105 ~]$ clear
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE           COMMAND   CREATED         STATUS         PORTS     NAMES
e3aebdca7775   ubuntu:latest   "bash"    5 minutes ago   Up 5 minutes             myubuntucontainer1
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS                              PORTS                  NA
MES
e3aebdca7775   ubuntu:latest   "bash"                   5 minutes ago    Up 5 minutes                                               my
ubuntucontainer1
53586844ba3b   ubuntu:latest   "bash"                   7 minutes ago    Created                                                    my
ubuntucontainer
cd3bd8b9f907   ubuntu:latest   "/bin/bash"              3 hours ago      Exited (255) About an hour ago                             c1
46e3eca2c5c1   nginx:latest    "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8081->80/tcp      my
nginx1
1b23ea956d06   nginx:latest    "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8080->80/tcp      my
nginx
[ec2-user@ip-172-31-42-105 ~]$
[ec2-user@ip-172-31-42-105 ~]$ docker stop e3aebdca7775
e3aebdca7775
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS                          PORTS                  N
AMES
e3aebdca7775   ubuntu:latest   "bash"                   9 minutes ago    Exited (0) 11 seconds ago                              m
yubuntucontainer1
53586844ba3b   ubuntu:latest   "bash"                   11 minutes ago   Created                                                m
yubuntucontainer
cd3bd8b9f907   ubuntu:latest   "/bin/bash"              3 hours ago      Exited (255) About an hour ago                         c
1
46e3eca2c5c1   nginx:latest    "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8081->80/tcp  m
ynginx1
1b23ea956d06   nginx:latest    "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8080->80/tcp  m
ynginx
[ec2-user@ip-172-31-42-105 ~]$
```

Start container

Start the container, if present in stopped state.

docker start <container-id/name>

docker start e3aebdca7775

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                          PORTS                  N
AMES
e3aebdca7775   ubuntu:latest  "bash"                   11 minutes ago   Exited (0) 2 minutes ago                               m
yubuntucontainer1
53586844ba3b   ubuntu:latest  "bash"                   13 minutes ago   Created                                                m
yubuntucontainer
cd3bd8b9f907   ubuntu:latest  "/bin/bash"              3 hours ago      Exited (255) About an hour ago                         c
1
46e3eca2c5c1   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8081->80/tcp  m
ynginx1
1b23ea956d06   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8080->80/tcp  m
ynginx
[ec2-user@ip-172-31-42-105 ~]$
[ec2-user@ip-172-31-42-105 ~]$
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-42-105 ~]$
[ec2-user@ip-172-31-42-105 ~]$ docker start e3aebdca7775
e3aebdca7775
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND   CREATED          STATUS         PORTS     NAMES
e3aebdca7775   ubuntu:latest  "bash"    12 minutes ago   Up 4 seconds             myubuntucontainer1
[ec2-user@ip-172-31-42-105 ~]$
```

Restart container

It is used to restart the container as well as processes running inside the container.

docker restart <container-id/name>

docker restart e3aebdca7775

docker restart 1b23ea956d06

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND   CREATED          STATUS             PORTS     NAMES
e3aebdca7775   ubuntu:latest  "bash"    13 minutes ago   Up About a minute            myubuntucontainer1
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                          PORTS                  N
AMES
e3aebdca7775   ubuntu:latest  "bash"                   13 minutes ago   Up About a minute                                      m
yubuntucontainer1
53586844ba3b   ubuntu:latest  "bash"                   15 minutes ago   Created                                                m
yubuntucontainer
cd3bd8b9f907   ubuntu:latest  "/bin/bash"              3 hours ago      Exited (255) About an hour ago                         c
1
46e3eca2c5c1   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8081->80/tcp  m
ynginx1
1b23ea956d06   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8080->80/tcp  m
ynginx
[ec2-user@ip-172-31-42-105 ~]$ docker restart e3aebdca7775
e3aebdca7775
[ec2-user@ip-172-31-42-105 ~]$ docker restart 1b23ea956d06
1b23ea956d06
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                  NAMES
e3aebdca7775   ubuntu:latest  "bash"                   14 minutes ago   Up 19 seconds                          myubuntucontainer1
1b23ea956d06   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Up 6 seconds    0.0.0.0:8080->80/tcp   mynginx
[ec2-user@ip-172-31-42-105 ~]$
```

Kill container

We can kill the running container.

docker kill <container-id/name>

docker kill e3aebdca7775 e3aebdca7775

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS         PORTS                  NAMES
e3aebdca7775   ubuntu:latest  "bash"                   16 minutes ago   Up 2 minutes                          myubuntucontainer1
1b23ea956d06   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Up 2 minutes   0.0.0.0:8080->80/tcp   mynginx
[ec2-user@ip-172-31-42-105 ~]$ docker kill e3aebdca7775 e3aebdca7775
e3aebdca7775
e3aebdca7775
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS         PORTS                  NAMES
1b23ea956d06   nginx:latest   "/docker-entrypoint.…"   3 hours ago   Up 2 minutes   0.0.0.0:8080->80/tcp   mynginx
[ec2-user@ip-172-31-42-105 ~]$ docker kill e3aebdca7775 e3aebdca7775
Error response from daemon: Cannot kill container: e3aebdca7775: Container e3aebdca77757b014cbf0fdb0206c6022e7b22d3a39a0ee61680fef
1c103e237 is not running
Error response from daemon: Cannot kill container: e3aebdca7775: Container e3aebdca77757b014cbf0fdb0206c6022e7b22d3a39a0ee61680fef
1c103e237 is not running
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS         PORTS                  NAMES
1b23ea956d06   nginx:latest   "/docker-entrypoint.…"   3 hours ago   Up 2 minutes   0.0.0.0:8080->80/tcp   mynginx
[ec2-user@ip-172-31-42-105 ~]$
```

Destroy container

Its preferred to destroy container, only if present in stopped state instead of forcefully destroying the running container.

docker rm <container-id/name>

To remove all the stopped docker containers

docker rm $(docker ps –aq)

docker rm myubuntucontainer

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                       PORTS                  N
AMES
e3aebdca7775   ubuntu:latest  "bash"                   18 minutes ago   Exited (137) 2 minutes ago                          m
yubuntucontainer1
53586844ba3b   ubuntu:latest  "bash"                   20 minutes ago   Created                                             m
yubuntucontainer
cd3bd8b9f907   ubuntu:latest  "/bin/bash"              3 hours ago      Exited (255) About an hour ago                      c
1
46e3eca2c5c1   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8081->80/tcp   m
ynginx1
1b23ea956d06   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Exited (0) 15 seconds ago                           m
ynginx
[ec2-user@ip-172-31-42-105 ~]$ docker rm myubuntucontainer
myubuntucontainer
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                       PORTS                  N
AMES
e3aebdca7775   ubuntu:latest  "bash"                   18 minutes ago   Exited (137) 2 minutes ago                          m
yubuntucontainer1
cd3bd8b9f907   ubuntu:latest  "/bin/bash"              3 hours ago      Exited (255) About an hour ago                      c
1
46e3eca2c5c1   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8081->80/tcp   m
ynginx1
1b23ea956d06   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Exited (0) 36 seconds ago                           m
ynginx
[ec2-user@ip-172-31-42-105 ~]$
```

docker rm $(docker ps -aq)

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                       PORTS                  N
AMES
b6f974d392a5   ubuntu:latest  "/bin/bash"              39 seconds ago   Exited (0) 35 seconds ago                           m
ystifying_ganguly
aa8b72baf34e   ubuntu:latest  "/bin/bash"              45 seconds ago   Exited (0) 41 seconds ago                           v
ibrant_kilby
191886891e54   ubuntu:latest  "/bin/bash"              51 seconds ago   Exited (0) 47 seconds ago                           v
igorous_ptolemy
38e21a7d1e99   ubuntu:latest  "/bin/bash"              58 seconds ago   Exited (0) 54 seconds ago                           d
istracted_mccarthy
e3aebdca7775   ubuntu:latest  "bash"                   21 minutes ago   Exited (137) 5 minutes ago                          m
yubuntucontainer1
cd3bd8b9f907   ubuntu:latest  "/bin/bash"              3 hours ago      Exited (255) About an hour ago                      c
1
46e3eca2c5c1   nginx:latest   "/docker-entrypoint.…"   3 hours ago      Exited (255) About an hour ago   0.0.0.0:8081->80/tcp   m
ynginx1
1b23ea956d06   nginx:latest   "/docker-entrypoint.…"   4 hours ago      Exited (0) 3 minutes ago                            m
ynginx
```

```
[ec2-user@ip-172-31-42-105 ~]$ docker rm $(docker ps -aq )
b6f974d392a5
aa8b72baf34e
191886891e54
38e21a7d1e99
e3aebdca7775
cd3bd8b9f907
46e3eca2c5c1
1b23ea956d06
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-42-105 ~]$
```

Accessing running container

By using exec and attach commands we can enter to the running container.

If we use exec one new process will be create.

docker exec -it mycontainer /bin/bash

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS                   NAMES
d74c42026f2e   ubuntu:latest   "/bin/bash"              27 minutes ago   Up 4 minutes                            mycontainer
1a85c10bb698   nginx:latest    "/docker-entrypoint.…"   42 minutes ago   Up 2 minutes    0.0.0.0:49156->80/tcp   mynginxcontainer
20e2c4cf55ad   nginx:latest    "/docker-entrypoint.…"   42 minutes ago   Up 42 minutes   0.0.0.0:49153->80/tcp   zealous_jennings
f285a98ec26a   nginx:latest    "/docker-entrypoint.…"   45 minutes ago   Up 45 minutes   80/tcp                  vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$ docker exec -it mycontainer /bin/bash
root@d74c42026f2e:/# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 06:59 pts/0    00:00:00 /bin/bash
root         9     0  2 07:03 pts/1    00:00:00 /bin/bash
root        17     9  0 07:03 pts/1    00:00:00 ps -ef
root@d74c42026f2e:/#
```

If we use attach enter into the container without creating new process.

docker attach mycontainer

```
[ec2-user@ip-172-31-42-105 ~]$ docker attach  mycontainer
root@d74c42026f2e:/# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 06:59 pts/0    00:00:00 /bin/bash
root         9     0  0 07:03 pts/1    00:00:00 /bin/bash
root        18     1  0 07:05 pts/0    00:00:00 ps -ef
root@d74c42026f2e:/#
```

Checking resource utilization

By using stats command we can verify the resource utilization of a container

docker stats d74c42026f2e

```
[ec2-user@ip-172-31-42-105 ~]$ docker stats d74c42026f2e
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT     MEM %     NET I/O     BLOCK I/O     PIDS
d74c42026f2e   mycontainer   0.00%     4.848MiB / 983.3MiB   0.49%     920B / 0B   5.58MB / 0B   1
```

Checking internal process

By using top command we can verify the internal process of the container.

docker top mycontainer

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS       PORTS                     NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              7 hours ago   Up 7 hours                             mycontainer
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   7 hours ago   Up 7 hours   0.0.0.0:49156->80/tcp     mynginxcontainer
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   7 hours ago   Up 7 hours   0.0.0.0:49153->80/tcp     zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   7 hours ago   Up 7 hours   80/tcp                    vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$ docker top mycontainer
UID                 PID                 PPID                C                   STIME               TTY                 TIME
        CMD
root                7953                7910                0                   06:59               pts/0               00:00:00
        /bin/bash
[ec2-user@ip-172-31-42-105 ~]$
```

Checking complete properties

By using docker inspect command we can verify the complete properties of a file.

docker inspect d74c42026f2e

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS       PORTS                     NAMES
d74c42026f2e   ubuntu:latest  "/bin/bash"              7 hours ago   Up 7 hours                             mycontainer
1a85c10bb698   nginx:latest   "/docker-entrypoint.…"   7 hours ago   Up 7 hours   0.0.0.0:49156->80/tcp     mynginxcontainer
20e2c4cf55ad   nginx:latest   "/docker-entrypoint.…"   7 hours ago   Up 7 hours   0.0.0.0:49153->80/tcp     zealous_jennings
f285a98ec26a   nginx:latest   "/docker-entrypoint.…"   8 hours ago   Up 8 hours   80/tcp                    vigilant_knuth
[ec2-user@ip-172-31-42-105 ~]$ docker inspect d74c42026f2e
[
    {
        "Id": "d74c42026f2e9d6a42f6966b1fcb9d3fbfd34555291e905f03dbdbed5f1e2b7c",
        "Created": "2021-06-25T06:35:56.345121547Z",
        "Path": "/bin/bash",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 7953,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-06-25T06:59:04.905011609Z",
            "FinishedAt": "2021-06-25T06:54:42.611520475Z"
        },
        "Image": "sha256:9873176a8ff5ac192ce4d7df8a403787558b9f3981a4c4d74afb3edceeda451c",
        "ResolvConfPath": "/var/lib/docker/containers/d74c42026f2e9d6a42f6966b1fcb9d3fbfd34555291e905f03dbdbed5f1e2b7c/resolv.conf
",
        "HostnamePath": "/var/lib/docker/containers/d74c42026f2e9d6a42f6966b1fcb9d3fbfd34555291e905f03dbdbed5f1e2b7c/hostname",
```

Removing the docker objects

docker system prune

```
[ec2-user@ip-172-31-42-105 ~]$ docker system prune
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all dangling images
  - all dangling build cache

Are you sure you want to continue? [y/N] y
Deleted Networks:
docker_compose_demo_default

Total reclaimed space: 0B
[ec2-user@ip-172-31-42-105 ~]$
```

docker image prune –a

```
[ec2-user@ip-172-31-42-105 ~]$ docker image prune -a
WARNING! This will remove all images without at least one container associated to them.
Are you sure you want to continue? [y/N] y
Deleted Images:
untagged: nginx:latest
untagged: nginx@sha256:47ae43cdfc7064d28800bc42e79a429540c7c80168e8c8952778c0d5af1c09db
deleted: sha256:4f380adfc10f4cd34f775ae57a17d2835385efd5251d6dfe0f246b0018fb0399
deleted: sha256:2855bbcefcf95050e64049447e99e77efa2bff32374e586982d69be4612467ce
deleted: sha256:bad169ad8b30eab551acbb8cd8fbdcd824528189e3dd0cc52dd88a37bbf121cd
deleted: sha256:36d83ebf5fec7ae1be4c431f0945f2dbe6828ecdc936c604daa48f17c0b50ed7
deleted: sha256:b4c9a251dc81d52dd1cca9b4c69ca9e4db602a9a7974019f212846577f739699
deleted: sha256:038ca5b801cea48e9f40f6ffb4cda61a2fe0b6b0f378a7434a0d39d2575a4082
deleted: sha256:764055ebc9a7a290b64d17cf9ea550f1099c202d83795aa967428ebdf335c9f7
untagged: ubuntu:latest
untagged: ubuntu@sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
deleted: sha256:9873176a8ff5ac192ce4d7df8a403787558b9f3981a4c4d74afb3edceeda451c
deleted: sha256:feef05f055c989eea0367f5d3a2cba79896dd6d8a8b72bea2c729548a4ca5aef

Total reclaimed space: 205.9MB
[ec2-user@ip-172-31-42-105 ~]$
```

docker container prune

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE           COMMAND       CREATED          STATUS                      PORTS     NAMES
6a4ce6d5c743   ubuntu:latest   "/bin/bash"   12 seconds ago   Exited (0) 8 seconds ago              beautiful_cerf
03d70fc8f880   ubuntu:latest   "/bin/bash"   18 seconds ago   Exited (0) 15 seconds ago             infallible_napier
41ade6f930b7   ubuntu:latest   "/bin/bash"   27 seconds ago   Exited (0) 22 seconds ago             gracious_lovelace
cea9e6b5cd70   ubuntu:latest   "/bin/bash"   36 seconds ago   Exited (0) 32 seconds ago             xenodochial_dijkstra
[ec2-user@ip-172-31-42-105 ~]$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
6a4ce6d5c7433b3f3236e7a7f17f1b849e059fe8c1fc479ce8ed5ffe23366ac2
03d70fc8f88025986f677692aa32ffa5eb49d522ac3e4f7a406aafd5a538146b
41ade6f930b7050efebbe47f2983155dd64fa4373fcfd65601331d90148ed82e
cea9e6b5cd702af818c0d0da3ff20a6837b67fd7de766e88a87d1f338cdfedae

Total reclaimed space: 20B
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-42-105 ~]$
```

Docker restart policy

To configure the restart policy for a container, use the *--restart* flag when using the docker run command. The value of the *--restart* flag can be any of the following.

no: Do not automatically restart the container. (the default)

on-failure: Restart the container if it exits due to an error, which manifests as a non-zero exit code.

always: Always restart the container if it stops. If it is manually stopped, it is restarted only when Docker daemon restarts or the container itself is manually restarted.

unless-stopped: Similar to always, except that when the container is stopped (manually or otherwise), it is not restarted even after Docker daemon restarts

Restart policy details

Keep the following in mind when using restart policies:

- A restart policy only takes effect after a container starts successfully. In this case, starting successfully means that the container is up for at least 10 seconds and Docker has started monitoring it. This prevents a container which does not start at all from going into a restart loop.
- If you manually stop a container, its restart policy is ignored until the Docker daemon restarts or the container is manually restarted. This is another attempt to prevent a restart loop.
- Restart policies only apply to containers. Restart policies for swarm services are configured differently.

docker run -d --name nocontainer --restart no -P nginx:latest

docker inspect 63ee7042edae

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -d --name nocontainer --restart no -P nginx:latest
63ee7042edae7c1662fb3eb1c17a6fa4a6ab2d32f3ecff3b136f5f06a67dd973
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                   NAMES
63ee7042edae   nginx:latest   "/docker-entrypoint.…"   6 seconds ago   Up 5 seconds   0.0.0.0:49154->80/tcp   nocontainer
[ec2-user@ip-172-31-42-105 ~]$ docker inspect 63ee7042edae
[
    {
        "Id": "63ee7042edae7c1662fb3eb1c17a6fa4a6ab2d32f3ecff3b136f5f06a67dd973",
        "Created": "2021-06-28T06:18:00.870732155Z",
        "Path": "/docker-entrypoint.sh",
        "Args": [
            "nginx",
            "-g",
            "daemon off;"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 7303,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-06-28T06:18:01.374202261Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:4f380adfc10f4cd34f775ae57a17d2835385efd5251d6dfe0f246b0018fb0399",
```

```
        "HostConfig": {
            "Binds": null,
            "ContainerIDFile": "",
            "LogConfig": {
                "Type": "json-file",
                "Config": {}
            },
            "NetworkMode": "default",
            "PortBindings": {},
            "RestartPolicy": {
                "Name": "no",
                "MaximumRetryCount": 0
            },
            "AutoRemove": false,
            "VolumeDriver": "",
```

docker run -d --name ofcontainer --restart on-failure -P nginx:latest

docker inspect ofcontainer

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -d --name ofcontainer --restart on-failure -P nginx:latest
3338bf719cc4baded141427bf1248de89aa60d9b17af7e8b5d9ab6d1692a59e8
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                   NAMES
3338bf719cc4   nginx:latest   "/docker-entrypoint.…"   7 seconds ago   Up 6 seconds   0.0.0.0:49155->80/tcp   ofcontainer
63ee7042edae   nginx:latest   "/docker-entrypoint.…"   5 minutes ago   Up 5 minutes   0.0.0.0:49154->80/tcp   nocontainer
[ec2-user@ip-172-31-42-105 ~]$ docker inspect ofcontainer
[
    {
        "Id": "3338bf719cc4baded141427bf1248de89aa60d9b17af7e8b5d9ab6d1692a59e8",
        "Created": "2021-06-28T06:23:49.499777504Z",
        "Path": "/docker-entrypoint.sh",
        "Args": [
            "nginx",
            "-g",
            "daemon off;"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 7489,
            "ExitCode": 0,
            "Error": "",
```

```
        "HostConfig": {
            "Binds": null,
            "ContainerIDFile": "",
            "LogConfig": {
                "Type": "json-file",
                "Config": {}
            },
            "NetworkMode": "default",
            "PortBindings": {},
            "RestartPolicy": {
                "Name": "on-failure",
                "MaximumRetryCount": 0
            },
```

docker run -d --name alwayscontainer --restart always -P nginx:latest

docker inspect alwayscontainer

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -d --name alwayscontainer --restart always -P nginx:latest
8087f60ed50d09929012e92e7b13e23f9251d8d8dcd04aeb71f7834ec721c083
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                   NAMES
8087f60ed50d   nginx:latest   "/docker-entrypoint.…"   9 seconds ago    Up 8 seconds    0.0.0.0:49156->80/tcp   alwayscontainer
3338bf719cc4   nginx:latest   "/docker-entrypoint.…"   4 minutes ago    Up 4 minutes    0.0.0.0:49155->80/tcp   ofcontainer
63ee7042edae   nginx:latest   "/docker-entrypoint.…"   10 minutes ago   Up 10 minutes   0.0.0.0:49154->80/tcp   nocontainer
[ec2-user@ip-172-31-42-105 ~]$ docker inspect alwayscontainer
[
    {
        "Id": "8087f60ed50d09929012e92e7b13e23f9251d8d8dcd04aeb71f7834ec721c083",
        "Created": "2021-06-28T06:28:21.09018813Z",
        "Path": "/docker-entrypoint.sh",
        "Args": [
            "nginx",
            "-g",
            "daemon off;"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 7651,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-06-28T06:28:21.602546428Z",
```

```
        "HostConfig": {
            "Binds": null,
            "ContainerIDFile": "",
            "LogConfig": {
                "Type": "json-file",
                "Config": {}
            },
            "NetworkMode": "default",
            "PortBindings": {},
            "RestartPolicy": {
                "Name": "always",
                "MaximumRetryCount": 0
            },
```

docker run -d --name uscontainer --restart unless-stopped -P nginx:latest

docker inspect uscontainer

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -d --name uscontainer --restart unless-stopped -P nginx:latest
3e0d37df2190eeccc8c5b6a067c47678234e02ec8234c96626f0fa6b0a205884
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                   NAMES
3e0d37df2190   nginx:latest   "/docker-entrypoint.…"   7 seconds ago    Up 6 seconds    0.0.0.0:49157->80/tcp   uscontainer
8087f60ed50d   nginx:latest   "/docker-entrypoint.…"   3 minutes ago    Up 3 minutes    0.0.0.0:49156->80/tcp   alwayscontainer
3338bf719cc4   nginx:latest   "/docker-entrypoint.…"   7 minutes ago    Up 7 minutes    0.0.0.0:49155->80/tcp   ofcontainer
63ee7042edae   nginx:latest   "/docker-entrypoint.…"   13 minutes ago   Up 13 minutes   0.0.0.0:49154->80/tcp   nocontainer
[ec2-user@ip-172-31-42-105 ~]$ docker inspect uscontainer
[
    {
        "Id": "3e0d37df2190eeccc8c5b6a067c47678234e02ec8234c96626f0fa6b0a205884",
        "Created": "2021-06-28T06:31:39.940639875Z",
        "Path": "/docker-entrypoint.sh",
        "Args": [
            "nginx",
            "-g",
            "daemon off;"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 7823,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-06-28T06:31:40.424187297Z",
```

```
        "HostConfig": {
            "Binds": null,
            "ContainerIDFile": "",
            "LogConfig": {
                "Type": "json-file",
                "Config": {}
            },
            "NetworkMode": "default",
            "PortBindings": {},
            "RestartPolicy": {
                "Name": "unless-stopped",
                "MaximumRetryCount": 0
            },
```

Copy files from Host to Docker Container

using docker cp command we can copy the files from host machine to docker container.

Syntax:

docker cp /file/path/in/host <containerId>:/file/path/in/container/

docker run -it --name ubuntucontainer ubuntu:latest

root@65d5e0278032:/# mkdir psddevops

docker cp ppreddy.txt 65d5e0278032:/psddevops/

```
[ec2-user@ip-172-31-42-105 ~]$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
[ec2-user@ip-172-31-42-105 ~]$ clear
[ec2-user@ip-172-31-42-105 ~]$ docker run -it --name ubuntucontainer ubuntu:latest
root@65d5e0278032:/# pwd
/
root@65d5e0278032:/# mkdir psddevops
root@65d5e0278032:/# cd psddevops/
root@65d5e0278032:/psddevops# pwd
/psddevops
root@65d5e0278032:/psddevops# [ec2-user@ip-172-31-42-105 ~]$
[ec2-user@ip-172-31-42-105 ~]$ touch ppreddy.txt
[ec2-user@ip-172-31-42-105 ~]$ docker cp ppreddy.txt 65d5e0278032:/psddevops/
[ec2-user@ip-172-31-42-105 ~]$ docker exec -it 65d5e0278032 ls /psddevops
ppreddy.txt
[ec2-user@ip-172-31-42-105 ~]$ docker exec -it 65d5e0278032 /bin/bash
root@65d5e0278032:/# cd psddevops/
root@65d5e0278032:/psddevops# ls -lrt
total 0
-rw-rw-r-- 1 1000 1000 0 Jun 28 06:45 ppreddy.txt
root@65d5e0278032:/psddevops#
```

Copy files from Container to Host

Using docker cp command we can copy the files from docker container to host machine.

Syntax:

docker cp <containerId>:/file/path/in/container/ /file/path/in/host

docker cp 65d5e0278032:/psddevops/devops.txt .

```
root@65d5e0278032:/psddevops# ls
ppreddy.txt
root@65d5e0278032:/psddevops# touch devops.txt
root@65d5e0278032:/psddevops# ls
devops.txt  ppreddy.txt
root@65d5e0278032:/psddevops# read escape sequence
[ec2-user@ip-172-31-42-105 ~]$ docker cp 65d5e0278032:/psddevops/devops.txt .
[ec2-user@ip-172-31-42-105 ~]$ ls -lrt
total 73436
-rw-rw-r-- 1 ec2-user ec2-user      256 Jun 19 02:11 mypage.html
-rw-rw-r-- 1 ec2-user ec2-user    21674 Jun 21 02:46 mydata.json
-rw-r--r-- 1 ec2-user ec2-user    21674 Jun 21 02:50 mydata.json_21062021
drwxrwxr-x 2 ec2-user ec2-user       43 Jun 21 06:55 psddevops_volume
-rw-rw-r-- 1 ec2-user ec2-user      359 Jun 21 14:08 Dockerfile_1
-rw-rw-r-- 1 ec2-user ec2-user      364 Jun 21 14:11 Dockerfile
drwxrwxr-x 2 ec2-user ec2-user      196 Jun 23 13:29 dockerfiledemos
-rw------- 1 ec2-user ec2-user 75135488 Jun 23 15:57 ubuntu.tgz
drwxrwxr-x 2 ec2-user ec2-user       24 Jun 24 05:54 psdvolume
drwxrwxr-x 5 ec2-user ec2-user       87 Jun 24 13:57 docker_compose_demo
drwxrwxr-x 2 ec2-user ec2-user       64 Jun 28 02:20 devops6am
drwxrwxr-x 2 ec2-user ec2-user       38 Jun 28 02:35 myvolume
-rw-rw-r-- 1 ec2-user ec2-user        0 Jun 28 06:45 ppreddy.txt
-rw-r--r-- 1 ec2-user ec2-user        0 Jun 28 06:50 devops.txt
[ec2-user@ip-172-31-42-105 ~]$
```

Docker Volumes

By default all files created inside a container are stored on a writable container layer.

- The data doesn't persist when that container is no longer running, and it can be difficult to get the data out of the container if another process needs it.
- A container's writable layer is tightly coupled to the host machine where the container is running. You can't easily move the data somewhere else.
- Writing into a container's writable layer requires a storage driver to manage the filesystem. The storage driver provides a union filesystem, using the Linux kernel. This extra abstraction reduces performance as compared to using data volumes, which write directly to the host filesystem.
- Volumes can be create by Volume Command in Docker File.

Creating Volumes

docker volume ls

```
[ec2-user@ip-172-31-42-105 ~]$ docker volume ls
DRIVER    VOLUME NAME
[ec2-user@ip-172-31-42-105 ~]$
```

docker volume create my-volume

```
[ec2-user@ip-172-31-42-105 ~]$ docker volume create my-volume
my-volume
[ec2-user@ip-172-31-42-105 ~]$ docker volume ls
DRIVER    VOLUME NAME
local     my-volume
[ec2-user@ip-172-31-42-105 ~]$
```

Volumes path

/var/lib/docker/volumes

```
[root@ip-172-31-42-105 volumes]# ll
total 32
brw------- 1 root root 202, 1 Jun 21 06:11 backingFsBlockDev
-rw------- 1 root root  65536 Jun 21 06:18 metadata.db
drwx-----x 3 root root     19 Jun 21 06:18 my-volume
[root@ip-172-31-42-105 volumes]#
```

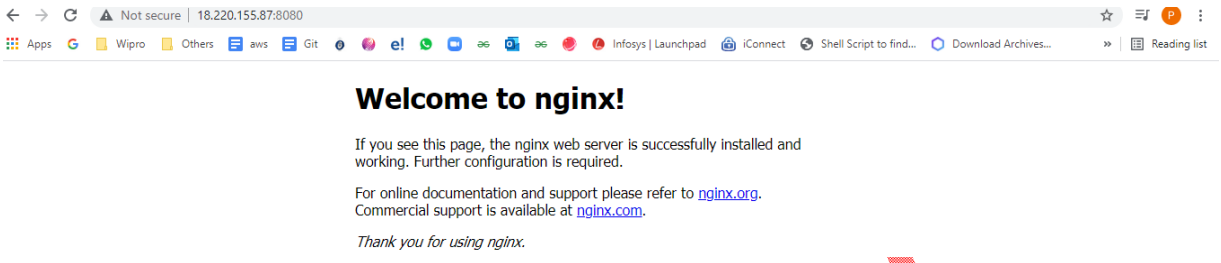
```
[root@ip-172-31-42-105 ~]# cd /var/lib/docker/volumes/my-volume/
[root@ip-172-31-42-105 my-volume]# ll
total 0
drwxr-xr-x 2 root root 6 Jun 21 06:18 _data
[root@ip-172-31-42-105 my-volume]# cd _data/
[root@ip-172-31-42-105 _data]# ll
total 0
```

Assigning volume to the container

docker run -itd --name mynginx -v my-volume:/usr/share/nginx/html -p 8080:80 nginx:latest

```
[ec2-user@ip-172-31-42-105 ~]$ clear
[ec2-user@ip-172-31-42-105 ~]$ docker run -itd --name mynginx -v my-volume:/usr/share/nginx/html -p 8080:80 nginx:latest
1dde0c5b248a6cb517be3ab233eb6d40440a4557a857412ca2c74882bacc4261
[ec2-user@ip-172-31-42-105 ~]$
```

```
[root@ip-172-31-42-105 _data]# ll
total 8
-rw-r--r-- 1 root root 494 May 25 12:28 50x.html
-rw-r--r-- 1 root root 612 May 25 12:28 index.html
[root@ip-172-31-42-105 _data]#
```

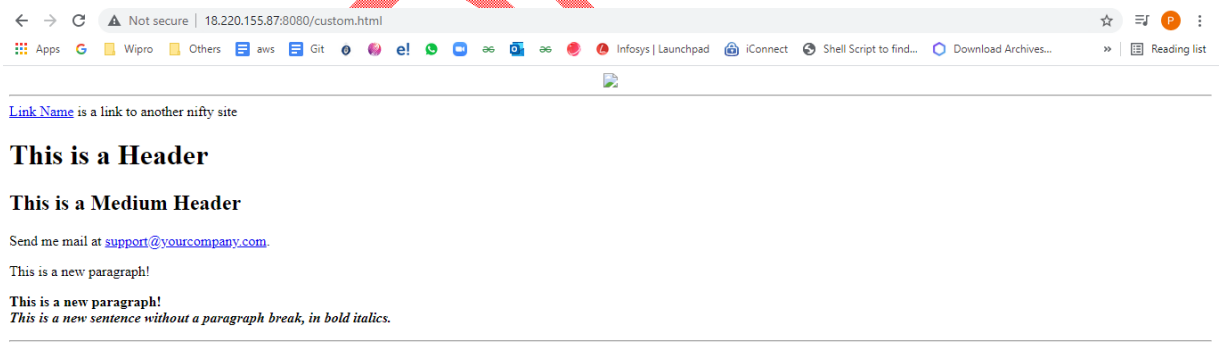


Add one file in container/volume path it will automatically present in volume/container path

Added custom.html file in volume path.

```
[root@ip-172-31-42-105 _data]# ll
total 12
-rw-r--r-- 1 root root 494 May 25 12:28 50x.html
-rw-r--r-- 1 root root 555 Jun 21 06:37 custom.html
-rw-r--r-- 1 root root 612 May 25 12:28 index.html
[root@ip-172-31-42-105 _data]#
```

```
root@1dde0c5b248a:/usr/share/nginx/html# ls  -lrt
total 12
-rw-r--r-- 1 root root 612 May 25 12:28 index.html
-rw-r--r-- 1 root root 494 May 25 12:28 50x.html
-rw-r--r-- 1 root root 555 Jun 21 06:37 custom.html
root@1dde0c5b248a:/usr/share/nginx/html#
```



Bind Volumes/Mounts

- Bind mounts means a file or directory on the host machine is mounted into a container.
- Bind mounts may be stored anywhere on the host system.
- Non-Docker processes on the Docker host or a Docker container can modify them at any time.
- Bind Mount can't be use in DockerFile.

Create a directory in the anywhere in host machine

mkdir psddevops_volume

```
[ec2-user@ip-172-31-42-105 ~]$ mkdir psddevops_volume
[ec2-user@ip-172-31-42-105 ~]$ cd psddevops_volume/
[ec2-user@ip-172-31-42-105 psddevops_volume]$ pwd
/home/ec2-user/psddevops_volume
[ec2-user@ip-172-31-42-105 psddevops_volume]$
```

Assigning bind mount to the container

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -itd --name mynginx1 -v /home/ec2-user/psddevops_volume:/usr/share/nginx/html -p 808
1:80 nginx:latest
a69068c55d51a953497787b4f5ab5a57f4486daabef95cb9d222ba8fd7629c8b
[ec2-user@ip-172-31-42-105 ~]$
```

```
[ec2-user@ip-172-31-42-105 psddevops_volume]$ touch myfile.txt
[ec2-user@ip-172-31-42-105 psddevops_volume]$
```

```
root@a69068c55d51:/# cd /usr/share/nginx/html/
root@a69068c55d51:/usr/share/nginx/html# ls -lrt
total 0
-rw-rw-r-- 1 1000 1000 0 Jun 21 06:52 myfile.txt
root@a69068c55d51:/usr/share/nginx/html# touch myhtml.html
root@a69068c55d51:/usr/share/nginx/html#
```

```
[ec2-user@ip-172-31-42-105 psddevops_volume]$ ll
total 0
-rw-rw-r-- 1 ec2-user ec2-user 0 Jun 21 06:52 myfile.txt
-rw-r--r-- 1 root     root     0 Jun 21 06:55 myhtml.html
[ec2-user@ip-172-31-42-105 psddevops_volume]$
```

Creating custom images

In Manual process first we need to take base image and base image is always any OS(operating system), so first we pull any OS image from docker hub, then we need to run this base image as container in interactive mode, now we will come inside the container, here we starts making the changes into the base image of OS as per our requirement , assume I am installing nginx server.

apt-get update

apt-get install nginx

modify default page of nginx

make some dir or file

come out of container

save the changes and this will be saved as new image which we can use anywhere.

docker pull ubuntu:latest

```
[ec2-user@ip-172-31-42-105 ~]$ docker pull ubuntu:latest
latest: Pulling from library/ubuntu
c549ccf8d472: Pull complete
Digest: sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    9873176a8ff5   3 days ago   72.7MB
```

docker run -it --name uc-1 ubuntu bash

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -it --name uc-1 ubuntu bash
root@dd605bdb92c6:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@dd605bdb92c6:/#
```

In above snap first, we had pulled the ubuntu image from docker hub and then ran that image to move inside the ubuntu container, now we will make some changes in this ubuntu container by installing nginx, modifying their default page and creating some directory.

apt-get update

```
root@dd605bdb92c6:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [328 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [884 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [27.6 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [730 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1301 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [984 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [355 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [32.0 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [4305 B]
Fetched 18.1 MB in 3s (6661 kB/s)
Reading package lists... Done
root@dd605bdb92c6:/#
```

apt-get install nginx -y

```
root@dd605bdb92c6:/# apt-get install nginx -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core iproute2 libatm1 libbsd0 libcap2 libcap2-bin libelf1 libexpat1 libfontconfig1
  libfreetype6 libgd3 libicu66 libjbig0 libjpeg-turbo8 libjpeg8 libmnl0 libnginx-mod-http-image-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libpam-cap libpng16-16 libssl1.1 libtiff5 libwebp6
  libx11-6 libx11-data libxau6 libxcb1 libxdmcp6 libxml2 libxpm4 libxslt1.1 libxtables12 nginx-common nginx-core tzdata ucf
Suggested packages:
  iproute2-doc libgd-tools fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  fontconfig-config fonts-dejavu-core iproute2 libatm1 libbsd0 libcap2 libcap2-bin libelf1 libexpat1 libfontconfig1
  libfreetype6 libgd3 libicu66 libjbig0 libjpeg-turbo8 libjpeg8 libmnl0 libnginx-mod-http-image-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libpam-cap libpng16-16 libssl1.1 libtiff5 libwebp6
  libx11-6 libx11-data libxau6 libxcb1 libxdmcp6 libxml2 libxpm4 libxslt1.1 libxtables12 nginx nginx-common nginx-core
  tzdata ucf
0 upgraded, 40 newly installed, 0 to remove and 6 not upgraded.
```

service nginx status

service nginx start

service nginx status

```
root@dd605bdb92c6:/# service nginx status
 * nginx is not running
root@dd605bdb92c6:/# service nginx start
 * Starting nginx nginx                                                                    [ OK ]
root@dd605bdb92c6:/# service nginx status
 * nginx is running
root@dd605bdb92c6:/#
```

ps -ef | grep nginx

```
root@dd605bdb92c6:/# ps -ef | grep nginx
root       828     1  0 14:52 ?        00:00:00 nginx: master process /usr/sbin/nginx
www-data   830   828  0 14:52 ?        00:00:00 nginx: worker process
root       843     1  0 14:53 pts/0    00:00:00 grep --color=auto nginx
root@dd605bdb92c6:/#
```

Now we can check that our nginx got installed inside /etc/nginx/ and the default output of nginx present in /var/www/html/index.nginx-debian.html , so we will change the output of this page to some random message, here I had created a directory named psddevops inside which I had created a text file named devops.txt , now I will come out of the container and saves the container as new image.

vi index.nginx-debian.html

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
dd605bdb92c6   ubuntu    "bash"    16 minutes ago   Up 16 minutes             uc-1
[ec2-user@ip-172-31-42-105 ~]$ docker exec -it dd605bdb92c6 /bin/bash
root@dd605bdb92c6:/# cd /var/www/html/
root@dd605bdb92c6:/var/www/html# vi
.index.nginx-debian.html.swp  index.nginx-debian.html
root@dd605bdb92c6:/var/www/html# vi index.nginx-debian.html
root@dd605bdb92c6:/var/www/html# cp index.nginx-debian.html index.html
root@dd605bdb92c6:/var/www/html#
```

```
root@dd605bdb92c6:~# mkdir psddevops
root@dd605bdb92c6:~# cd psddevops/
root@dd605bdb92c6:~/psddevops# vi devops.txt
root@dd605bdb92c6:~/psddevops# cat devops.txt
This is devops.txt file
root@dd605bdb92c6:~/psddevops#
```

In below image we have committed the container with name mynginx, but we can easily see when we ran the container no port is assigned to it and nginx is not in running status, because no process is running inside it and we haven't given any process to run inside the created newly image so it is taking the default process bash of OS, but to run nginx we need to run.

docker ps

docker commit -m "nginx installed" dd605bdb92c6 mynginx

docker images

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
dd605bdb92c6   ubuntu    "bash"    25 minutes ago   Up 10 seconds             uc-1
[ec2-user@ip-172-31-42-105 ~]$ docker commit -m "nginx installed" dd605bdb92c6 mynginx
sha256:34a8417bf94f2697f2ad453c8bc8edfd9d8395b183591e803e3bbf3a82ee10bd
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
mynginx       latest    34a8417bf94f   About a minute ago   223MB
ubuntu        latest    9873176a8ff5   3 days ago           72.7MB
[ec2-user@ip-172-31-42-105 ~]$
```

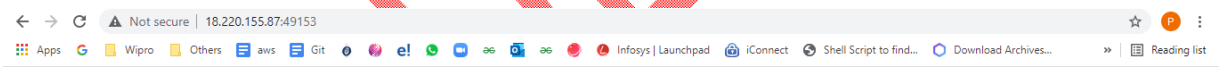
```
[ec2-user@ip-172-31-42-105 ~]$ docker run -itd mynginx
91beb0537bcc4fd39fb3e98f9d356a2d20fb3948e80984b9a1c99a321ca10bd0
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
91beb0537bcc   mynginx   "bash"    18 seconds ago   Up 17 seconds             confident_neumann
dd605bdb92c6   ubuntu    "bash"    32 minutes ago   Up 7 minutes              uc-1
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
mynginx       latest    34a8417bf94f   6 minutes ago    223MB
ubuntu        latest    9873176a8ff5   3 days ago       72.7MB
[ec2-user@ip-172-31-42-105 ~]$
```

docker commit -m "configured nginx" -c 'CMD /usr/sbin/nginx -g "daemon off;"' -c 'EXPOSE 80' dd605bdb92c6 new-nginx

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
91beb0537bcc   mynginx   "bash"    4 minutes ago    Up 4 minutes              confident_neumann
dd605bdb92c6   ubuntu    "bash"    37 minutes ago   Up 12 minutes             uc-1
[ec2-user@ip-172-31-42-105 ~]$ docker commit -m "configured nginx" -c 'CMD /usr/sbin/nginx -g "daemon off;"' -c 'EXPOSE 80' dd
605bdb92c6 new-nginx
sha256:6f5aec872df2e1eb44d62160509dd151827431d4aed5439d9301bd0cb421612b
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
new-nginx     latest    6f5aec872df2   29 seconds ago   223MB
mynginx       latest    34a8417bf94f   12 minutes ago   223MB
ubuntu        latest    9873176a8ff5   3 days ago       72.7MB
[ec2-user@ip-172-31-42-105 ~]$
```

```
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
new-nginx     latest    6f5aec872df2   2 minutes ago    223MB
mynginx       latest    34a8417bf94f   13 minutes ago   223MB
ubuntu        latest    9873176a8ff5   3 days ago       72.7MB
[ec2-user@ip-172-31-42-105 ~]$ docker run -itd new-nginx
6e301f7fbcc64c7771eaca39c65b65885d7ba55e9d28955a5004be61bf820710
[ec2-user@ip-172-31-42-105 ~]$ docker run -itd -P new-nginx
```

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED              STATUS              PORTS     NAMES
6e301f7fbcc6   new-nginx   "/bin/sh -c '/usr/sb…"   About a minute ago   Up About a minute   80/tcp    competent_leavitt
91beb0537bcc   mynginx     "bash"                   10 minutes ago       Up 10 minutes               confident_neumann
dd605bdb92c6   ubuntu      "bash"                   42 minutes ago       Up 17 minutes               uc-1
[ec2-user@ip-172-31-42-105 ~]$ docker run -itd -P new-nginx
a0c60a8e6a3d099b56ef6b0a28539fc706aecaf4979f61e1faafbb049c294f39
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                     NAMES
a0c60a8e6a3d   new-nginx   "/bin/sh -c '/usr/sb…"   11 seconds ago   Up 11 seconds   0.0.0.0:49153->80/tcp     keen_proskuriakov
a
6e301f7fbcc6   new-nginx   "/bin/sh -c '/usr/sb…"   2 minutes ago    Up 2 minutes    80/tcp                    competent_leavitt
91beb0537bcc   mynginx     "bash"                   11 minutes ago   Up 11 minutes                             confident_neumann
dd605bdb92c6   ubuntu      "bash"                   43 minutes ago   Up 18 minutes                             uc-1
[ec2-user@ip-172-31-42-105 ~]$
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

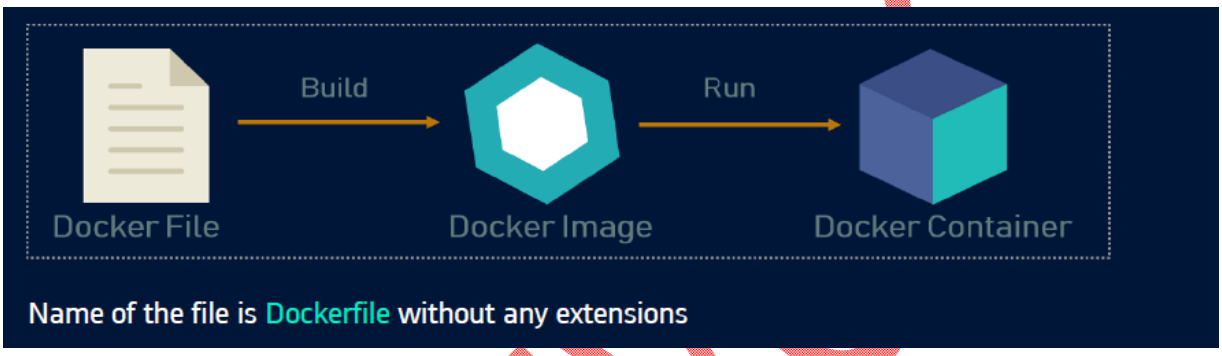
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

```
[ec2-user@ip-172-31-42-105 ~]$ docker exec -it a0c60a8e6a3d sh
# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# cd
# ls
psddevops
# cd psddevops
# ls
devops.txt
# df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay         8.0G  2.1G  6.0G  26% /
tmpfs            64M     0   64M   0% /dev
tmpfs           492M     0  492M   0% /sys/fs/cgroup
shm              64M     0   64M   0% /dev/shm
/dev/xvda1      8.0G  2.1G  6.0G  26% /etc/hosts
tmpfs           492M     0  492M   0% /proc/acpi
tmpfs           492M     0  492M   0% /proc/scsi
tmpfs           492M     0  492M   0% /sys/firmware
# uname -a
Linux a0c60a8e6a3d 4.14.232-176.381.amzn2.x86_64 #1 SMP Wed May 19 00:31:54 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
# uname -r
4.14.232-176.381.amzn2.x86_64
#
```

Dockerfile

A Dockerfile is a text configuration file with specific syntax and pattern. From the dockerfile, docker image is generated. Dockerfile describes step-by-step instruction of all the commands we need to generate a docker image. Dockerfile must be saved with the proper name "Dockerfile", it's case sensitive, so we need to take care of that. We can generate docker image by firing the command "docker build", where the Dockerfile is placed.



Dockerfile Creation

Creating a Dockerfile is as easy as creating a regular file in the system. We can landup into any directory which have proper permission to run the docker command. In the same directory, we can simply create a new file and named it as a Dockerfile.

In the below Code is how the exact Dockerfile looks like.

```
# The line below states we will base our new image on the Latest Official Ubuntu
FROM ubuntu:latest
#
# Identify the maintainer of an image
LABEL maintainer="psddevops@dockerfile.com"
#
# Update the image to the latest packages
RUN apt-get update && apt-get upgrade -y
#
# Install NGINX to test.
RUN apt-get install nginx -y
#
# Expose port 80
```

```
EXPOSE 80
#
# Last is the actual command to start up NGINX within our Container
CMD ["nginx", "-g", "daemon off;"]
```

Create Docker Image from the dockerfile

We can specify the command we need in order to make the image from the docker file. In the image code, I have deployed nginx web-server to run the application. To build an image from the Dockerfile, we need to get into the place where Dockerfile resides.

And then fire the command as shown below, "docker build -t nginx-demo ." Here, -t stands for the tag of the image, which here we have given nginx-demo, and dot(.) represnets the dockerfile in the current directory.

Syntax:

docker build –t <imagename>:<tag-name> .

docker build -t myimage:1 .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t myimage:1 .
Sending build context to Docker daemon  2.048kB
Step 1/6 : FROM ubuntu:latest
latest: Pulling from library/ubuntu
c549ccf8d472: Pull complete
Digest: sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
Status: Downloaded newer image for ubuntu:latest
 ---> 9873176a8ff5
Step 2/6 : LABEL maintainer="learn@dockerfile.com"
 ---> Running in cddedfafc5ed
Removing intermediate container cddedfafc5ed
 ---> 77292d6cfe21
Step 3/6 : RUN apt-get update && apt-get upgrade -y
 ---> Running in 8b539bf52878
```

```
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
Removing intermediate container e7d1ca80e269
 ---> ba4686552d80
Step 5/6 : EXPOSE 80
 ---> Running in 2f50c28cd17d
Removing intermediate container 2f50c28cd17d
 ---> bff155bd1061
Step 6/6 : CMD ["nginx", "-g", "daemon off;"]
 ---> Running in 40bf0f803d7a
Removing intermediate container 40bf0f803d7a
 ---> 66e7797482a6
Successfully built 66e7797482a6
Successfully tagged myimage:1
```

docker images

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
myimage      1         66e7797482a6   About a minute ago   164MB
ubuntu       latest    9873176a8ff5   3 days ago           72.7MB
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

Run second time

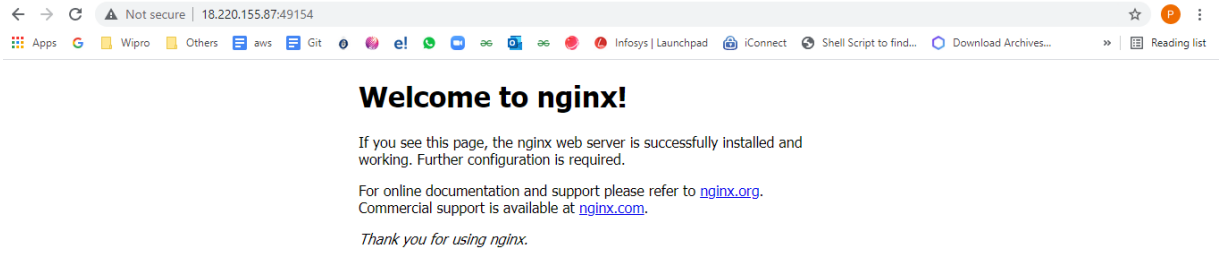
docker build -t myimage:2 .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t myimage:2 .
Sending build context to Docker daemon  2.048kB
Step 1/6 : FROM ubuntu:latest
 ---> 9873176a8ff5
Step 2/6 : LABEL maintainer="learn@dockerfile.com"
 ---> Using cache
 ---> 77292d6cfe21
Step 3/6 : RUN apt-get update && apt-get upgrade -y
 ---> Using cache
 ---> e49bfd3e0584
Step 4/6 : RUN apt-get install nginx -y
 ---> Using cache
 ---> ba4686552d80
Step 5/6 : EXPOSE 80
 ---> Using cache
 ---> bff155bd1061
Step 6/6 : CMD ["nginx", "-g", "daemon off;"]
 ---> Using cache
 ---> 66e7797482a6
Successfully built 66e7797482a6
Successfully tagged myimage:2
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

Create container from custom image

docker run -d --name mynginxcont -P myimage:2

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run -d --name mynginxcont -P myimage:2
7d7cb719112a4bd4126b5cccb02e3707420f45fcf6dd6c0f1f1343c53ae7d28c
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS                     NAMES
7d7cb719112a   myimage:2   "nginx -g 'daemon of…"   7 seconds ago   Up 6 seconds   0.0.0.0:49154->80/tcp   mynginxcont
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```



If dockerfile name is different use below command to create image

docker build -t myimage:3 -f mydockerfile .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t myimage:3 -f mydockerfile .
Sending build context to Docker daemon  2.048kB
Step 1/6 : FROM ubuntu:latest
 ---> 9873176a8ff5
Step 2/6 : LABEL maintainer="learn@dockerfile.com"
 ---> Using cache
 ---> 77292d6cfe21
Step 3/6 : RUN apt-get update && apt-get upgrade -y
 ---> Using cache
 ---> e49bfd3e0584
Step 4/6 : RUN apt-get install nginx -y
 ---> Using cache
 ---> ba4686552d80
Step 5/6 : EXPOSE 80
 ---> Using cache
 ---> bff155bd1061
Step 6/6 : CMD ["nginx", "-g", "daemon off;"]
 ---> Using cache
 ---> 66e7797482a6
Successfully built 66e7797482a6
Successfully tagged myimage:3
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

Dockerfile Instructions

By using Dockerfile we can build the docker images and containers.Before we construct our Dockerfile, you need to understand what makes up the file. This will be a text file, named Dockerfile that includes specific keywords that dictate how to build a specific image. Below specific keywords we can use in our Dpckerfile.

FROM

This instruction is used to set the Base Image for the subsequent instructions. A valid Dockerfile must have FROM as its first instruction.

Syntax

FROM <image>[:tag] [AS <name>]

Ex:

FROM ubuntu

FROM openjdk:8

Note:

Always mention tag don't use latest and also gives optional way to specify platforms lick linux/amd64, linux/arm64, window/arm64 etc.

FROM [--platform=<platform>] <image>[:tag] [AS <name>]

MAINTAINER

Defines full name and email address of the image creator, It's simply defines who wrote the Dockerfile like comment.

Syntax

MAINTAINER <Message>

Ex:

MAINTAINER DevOpsTeam

RUN

- These instructions execute the commands in the created base image container in a new layer.
- Run command runs with /bin/bash -c on Linux and cmd on windows

Syntax:

"RUN <command>"

RUN ["executable", "param1", "param2"]

Ex:

RUN wget <jar-url>

RUN /bin/bash -c 'echo sample'

RUN ["/bin/bash", "-c", "echo sample"]

Commands have two forms

Shell form

> /bin/bash -c 'echo sample'

> java -jar helloworld.jar

Exec form

Represent in square brackets

["/bin/bash", "-c", "echo sample"]

["java", "-jar", " helloworld.jar "]

USER

It will create the user and all instructions are executed under this user.

Syntax:

USER <username>

docker run –it –u ppreddy ubuntu /bin/bash

CMD

Can be used for executing a specific command within the container,CMD command that runs when the container starts.

Syntax:

CMD ["executable", "param-1", "param-2"]

CMD executable param-1 param-2

CMD param-1 param-2

ENTRYPOINT

Sets a default application to be used every time a container is created with the image .ENTRYPOINT command that runs when the container starts

Syntax:

ENTRYPOINT ["executable", "param-1", "param-2"]

ENTRYPOINT executable param-1 param-2

ENV

- This instruction will set the environmental variable for all the subsequent instruction and also in containers Build the image.
- Environmental variables can be replaced while creating container

Syntax:

docker container run –it –e mycontainername=nothing <image name>

ENV myfilename=helloworld.jar

EXPOSE

- This instruction informs Docker that the application listens on the specific network ports at runtime.
- Two Protocols can be specified TCP and UDP. TCP is default

Syntax:

EXPOSE <port> [<port/protocol>]

EXPOSE 8080

VOLUME

- This instruction is used to enable access from the container to a directory on the host machine.
- The VOLUME instruction creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers.
- The value can be a JSON array, VOLUME ["/var/log/"], or a plain string with multiple arguments, such as VOLUME /var/log or VOLUME /var/log /var/db.

Syntax:

VOLUME ["/data"]

Demo

1. Create a Dockerfile with below content.

```
FROM ubuntu
RUN mkdir /myvol
RUN echo "hello world" > /myvol/greeting
VOLUME /myvol
```

2. Verify the images and volumes in the host

docker images

docker volume ls

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker volume ls
DRIVER    VOLUME NAME
```

3. Build the docker image.

   docker build -t volimage:1 .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t volimage:1 .
Sending build context to Docker daemon  3.072kB
Step 1/4 : FROM ubuntu
latest: Pulling from library/ubuntu
c549ccf8d472: Pull complete
Digest: sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
Status: Downloaded newer image for ubuntu:latest
 ---> 9873176a8ff5
Step 2/4 : RUN mkdir /myvol
 ---> Running in 5ca8f81e8673
Removing intermediate container 5ca8f81e8673
 ---> fc7122fd9877
Step 3/4 : RUN echo "hello world" > /myvol/greeting
 ---> Running in 0db98d260c9a
Removing intermediate container 0db98d260c9a
 ---> 26bb49c1460a
Step 4/4 : VOLUME /myvol
 ---> Running in 9c5ed88d39e8
Removing intermediate container 9c5ed88d39e8
 ---> fe187539f806
Successfully built fe187539f806
Successfully tagged volimage:1
```

```
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
volimage     1         fe187539f806   7 minutes ago    72.7MB
ubuntu       latest    9873176a8ff5   5 days ago       72.7MB
[ec2-user@ip-172-31-42-105 ~]$ docker volume ls
DRIVER    VOLUME NAME
[ec2-user@ip-172-31-42-105 ~]$
```

4. Create the container

   docker run --name volcont volimage:1 /bin/bash

   docker volume ls

   cd

   /var/lib/docker/volumes/0c4bdf7bca5c8e523251734e437d36074591b32aee6522526a02bbc41ec927d9/ _data

   cat greeting

```
[ec2-user@ip-172-31-42-105 ~]$ docker run --name volcont volimage:1 /bin/bash
[ec2-user@ip-172-31-42-105 ~]$ docker volume ls
DRIVER    VOLUME NAME
local     0c4bdf7bca5c8e523251734e437d36074591b32aee6522526a02bbc41ec927d9
[ec2-user@ip-172-31-42-105 ~]$ sudo su
[root@ip-172-31-42-105 ec2-user]# cd /var/lib/docker/volumes/0c4bdf7bca5c8e523251734e437d36074591b32aee6522526a02bbc41ec927d9/
[root@ip-172-31-42-105 0c4bdf7bca5c8e523251734e437d36074591b32aee6522526a02bbc41ec927d9]# ll
total 0
drwxr-xr-x 2 root root 22 Jun 23 05:48 _data
[root@ip-172-31-42-105 0c4bdf7bca5c8e523251734e437d36074591b32aee6522526a02bbc41ec927d9]# cd _data/
[root@ip-172-31-42-105 _data]# ll
total 4
-rw-r--r-- 1 root root 12 Jun 23 05:39 greeting
[root@ip-172-31-42-105 _data]# cat greeting
hello world
[root@ip-172-31-42-105 _data]#
```

WORKDIR

- The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile. If work directory does not exist, it will be created by default.
- We can use WORKDIR multiple times in a Dockerfile.

Syntax:

WORKDIR /path/to/workdir

Demo

Create docker file with below content and build the image and containers.

```
FROM ubuntu
WORKDIR /a
WORKDIR b
WORKDIR c
RUN pwd
```

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t wdimage:1 .
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM ubuntu
latest: Pulling from library/ubuntu
c549ccf8d472: Pull complete
Digest: sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
Status: Downloaded newer image for ubuntu:latest
 ---> 9873176a8ff5
Step 2/5 : WORKDIR /a
 ---> Running in 454c0a6ea8a6
Removing intermediate container 454c0a6ea8a6
 ---> 64850942b6d0
Step 3/5 : WORKDIR b
 ---> Running in ae3ba5091b98
Removing intermediate container ae3ba5091b98
 ---> 7392cbca498b
Step 4/5 : WORKDIR c
 ---> Running in 7a9ae9d4e3ea
Removing intermediate container 7a9ae9d4e3ea
 ---> 1d4201fada20
Step 5/5 : RUN pwd
 ---> Running in b8aff7792899
/a/b/c
Removing intermediate container b8aff7792899
 ---> 972f911d8a21
Successfully built 972f911d8a21
Successfully tagged wdimage:1
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run -it wdimage:1 /bin/bash
root@2e1db5192c5b:/a/b/c#
```

LABEL

The LABEL instruction adds metadata to an image. A LABEL is a key-value pair. To include spaces within a LABEL value, use quotes and backslashes as you would in command-line parsing.

Syntax:

LABEL <key>=<value> <key>=<value> <key>=<value> ...

Demo

```
FROM ubuntu
```

```
LABEL "com.example.vendor"="ACME Incorporated"
LABEL com.example.label-with-value="foo"
LABEL version="1.0"
LABEL description="This text illustrates \
that label-values can span multiple lines."
```

docker build -t labelimage:1 .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t labelimage:1 .
Sending build context to Docker daemon  15.87kB
Step 1/5 : FROM ubuntu
 ---> 9873176a8ff5
Step 2/5 : LABEL "com.example.vendor"="ACME Incorporated"
 ---> Running in ad8e1ef64d7f
Removing intermediate container ad8e1ef64d7f
 ---> 6fa978ddec43
Step 3/5 : LABEL com.example.label-with-value="foo"
 ---> Running in bd0856ccff6b
Removing intermediate container bd0856ccff6b
 ---> 4d7510af53a7
Step 4/5 : LABEL version="1.0"
 ---> Running in 75784ba2e1c2
Removing intermediate container 75784ba2e1c2
 ---> 1977a1312519
Step 5/5 : LABEL description="This text illustrates that label-values can span multiple lines."
 ---> Running in c1123579d744
Removing intermediate container c1123579d744
 ---> 37886919acdb
Successfully built 37886919acdb
Successfully tagged labelimage:1
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

docker run labelimage:1 /bin/bash

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
labelimage   1         37886919acdb   About a minute ago   72.7MB
wdimage      1         972f911d8a21   26 minutes ago       72.7MB
ubuntu       latest    9873176a8ff5   5 days ago           72.7MB
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run labelimage:1 /bin/bash
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS                      PORTS     NAMES
5ed901b96afa   labelimage:1   "/bin/bash"   12 seconds ago   Exited (0) 10 seconds ago             romantic_taussig
2e1db5192c5b   wdimage:1      "/bin/bash"   26 minutes ago   Exited (0) 25 minutes ago             admiring_keller
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker inspect 5ed901b96afa
```

docker inspect 5ed901b96afa

```
    "OnBuild": null,
    "Labels": {
        "com.example.label-with-value": "foo",
        "com.example.vendor": "ACME Incorporated",
        "description": "This text illustrates that label-values can span multiple lines.",
        "version": "1.0"
    }
},
```

ADD and COPY:

- Both ADD and COPY instructions are used to copy the files into docker image.
- ADD instruction supports copying files from url and local machines
- COPY instruction supports only copying files from local machines

Syntax:

ADD <src><dest>

COPY <src><dest>

ADD Demo

Create the docker file with below content

```
FROM openjdk:8
LABEL author="ppnreddy"
LABEL version="0.3"
LABEL project="PSDDEVOPS"
ADD https://referenceappkhaja.s3-us-west-2.amazonaws.com/spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar /spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar
EXPOSE 8080
ENTRYPOINT ["java"]
CMD ["-jar", "/spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar"]
```

docker build -t addimage:1 .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t addimage:1 .
Sending build context to Docker daemon  15.87kB
Step 1/8 : FROM openjdk:8
8: Pulling from library/openjdk
d960726af2be: Pull complete
e8d62473a22d: Pull complete
8962bc0fad55: Pull complete
65d943ee54c1: Pull complete
da20b77f10ac: Pull complete
fb6a778e6477: Pull complete
ae7884f0e61b: Pull complete
Digest: sha256:fad3fc6bce4fe7b372b61259537decfed5b432b8f4c4b30866f8cdd7c1ed84d2
Status: Downloaded newer image for openjdk:8
 ---> eca41db787bd
Step 2/8 : LABEL author="ppnreddy"
 ---> Running in e3bfe57a3260
Removing intermediate container e3bfe57a3260
 ---> 41e399e13cbb
Step 3/8 : LABEL version="0.3"
 ---> Running in 3cfd3892349b
Removing intermediate container 3cfd3892349b
 ---> f45f196b2b72
Step 4/8 : LABEL project="PSDDEVOPS"
 ---> Running in b830c8829b76
Removing intermediate container b830c8829b76
 ---> 523e82e2b3f7
Step 5/8 : ADD https://referenceappkhaja.s3-us-west-2.amazonaws.com/spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar  /spring-petclin
ic-2.2.0.BUILD-SNAPSHOT.jar
Downloading [==================================================>]  47.65MB/47.65MB
 ---> 26fe331c92b2
Step 6/8 : EXPOSE 8080
 ---> Running in 7aa3fb2315da
Removing intermediate container 7aa3fb2315da
 ---> d8f41a729719
Step 7/8 : ENTRYPOINT ["java"]
 ---> Running in 0903a771d77e
Removing intermediate container 0903a771d77e
 ---> 9ab4937ea3e8
Step 8/8 : CMD ["-jar", "/spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar"]
```

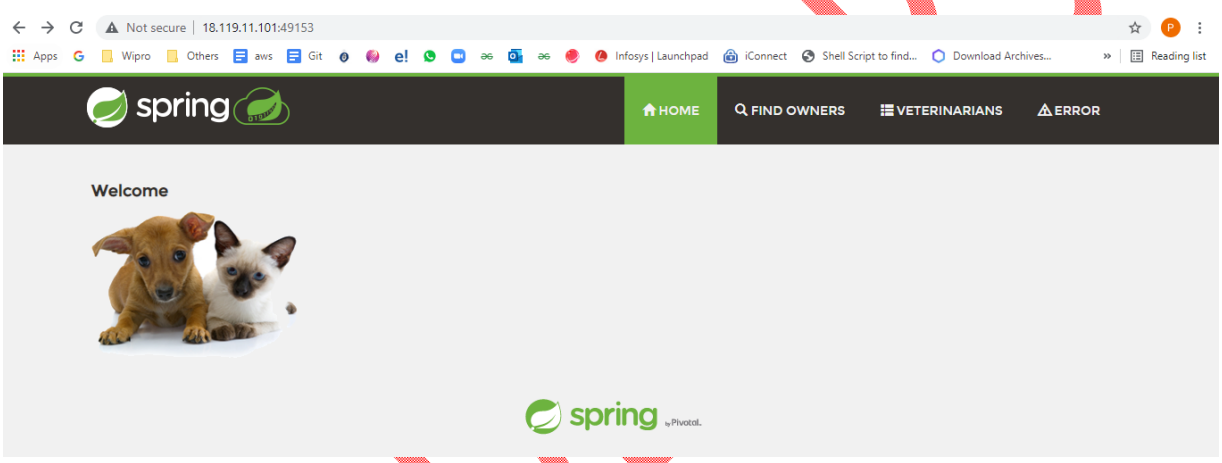
docker images

docker run -d --name addcont -P addimage:1

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED             SIZE
addimage      1         a0e9a84f1851   About a minute ago   562MB
labelimage    1         37886919acdb   14 minutes ago       72.7MB
wdimage       1         972f911d8a21   39 minutes ago       72.7MB
ubuntu        latest    9873176a8ff5   5 days ago           72.7MB
openjdk       8         eca41db787bd   6 weeks ago          514MB
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run -d --name addcont -P addimage:1
bd7f5e916c5c11d75c1c034ccfbf3578d428de02eadd4c5ff3ac7e5265f57f8f
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS                     NAMES
bd7f5e916c5c   addimage:1    "java -jar /spring-p…"   21 seconds ago   Up 20 seconds   0.0.0.0:49153->8080/tcp   addcont
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

docker exec -it bd7f5e916c5c /bin/bash

```
bd7f5e916c5c11d75c1c034ccfbf3578d428de02eadd4c5ff3ac7e5265f57f8f
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS                     NAMES
bd7f5e916c5c   addimage:1    "java -jar /spring-p…"   21 seconds ago   Up 20 seconds   0.0.0.0:49153->8080/tcp   addcont
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker exec -it bd7f5e916c5c /bin/bash
root@bd7f5e916c5c:/# ls -lrt
total 46536
-rw-------    1 root root 47650416 Jun 10  2020 spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar
drwxr-xr-x   2 root root        6 Mar 19 23:44 home
drwxr-xr-x   2 root root        6 Mar 19 23:44 boot
drwxr-xr-x   1 root root       19 May 11 00:00 var
drwxr-xr-x   1 root root       19 May 11 00:00 usr
drwxr-xr-x   2 root root        6 May 11 00:00 srv
drwxr-xr-x   3 root root       30 May 11 00:00 run
drwxr-xr-x   2 root root        6 May 11 00:00 opt
drwxr-xr-x   2 root root        6 May 11 00:00 mnt
drwxr-xr-x   2 root root        6 May 11 00:00 media
drwxr-xr-x   2 root root       34 May 11 00:00 lib64
drwxr-xr-x   1 root root       30 May 12 01:55 lib
drwxr-xr-x   1 root root       20 May 12 01:56 sbin
drwxr-xr-x   1 root root      179 May 12 06:49 bin
drwx------   1 root root       24 May 12 06:52 root
drwxr-xr-x   1 root root       66 Jun 23 06:44 etc
dr-xr-xr-x 140 root root        0 Jun 23 06:44 proc
dr-xr-xr-x  13 root root        0 Jun 23 06:44 sys
drwxr-xr-x   5 root root      340 Jun 23 06:44 dev
drwxrwxrwt   1 root root      115 Jun 23 06:44 tmp
root@bd7f5e916c5c:/#
```



COPY Demo

Create the docker file with below content

```
FROM openjdk:8
LABEL author="ppreddy"
LABEL version="1.0"
LABEL project="PSDDEVOPS"
COPY spring-petclinic.jar  /spring-petclinic.jar
EXPOSE 8080
ENTRYPOINT ["java"]
CMD ["-jar",  "/spring-petclinic.jar"]
```

ls –lrt

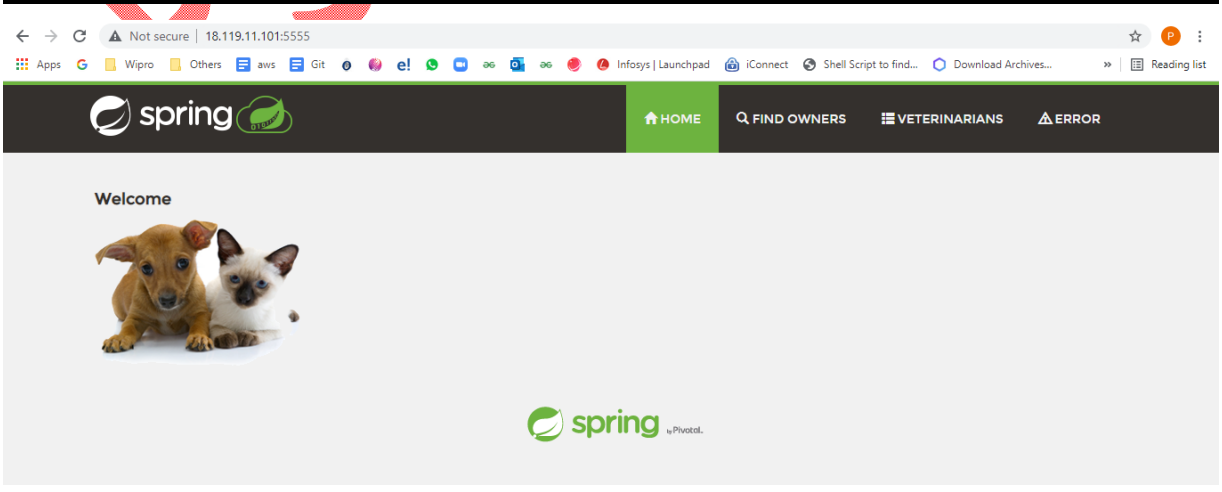
```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ ls -lrt
total 46540
-rw-rw-r-- 1 ec2-user ec2-user 47650416 Jun 10  2020 spring-petclinic.jar
-rw-rw-r-- 1 ec2-user ec2-user      204 Jun 23 06:53 Dockerfile
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

docker build -t copyimage:1 .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t copyimage:1 .
Sending build context to Docker daemon  47.67MB
Step 1/8 : FROM openjdk:8
 ---> eca41db787bd
Step 2/8 : LABEL author="ppreddy"
 ---> Using cache
 ---> 44959c03f721
Step 3/8 : LABEL version="1.0"
 ---> Using cache
 ---> 5831bdf26a12
Step 4/8 : LABEL project="PSDDEVOPS"
 ---> Using cache
 ---> fbb452b5c8f8
Step 5/8 : COPY spring-petclinic.jar  /spring-petclinic.jar
 ---> 32d501cc3fcc
Step 6/8 : EXPOSE 8080
 ---> Running in 80b2329ce5ba
Removing intermediate container 80b2329ce5ba
 ---> e00af3a2a0ad
Step 7/8 : ENTRYPOINT ["java"]
 ---> Running in 3f03e8e21a79
Removing intermediate container 3f03e8e21a79
 ---> 3c4e21ba2dac
Step 8/8 : CMD ["-jar",  "/spring-petclinic.jar"]
 ---> Running in 3369ef9ca79d
Removing intermediate container 3369ef9ca79d
 ---> 65f9df8f58d1
Successfully built 65f9df8f58d1
Successfully tagged copyimage:1
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

docker run -p 5555:8080 copyimage:1

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run -p 5555:8080 copyimage:1

              |\      _,,,--,,_
             /,`.-'`'   ._  \-;;,_
  _______ __|,4-  ) )_   .;.(  `'-'
 |       | '---''(_/._)-'(_\_)   |   |    |   |  |  | |   |       |
 |    _  |    ___|_     _|       |   |    |   |  |_| |   |       | __ _ _
 |   |_| |   |___  |   | |       |   |    |   |       |   |       | \ \ \ \
 |    ___|    ___| |   | |      _|   |___ |   | _     |   |       |  \ \ \ \
 |   |   |   |___  |   | |     |_|       ||   ||_|    |   |_      |_   ) ) ) )
 |___|   |_______| |___| |_______|_______||___|   |___|___|       |  / / / /
 ==========================================================================/_/_/_/

:: Built with Spring Boot :: 2.2.1.RELEASE


2021-06-23 06:59:23.629  INFO 1 --- [           main] o.s.s.petclinic.PetClinicApplication     : Starting PetClinicApplication
 v2.2.0.BUILD-SNAPSHOT on 7d23eb385efe with PID 1 (/spring-petclinic.jar started by root in /)
2021-06-23 06:59:23.642  INFO 1 --- [           main] o.s.s.petclinic.PetClinicApplication     : No active profile set, fallin
g back to default profiles: default
2021-06-23 06:59:26.446  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data rep
ositories in DEFAULT mode.
2021-06-23 06:59:26.611  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data reposito
```



……. Prepared by PPNREDDY

ARG

- While building the image. Values of ARG will be changed
- ARG instruction allows variables to be passed while building image
- ARG is available only while building the docker image
- While building images we need some customizations we go with ARG

Syntax:

ARG <key>=<value>

Demo

Create Dockerfile with below content

```
FROM openjdk:8
ARG url=https://referenceappkhaja.s3-us-west-2.amazonaws.com/spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar
RUN wget ${url}
EXPOSE 8080
CMD ["java", "-jar", "spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar"]
```

docker build -t argimage:1 .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t argimage:1 .
Sending build context to Docker daemon  47.67MB
Step 1/5 : FROM openjdk:8
 ---> eca41db787bd
Step 2/5 : ARG url=https://referenceappkhaja.s3-us-west-2.amazonaws.com/spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar
 ---> Running in eb238d50e8e1
Removing intermediate container eb238d50e8e1
 ---> 062e00d6d12c
Step 3/5 : RUN wget ${url}
 ---> Running in 900afeb6e88f
--2021-06-23 07:39:17--  https://referenceappkhaja.s3-us-west-2.amazonaws.com/spring-petclinic-2.2.0.BUILD-SNAPSHOT.jar
Resolving referenceappkhaja.s3-us-west-2.amazonaws.com (referenceappkhaja.s3-us-west-2.amazonaws.com)... 52.218.246.225
Connecting to referenceappkhaja.s3-us-west-2.amazonaws.com (referenceappkhaja.s3-us-west-2.amazonaws.com)|52.218.246.225|:443.
.. connected.
```

CMD Vs ENTRYPOINT

Both of these docker instructions are used to define command/executables which is executed during container invocation. These are very similar but different instructions which can be used independently or together to achieve better flexibility to define what a container should execute. There are two ways/syntaxes to define them:

Exec form: (Preferred form)

CMD [“executable”, “arg1”, “arg2”]

ENTRYPOINT [“executable”, “arg1”, “arg2”]

Shell form:

CMD executable arg1 arg2

ENTRYPOINT executable arg1 arg2

CMD

Used to provide all the default scenarios which can be overridden

Default executable

This instruction is used to define a default executable for a container to execute. If you want to create a generic docker image, where users can pass any supported command to be executed on container invocation, then this instruction is the one to use. Entrypoint instruction should not be defined in Dockerfile for this use case.

CMD ["executable", "arg1", "arg2"]

Default arguments

This instruction can also be used to provide default arguments. For this use case, we don't specify executable in this instruction at all, but simply define some arguments which are used as default/additional arguments for executable defined in the entrypoint instruction. Thus, entrypoint instruction is required in dockerfile for this use case to define an executable.

CMD ["arg1", "arg2"]

ENTRYPOINT ["executable"]

Note: Anything defined in CMD can be overridden by passing arguments in docker run command.

ENTRYPOINT

Used to define specific executable and arguments to be executed during container invocation which cannot be overridden.This is used to constraint the user to execute anything else. User can however define arguments to be passed in the executable by adding them in the docker run command.

ENTRYPOINT ["executable", "arg1", "arg2"]

Example Demos

For the demo, let's take the following Dockerfile which fetches necessary cowsay and screenfetch libraries so that we can run these commands when running the container.

Demo 1: Use of CMD to define Default executable

```
FROM debian:jessie-slim
RUN apt-get update                                  && \
    apt-get install -y --no-install-recommends          \
        cowsay                                          \
        screenfetch                                 && \
    rm -rf /var/lib/apt/lists/*
ENV PATH "$PATH:/usr/games"
CMD ["cowsay", "Yo, CMD !!"]
```

Build the Image:

docker build -t cmddemoimage:latest .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t cmddemoimage:latest .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM debian:jessie-slim
 ---> 2045588e2542
Step 2/4 : RUN apt-get update                                   &&     apt-get install -y --no-install-recommends
          cowsay                                                   screenfetch                                        &&     r
m -rf /var/lib/apt/lists/*
 ---> Using cache
 ---> 58f1423817e3
Step 3/4 : ENV PATH "$PATH:/usr/games"
 ---> Using cache
 ---> ae97b4ca4f22
Step 4/4 : CMD ["cowsay", "Yo, CMD !!"]
 ---> Using cache
 ---> 4ac42b529d34
Successfully built 4ac42b529d34
Successfully tagged cmddemoimage:latest
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
testentryimage      1                   23bdd0adf661        25 minutes ago      115MB
cmddemoimage        latest              4ac42b529d34        28 minutes ago      115MB
testimage           1                   4ac42b529d34        28 minutes ago      115MB
new-nginx           latest              6f5aec872df2        6 hours ago         223MB
mynginx             latest              34a8417bf94f        6 hours ago         223MB
ubuntu              latest              9873176a8ff5        3 days ago          72.7MB
debian              jessie-slim         2045588e2542        2 months ago        81.4MB
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

Run the container:

docker run cmddemoimage:latest

This will run the default executable specified in CMD instruction and output following.

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run cmddemoimage:latest
 _____
< Yo, CMD !! >
 -----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

However, you can override this executable, when running the container as

docker run cmddemoimage:latest screenfetch -E which will output

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run cmddemoimage:latest screenfetch -E
          _,met$$$$$gg.          @44e55d8d770a
       ,g$$$$$$$$$$$$$$$P.       OS: Debian
     ,g$$P"         """Y$$.".    Kernel: x86_64 Linux 4.14.232-176.381.amzn2.x86_64
    ,$$P'               `$$$.    Uptime: 8h 30m
   ',$$P       ,ggs.     `$$b:   Packages: 106
   `d$$'     ,$P"'   .    $$$    Shell:
    $$P      d$'     ,    $$P    CPU: Intel Xeon CPU E5-2676 v3 @ 2.4GHz
    $$:      $$.   -    ,d$$'    RAM: 338MB / 983MB
    $$\;      Y$b._   _,d$P'
    Y$$.    `.`"Y$$$$P"'
    `$$b      "-.__
     `Y$$
      `Y$$.
        `$$b.
          `Y$$b.
            `"Y$b._
                `"""

[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

Demo 2: Use of ENTRYPOINT

If you want to restrict users from passing any other executable, you can specify 'ENTRYPOINT' instruction as

```
FROM debian:jessie-slim
RUN apt-get update                                  && \
    apt-get install -y --no-install-recommends      \
        cowsay                                      \
        screenfetch                                 &&    \
    rm -rf /var/lib/apt/lists/*
ENV PATH "$PATH:/usr/games"
ENTRYPOINT ["cowsay", "Yo, Entrypoint!!"]
```

Build the Image:

docker build -t entrypointdemoimage:latest .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t entrypointdemoimage:latest .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM debian:jessie-slim
 ---> 2045588e2542
Step 2/4 : RUN apt-get update                                    &&     apt-get install -y --no-install-recommends
       cowsay                                                  screenfetch                                &&          rm -rf /va
r/lib/apt/lists/*
 ---> Using cache
 ---> cbf6e06a5c35
Step 3/4 : ENV PATH "$PATH:/usr/games"
 ---> Using cache
 ---> 41525ea79782
Step 4/4 : ENTRYPOINT ["cowsay", "Yo, Entrypoint!!"]
 ---> Using cache
 ---> 23bdd0adf661
Successfully built 23bdd0adf661
Successfully tagged entrypointdemoimage:latest
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker images
REPOSITORY            TAG           IMAGE ID       CREATED          SIZE
entrypointdemoimage   latest        23bdd0adf661   35 minutes ago   115MB
testentryimage        1             23bdd0adf661   35 minutes ago   115MB
cmddemoimage          latest        4ac42b529d34   39 minutes ago   115MB
testimage             1             4ac42b529d34   39 minutes ago   115MB
new-nginx             latest        6f5aec872df2   6 hours ago      223MB
mynginx               latest        34a8417bf94f   7 hours ago      223MB
ubuntu                latest        9873176a8ff5   3 days ago       72.7MB
debian                jessie-slim   2045588e2542   2 months ago     81.4MB
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

Run the container:

docker run entrypointdemoimage:latest

This will run the executable specified in ENTRYPOINT and output following.

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run entrypointdemoimage:latest
 _________________
< Yo, Entrypoint!! >
 -----------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

With ENTRYPOINT defined in the docker file, if you pass an executable in the docker run command, it will not take that as an executable, but, as an argument for the executable defined in the ENTRYPOINT instruction. So if you run following command.

docker run entrypointdemoimage:latest screenfetch -E

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run entrypointdemoimage:latest screenfetch -E
 _____________________________________
< Yo, Entrypoint!! screenfetch -E >
 -------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

As expected, the arguments passed in the docker run command, is appended in the executable and argument, set in the entryPoint instruction.

Demo 3: Use of ENTRYPOINT and CMD together

If ENTRYPOINT is defined, anything defined in CMD will be taken as arguments for the executable defined in ENTRYPOINT.

```
FROM debian:jessie-slim

RUN apt-get update                                  && \
    apt-get install -y --no-install-recommends        \
        cowsay                                        \
        screenfetch                            &&     \
    rm -rf /var/lib/apt/lists/*

ENV PATH "$PATH:/usr/games"
CMD ["Yo, CMD!!"]
ENTRYPOINT ["cowsay", "Yo, Entrypoint!!"]
```

Build the Image:

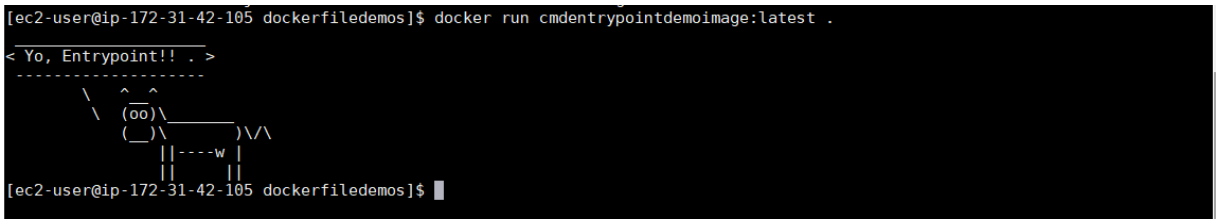
docker build -t cmdentrypointdemoimage:latest .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t cmdentrypointdemoimage:latest .
Sending build context to Docker daemon  2.048kB
Step 1/5 : FROM debian:jessie-slim
 ---> 2045588e2542
Step 2/5 : RUN apt-get update                                  &&     apt-get install -y --no-install-recommends
        cowsay                                                screenfetch                                &&          rm -rf /va
r/lib/apt/lists/*
 ---> Using cache
 ---> cbf6e06a5c35
Step 3/5 : ENV PATH "$PATH:/usr/games"
 ---> Using cache
 ---> 41525ea79782
Step 4/5 : CMD ["Yo, CMD!!"]
 ---> Running in 4b7023cfcd6d
Removing intermediate container 4b7023cfcd6d
 ---> 84a77ea2d821
Step 5/5 : ENTRYPOINT ["cowsay", "Yo, Entrypoint!!"]
 ---> Running in 2959637f9d56
Removing intermediate container 2959637f9d56
 ---> a0dbd4542077
Successfully built a0dbd4542077
Successfully tagged cmdentrypointdemoimage:latest
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

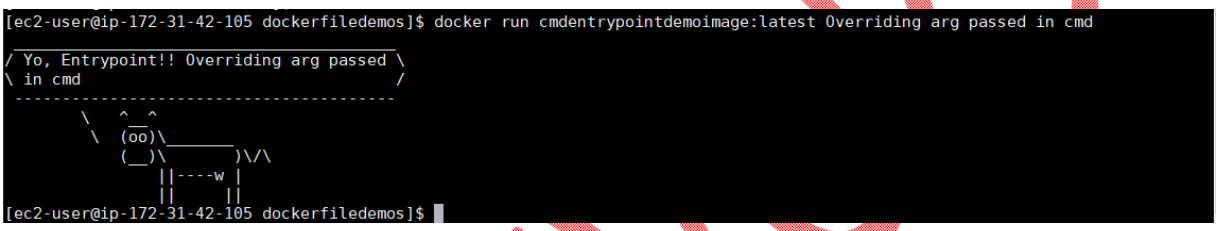
```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker images
REPOSITORY              TAG           IMAGE ID       CREATED             SIZE
cmdentrypointdemoimage  latest        a0dbd4542077   About a minute ago  115MB
entrypointdemoimage     latest        23bdd0adf661   50 minutes ago      115MB
testentryimage          1             23bdd0adf661   50 minutes ago      115MB
cmddemoimage            latest        4ac42b529d34   53 minutes ago      115MB
testimage               1             4ac42b529d34   53 minutes ago      115MB
new-nginx               latest        6f5aec872df2   7 hours ago         223MB
mynginx                 latest        34a8417bf94f   7 hours ago         223MB
ubuntu                  latest        9873176a8ff5   3 days ago          72.7MB
debian                  jessie-slim   2045588e2542   2 months ago        81.4MB
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

Run the container:

docker run cmdentrypointdemoimage:latest .



As you now know, the argument defined in CMD can easily be overridden by passing other arguments in docker run command, if you execute *docker run cmdentrypointdemoimage:latest Overriding arg passed in cmd*, It will output:



Thus, the main point to remember is CMD is used to provide default executable and arguments, which can be overridden, while ENTRYPOINT is used to specify specific executable and arguments, to constraint the usage of image.

Dockerfile for ubuntu and Tomcat installation

Copy the below content in Dockerfile

```
# PULL the base image from dockerhub
FROM ubuntu:latest

# Author name
MAINTAINER PSDDEVOPS

# Install the java and git
RUN apt-get update && \
    apt-get install -y openjdk-8-jdk && \
    apt-get install -y ant && \
        apt-get install git -y && \
    apt-get clean;
```

```
# Fix certificate issues
RUN apt-get update && \
    apt-get install ca-certificates-java && \
    apt-get clean && \
    update-ca-certificates -f;

# Download the maven
ADD  https://downloads.apache.org/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz /

# Download the tomcat
ADD    https://downloads.apache.org/tomcat/tomcat-8/v8.5.68/bin/apache-tomcat-8.5.68.tar.gz /

# Extracting maven and tomcat
RUN tar -xvzf apache-maven-3.6.3-bin.tar.gz && \
    mv apache-maven-3.6.3 maven && \
    tar -xvzf apache-tomcat-8.5.68.tar.gz && \
        mv apache-tomcat-8.5.68 tomcat && \
        rm apache-maven-3.6.3-bin.tar.gz apache-tomcat-8.5.68.tar.gz

# Setting maven path
ENV MAVEN_HOME /maven
RUN export MAVEN_HOME
ENV PATH $PATH:$MAVEN_HOME/bin
RUN export PATH

# Clone the git url
ARG git_url=https://github.com/psddevops/sampletest.git

# Build and copy the war file to tomcat
RUN git clone ${git_url} && cd sampletest && mvn clean package && \
    cp /sampletest/target/psdapp.war /tomcat/webapps/
```

```
# Exposing the port
EXPOSE 8080

# Starting tomcat server
CMD ["/tomcat/bin/catalina.sh", "run"]
```

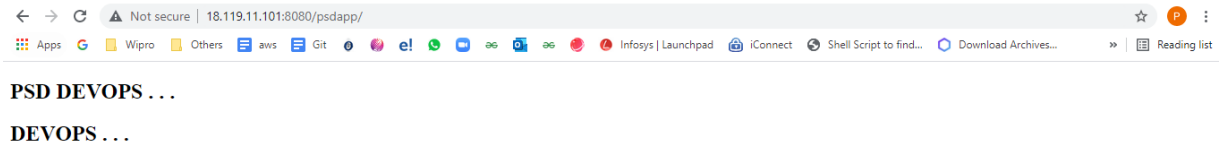
docker build -t mytommyimage:1.0 .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t mytommyimage:1.0 .
Sending build context to Docker daemon  47.69MB
Step 1/14 : FROM ubuntu:latest
 ---> 9873176a8ff5
Step 2/14 : MAINTAINER PSDDEVOPS
 ---> Using cache
 ---> 80d85152da6b
Step 3/14 : RUN apt-get update &&     apt-get install -y openjdk-8-jdk &&     apt-get install -y ant &&         apt-get instal
l git -y &&     apt-get clean;
 ---> Using cache
 ---> f17af63e0ee2
Step 4/14 : ADD https://downloads.apache.org/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz /
Downloading [==================================================>]  9.506MB/9.506MB
 ---> Using cache
 ---> a17ab8b57342
Step 5/14 : ADD https://downloads.apache.org/tomcat/tomcat-8/v8.5.68/bin/apache-tomcat-8.5.68.tar.gz /
Downloading [==================================================>]  10.56MB/10.56MB
 ---> Using cache
 ---> a81db3487850
Step 6/14 : RUN tar -xvzf apache-maven-3.6.3-bin.tar.gz &&     mv apache-maven-3.6.3 maven &&     tar -xvzf apache-tomcat-8.5.
68.tar.gz &&    mv apache-tomcat-8.5.68 tomcat &&       rm apache-maven-3.6.3-bin.tar.gz apache-tomcat-8.5.68.tar.gz
 ---> Running in 34f31299a63d
apache-maven-3.6.3/README.txt
apache-maven-3.6.3/LICENSE
apache-maven-3.6.3/NOTICE
apache-maven-3.6.3/lib/
apache-maven-3.6.3/lib/cdi-api.license
apache-maven-3.6.3/lib/commons-cli.license
apache-maven-3.6.3/lib/commons-io.license
apache-maven-3.6.3/lib/commons-lang3.license
```

docker run -it -d --name cicdconatainer -p 8080:8080 mytommyimage:1.0

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run -it -d --name cicdconatainer -p 8080:8080 mytommyimage:1.0
df0756bb32305693fbe3ff8e6156e11b8255c8b468f28392d31d9c93a9327b04
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS                    NAMES
df0756bb3230   mytommyimage:1.0   "/tomcat/bin/catalin…"   6 seconds ago   Up 4 seconds   0.0.0.0:8080->8080/tcp   cicdconatai
ner
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run -it -d --name cicdconatainer1 -p 8081:8080 mytommyimage:1.0
b7e527077ab5c2b3d87f54bbb33ef69939373200d834c55366ae1745adf38514
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                    NAMES
b7e527077ab5   mytommyimage:1.0   "/tomcat/bin/catalin…"   4 seconds ago    Up 3 seconds    0.0.0.0:8081->8080/tcp   cicdconat
ainer1
df0756bb3230   mytommyimage:1.0   "/tomcat/bin/catalin…"   32 seconds ago   Up 30 seconds   0.0.0.0:8080->8080/tcp   cicdconat
ainer
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

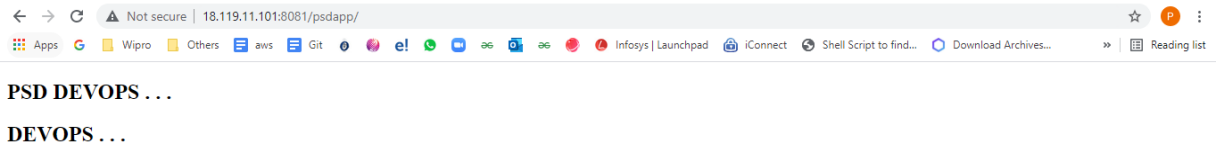
http://18.119.11.101:8080/psdapp/



PSD DEVOPS . . .

DEVOPS . . .

http://18.119.11.101:8081/psdapp/



Dockerfile for CentOS and Tomcat installation

Copy the below content in Dockerfile

```
FROM centos:latest

MAINTAINER ppreddy@psddevops.com

RUN mkdir /opt/tomcat/

WORKDIR /opt/tomcat

RUN curl -O https://downloads.apache.org/tomcat/tomcat-8/v8.5.68/bin/apache-tomcat-8.5.68.tar.gz
RUN tar xvfz apache*.tar.gz
RUN mv apache-tomcat-8.5.68/* /opt/tomcat/.
RUN yum -y install java
RUN java -version

WORKDIR /opt/tomcat/webapps
RUN curl -O -L https://github.com/psddevops/jenkins_pipelines/blob/master/psdapp.war

EXPOSE 8080

CMD ["/opt/tomcat/bin/catalina.sh", "run"]
```


docker build -t centostomcatimage:1 -f Centosdockerfile .

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker build -t centostomcatimage:1 -f Centosdockerfile .
Sending build context to Docker daemon  47.71MB
Step 1/13 : FROM centos:latest
latest: Pulling from library/centos
7a0437f04f83: Pull complete
Digest: sha256:5528e8b1b1719d34604c87e11dcd1c0a20bedf46e83b5632cdeac91b8c04efc1
Status: Downloaded newer image for centos:latest
 ---> 300e315adb2f
Step 2/13 : MAINTAINER ppreddy@psddevops.com
 ---> Running in ed5e64e6405d
Removing intermediate container ed5e64e6405d
 ---> ab7f02c38863
Step 3/13 : RUN mkdir /opt/tomcat/
 ---> Running in 01bdfed2265a
Removing intermediate container 01bdfed2265a
 ---> 9e2cc9a13091
Step 4/13 : WORKDIR /opt/tomcat
 ---> Running in 1d6731c56872
Removing intermediate container 1d6731c56872
 ---> d2bfa073b4a2
Step 5/13 : RUN curl -O https://downloads.apache.org/tomcat/tomcat-8/v8.5.68/bin/apache-tomcat-8.5.68.tar.gz
 ---> Running in 5b6dccec6190
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 10.0M  100 10.0M    0     0  5514k      0  0:00:01  0:00:01 --:--:-- 5511k
```

docker run -it -d --name mycentoscont -p 8080:8080 centostomcatimage:1

```
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED             SIZE
centostomcatimage   1         159638f65ba1   About a minute ago   479MB
mytommyimage        1.0       308255179973   19 minutes ago       703MB
ubuntu              latest    9873176a8ff5   5 days ago           72.7MB
centos              latest    300e315adb2f   6 months ago         209MB
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run -it -d --name mycentoscont -p 8080:8080 centostomcatimage:1
e434cf9409e15f831a599b127773f97a784c7986613eabae7902df632cf74b09
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ dockers ps
-bash: dockers: command not found
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                    NAMES
e434cf9409e1   centostomcatimage:1   "/opt/tomcat/bin/cat…"   18 seconds ago   Up 17 seconds   0.0.0.0:8080->8080/tcp   mycent
oscont
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker run -it -d --name mycentoscont1 -p 8081:8080 centostomcatimage:1
b3ea34ed0a8de81dbbdb5bc5d7ef3c4c30a87dc15fe934df277452d51839b10a
[ec2-user@ip-172-31-42-105 dockerfiledemos]$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                    NAMES
b3ea34ed0a8d   centostomcatimage:1   "/opt/tomcat/bin/cat…"   4 seconds ago    Up 3 seconds    0.0.0.0:8081->8080/tcp   mycent
oscont1
e434cf9409e1   centostomcatimage:1   "/opt/tomcat/bin/cat…"   48 seconds ago   Up 46 seconds   0.0.0.0:8080->8080/tcp   mycent
oscont
[ec2-user@ip-172-31-42-105 dockerfiledemos]$
```

How can we transfer the docker images?

➢ Push the images to a Repository & Pull it on required machines (using Docker Push & Pull Commands)

✓ Private Repo (Docker Trusted Registry DTR)

✓ Public Repo (Docker Hub)

(OR)

Save it as tar file & SCP to the required machines (not suggested)

To push the image to docker hub first we need to run "docker login" command from our terminal/client machine then we need to enter the username and password of the docker hub account (if you don't have docker hub account then first signup for docker hub account).

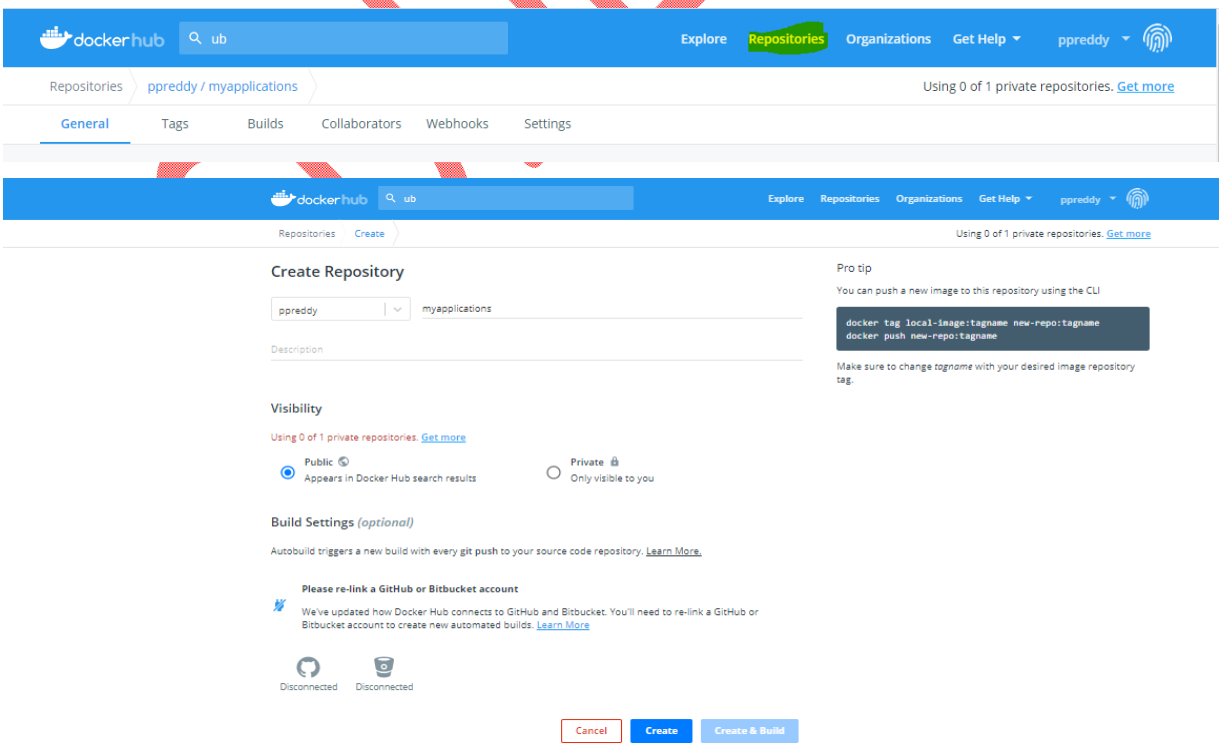
Note: We need to tag our image with our repo name to push them to our repo and the repo name we can get the dockerhub account with what id we have created the account.

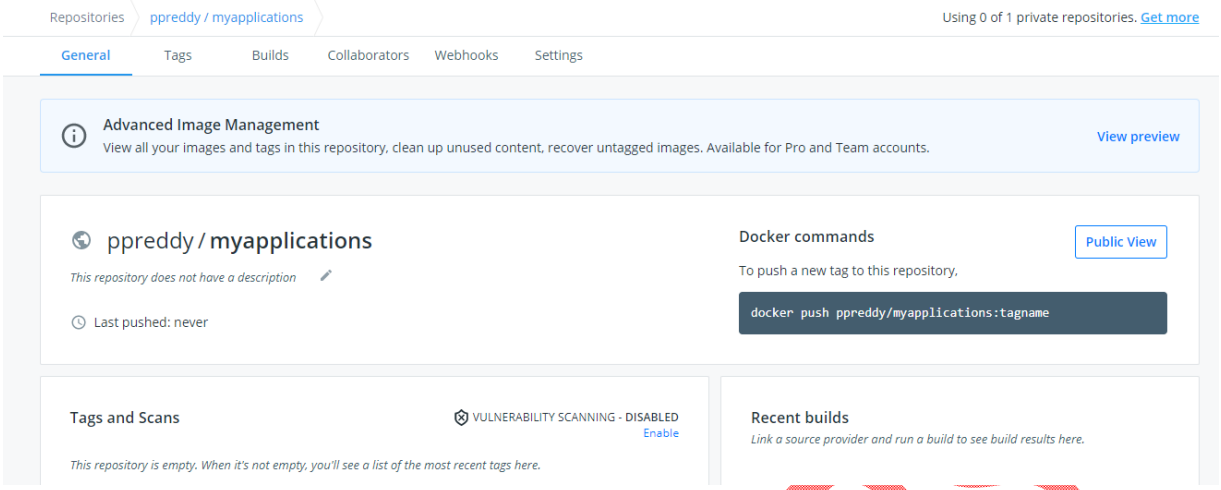
Importance of tag: see in below snap we do have two ubuntu image, one is latest version and other is 18.04, I had run the command docker rmi ubuntu:18.04, if I wouldn't have given 16.04 version here then by default docker takes the latest tag id and ubuntu with latest image should have deleted.

```
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
centostomcatimage   1         159638f65ba1   2 hours ago     479MB
mytommyimage        1.0       308255179973   2 hours ago     703MB
ubuntu              latest    9873176a8ff5   5 days ago      72.7MB
ubuntu              18.04     7d0d8fa37224   5 days ago      63.1MB
centos              latest    300e315adb2f   6 months ago    209MB
[ec2-user@ip-172-31-42-105 ~]$
```

```
[ec2-user@ip-172-31-42-105 ~]$ docker rmi ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
centostomcatimage   1         159638f65ba1   2 hours ago     479MB
mytommyimage        1.0       308255179973   2 hours ago     703MB
ubuntu              18.04     7d0d8fa37224   5 days ago      63.1MB
centos              latest    300e315adb2f   6 months ago    209MB
[ec2-user@ip-172-31-42-105 ~]$ docker rmi ubuntu
Error: No such image: ubuntu
[ec2-user@ip-172-31-42-105 ~]$ docker rmi ubuntu:18.04
Untagged: ubuntu:18.04
Untagged: ubuntu@sha256:139b3846cee2e63de9ced83cee7023a2d95763ee2573e5b0ab6dea9dfbd4db8f
Deleted: sha256:7d0d8fa372249c6a1de9868ad62af9a14aaae2a2b17da867d8fad099a637fd0f
Deleted: sha256:8f8f0266f8349dd17fdca5a1c827a7c8b11d84903b1226b05c6fbdc7edd30652
[ec2-user@ip-172-31-42-105 ~]$
```

First we need to go to docker hub and create a public repository with any name





Login to the docker hub using below command

```
[ec2-user@ip-172-31-42-105 ~]$ docker logout
Removing login credentials for https://index.docker.io/v1/
[ec2-user@ip-172-31-42-105 ~]$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.doc
ker.com to create one.
Username: ppreddy
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-42-105 ~]$
```

I can’t push any image directly to docker hub without tagging with my repository name because default repository is always docker.io and it will give us error when we will try to push any image directly.

```
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
centostomcatimage   1         159638f65ba1   2 hours ago     479MB
mytommyimage        1.0       308255179973   2 hours ago     703MB
centos              latest    300e315adb2f   6 months ago    209MB
[ec2-user@ip-172-31-42-105 ~]$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-42-105 ~]$ docker push mytommyimage:1.0
The push refers to repository [docker.io/library/mytommyimage]
8094ca45df82: Preparing
882bf34d0696: Preparing
876cf3995077: Preparing
bc26662dee57: Preparing
4d1dedd2c965: Preparing
5d245775f9d1: Waiting
feef05f055c9: Waiting
denied: requested access to the resource is denied
[ec2-user@ip-172-31-42-105 ~]$
```

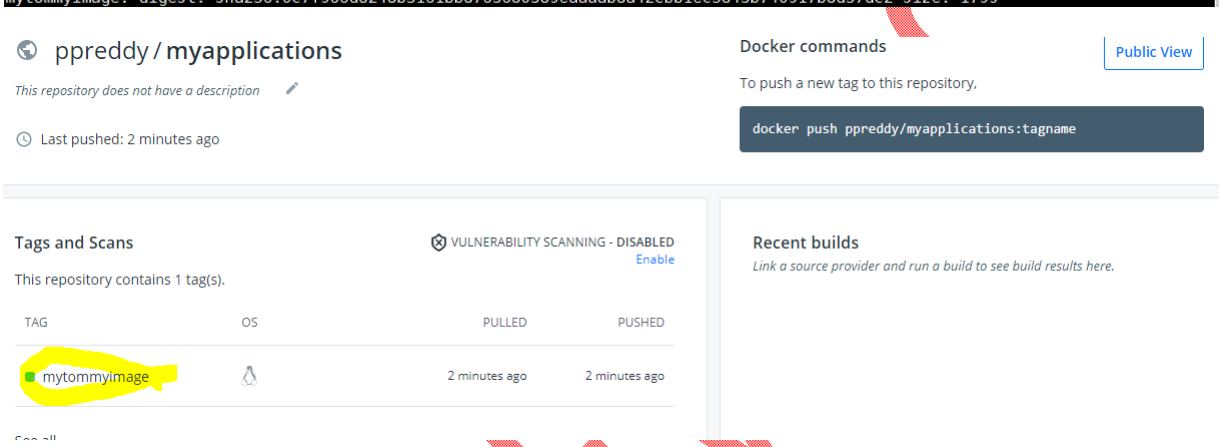
So, in below snap we have pushed the image to our docker hub account.

docker tag mytommyimage:1.0 ppreddy/myapplications:mytommyimage

docker push ppreddy/myapplications:mytommyimage

……. Prepared by PPNREDDY

```
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
centostomcatimage   1         159638f65ba1   2 hours ago    479MB
mytommyimage        1.0       308255179973   2 hours ago    703MB
centos              latest    300e315adb2f   6 months ago   209MB
[ec2-user@ip-172-31-42-105 ~]$ docker tag mytommyimage:1.0 ppreddy/myapplications:mytommyimage
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY                TAG             IMAGE ID       CREATED        SIZE
centostomcatimage         1               159638f65ba1   2 hours ago    479MB
mytommyimage              1.0             308255179973   2 hours ago    703MB
ppreddy/myapplications    mytommyimage    308255179973   2 hours ago    703MB
centos                    latest          300e315adb2f   6 months ago   209MB
[ec2-user@ip-172-31-42-105 ~]$ docker push ppreddy/myapplications:mytommyimage
The push refers to repository [docker.io/ppreddy/myapplications]
8094ca45df82: Pushed
882bf34d0696: Pushed
876cf3995077: Pushed
bc26662dee57: Pushed
4d1dedd2c965: Pushed
5d245775f9d1: Pushed
feef05f055c9: Pushed
mytommyimage: digest: sha256:0c7f966d8248b3161bb8763680389eaaaab8a42cbb1ec5843b740917b8d57dc2 size: 1799
```



In below snap I had pushed the image to docker hub, and then removed that image from my docker engine, after that again I pulled the same image from my docker hub to my docker engine

```
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY                TAG             IMAGE ID       CREATED        SIZE
centostomcatimage         1               159638f65ba1   2 hours ago    479MB
mytommyimage              1.0             308255179973   2 hours ago    703MB
ppreddy/myapplications    mytommyimage    308255179973   2 hours ago    703MB
centos                    latest          300e315adb2f   6 months ago   209MB
[ec2-user@ip-172-31-42-105 ~]$ docker rmi ppreddy/myapplications:mytommyimage
Untagged: ppreddy/myapplications:mytommyimage
Untagged: ppreddy/myapplications@sha256:0c7f966d8248b3161bb8763680389eaaaab8a42cbb1ec5843b740917b8d57dc2
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
centostomcatimage   1         159638f65ba1   2 hours ago    479MB
mytommyimage        1.0       308255179973   3 hours ago    703MB
centos              latest    300e315adb2f   6 months ago   209MB
[ec2-user@ip-172-31-42-105 ~]$ docker pull  ppreddy/myapplications:mytommyimag
Error response from daemon: manifest for ppreddy/myapplications:mytommyimag not found: manifest unknown: manifest unknown
[ec2-user@ip-172-31-42-105 ~]$ docker pull  ppreddy/myapplications:mytommyimage
mytommyimage: Pulling from ppreddy/myapplications
Digest: sha256:0c7f966d8248b3161bb8763680389eaaaab8a42cbb1ec5843b740917b8d57dc2
Status: Downloaded newer image for ppreddy/myapplications:mytommyimage
docker.io/ppreddy/myapplications:mytommyimage
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY                TAG             IMAGE ID       CREATED        SIZE
centostomcatimage         1               159638f65ba1   2 hours ago    479MB
ppreddy/myapplications    mytommyimage    308255179973   3 hours ago    703MB
mytommyimage              1.0             308255179973   3 hours ago    703MB
centos                    latest          300e315adb2f   6 months ago   209MB
[ec2-user@ip-172-31-42-105 ~]$
```

DTR (Docker Trusted Registry)

DTR is a Private repository just like docker-hub but it lives in your Data Center, so that images in DTR can be accessed by only your company.

We need to setup the DTR in your local to push pull the images, once we setup DTR we will have similar UI to login and create your own repo as we did in Docker Hub

We can use the same commands which we used for docker hub, the difference is you need to add your repo addr to login & push.

Ex:

docker login <repo addr>

docker tag <image id/name> repoaddr/loginid/reponame:tag

docker push repoaddr/loginid/reponame:tag

(If we have noticed by default always docker repository is docker.io, so in case of our private DTR we need to specify our repository as well)

We can also save any images in docker in .tgz format using save command and can run the tgz file back to images using load command as shown in below picture.

docker pull ubuntu:latest

docker images

```
[ec2-user@ip-172-31-42-105 ~]$ docker pull ubuntu:latest
latest: Pulling from library/ubuntu
Digest: sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY                TAG             IMAGE ID       CREATED         SIZE
centostomcatimage         1               159638f65ba1   2 hours ago     479MB
ppreddy/myapplications    centos          159638f65ba1   2 hours ago     479MB
ppreddy/myapplications    mytommyimage    308255179973   3 hours ago     703MB
ppreddy/mytommy           1               308255179973   3 hours ago     703MB
mytommyimage              1.0             308255179973   3 hours ago     703MB
ubuntu                    latest          9873176a8ff5   5 days ago      72.7MB
centos                    latest          300e315adb2f   6 months ago    209MB
[ec2-user@ip-172-31-42-105 ~]$
```

docker save -o ubuntu.tgz ubuntu

docker rmi ubuntu:latest

```
[ec2-user@ip-172-31-42-105 ~]$ docker save -o ubuntu.tgz ubuntu
[ec2-user@ip-172-31-42-105 ~]$ ls
Dockerfile  Dockerfile_1  dockerfiledemos  mydata.json  mydata.json_21062021  mypage.html  psddevops_volume  ubuntu.tgz
[ec2-user@ip-172-31-42-105 ~]$ docker rmi ubuntu:latest
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY                TAG             IMAGE ID       CREATED         SIZE
ppreddy/myapplications    centos          159638f65ba1   2 hours ago     479MB
centostomcatimage         1               159638f65ba1   2 hours ago     479MB
ppreddy/mytommy           1               308255179973   3 hours ago     703MB
mytommyimage              1.0             308255179973   3 hours ago     703MB
ppreddy/myapplications    mytommyimage    308255179973   3 hours ago     703MB
centos                    latest          300e315adb2f   6 months ago    209MB
[ec2-user@ip-172-31-42-105 ~]$
```

docker load < ubuntu.tgz

docker images

```
[ec2-user@ip-172-31-42-105 ~]$ docker load < ubuntu.tgz
Loaded image: ubuntu:latest
[ec2-user@ip-172-31-42-105 ~]$ docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
centostomcatimage         1                   159638f65ba1        2 hours ago         479MB
ppreddy/myapplications    centos              159638f65ba1        2 hours ago         479MB
mytommyimage              1.0                 308255179973        3 hours ago         703MB
ppreddy/myapplications    mytommyimage        308255179973        3 hours ago         703MB
ppreddy/mytommy           1                   308255179973        3 hours ago         703MB
ubuntu                    latest              9873176a8ff5        5 days ago          72.7MB
centos                    latest              300e315adb2f        6 months ago        209MB
[ec2-user@ip-172-31-42-105 ~]$
```

Docker Compose

Till now we had seen how to create single container out of single image, but what

- If we want to run multiple containers from a single image
- If we want to run multiple containers from multiple images

Here docker compose comes into the scenario, we can do the above-mentioned activities with docker compose.

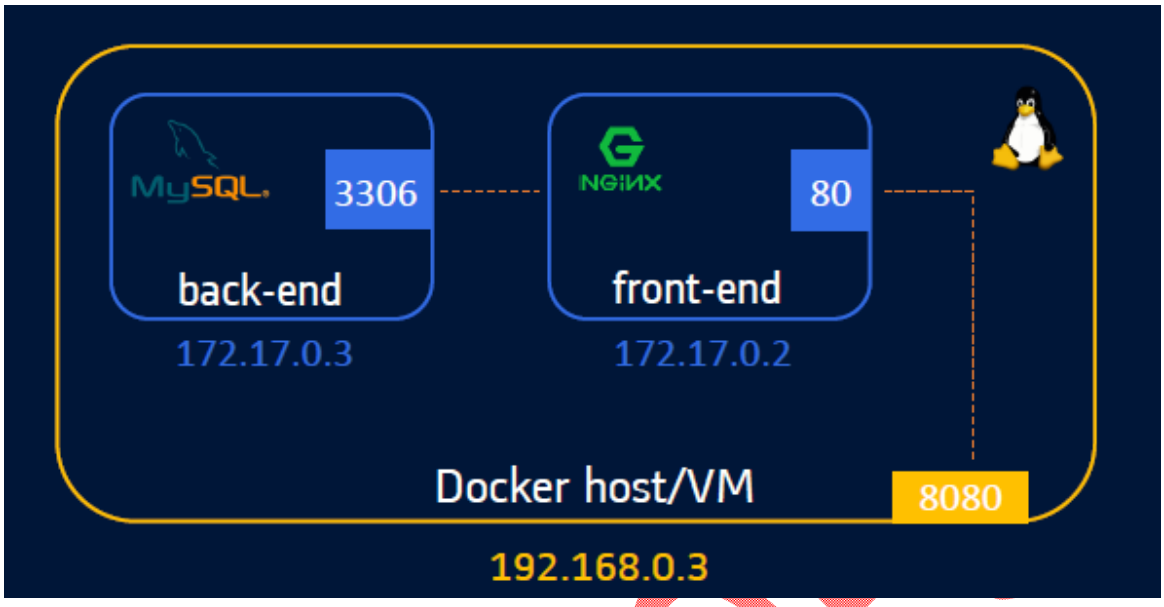
Docker compose is a third-party utility, with docker engine we won't get docker compose by default installed in our host machine, we need to install docker compose.

In docker compose we defines everything in declarative mode.Declarative mode means define everything in docker-compose.yml or docker-compose.yaml syntax

Docker compose follows yaml syntaxing, YAML is "YAML Ain't Markup Language", it uses indentation methodology similar to python, those who worked on Ansible or Kubernetes would be aware of YAML syntax.

Note: Dockerfile is used to create the images but docker compose is used to create the containers, many people's get confuses over here so please note down this point.

- Docker Compose is used to run multiple containers as a single service.
- For example, an application requires both NGNIX and MySQL containers, you could create one file which would start both the containers as a service (docker compose) or start each one separately (docker run).



Docker Compose versions

Version 1

- Compose files that do not declare a version are considered "version 1"
- Do not support named volumes, user-defined networks or build arguments
- Every container is placed on the default bridge network and is reachable from every other container at its IP address. You need to use links to enable discovery between containers
- No DNS resolution using container names

Version 2

- Links are deprecated. DNS resolution through container names
- All services must be declared under the 'services' key
- Named volumes can be declared under the volumes key, and networks can be declared under the networks key
- New bridge network to connect all containers

Version 3

- Support for docker swarm

Docker-compose installation

1. Run below command to download the current stable release of Docker Compose.

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

```
[ec2-user@ip-172-31-42-105 ~]$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname
 -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   633  100   633    0     0   8791      0 --:--:-- --:--:-- --:--:--  8791
100 12.1M  100 12.1M    0     0  17.5M      0 --:--:-- --:--:-- --:--:-- 31.9M
[ec2-user@ip-172-31-42-105 ~]$
```

2. Apply executable permissions to the binary

```
[ec2-user@ip-172-31-42-105 ~]$
[ec2-user@ip-172-31-42-105 ~]$ sudo chmod +x /usr/local/bin/docker-compose
[ec2-user@ip-172-31-42-105 ~]$
```

3. Verify the compose version

   docker-compose --version

```
[ec2-user@ip-172-31-42-105 ~]$ docker-compose --version
docker-compose version 1.29.2, build 5becea4c
[ec2-user@ip-172-31-42-105 ~]$
```

Syntax

version: '3' # if no version is specified then v1 assumed.

services : # containers. same as docker run

service_name : # container name. this is also DNS name inside network

image: # name of the image

command : # Optional, replace the default CMD specified by image

environment: # same as -e in docker run

ports : # same as –p in docker run

volumes : # same as -v in docker run

service_name2:

volumes : # Optional, same as docker volume create

networks : # Optional, same as docker network create

Compose file name: docker-compose.yml (OR) docker-compose.yaml

Up the services: docker-compose up -d

Down the services: docker-compose down

Demo-1

Place the below content in compose file

```
version: '3'
services:
  webserver:
    image: nginx
    ports:
      - "80"
    restart: always
```

```
    network_mode: bridge
    volumes:
      - /home/ec2-user/psdvolume:/usr/share/nginx/html
  appserver:
    image: tomcat
    ports:
      - "8080"
    network_mode: bridge
    restart: always
```

docker images

docker ps

docker ps –a

```
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
centostomcatimage         1                   159638f65ba1        16 hours ago        479MB
opreddy/myapplications    centos              159638f65ba1        16 hours ago        479MB
mytommyimage              1.0                 308255179973        16 hours ago        703MB
opreddy/myapplications    mytommyimage        308255179973        16 hours ago        703MB
opreddy/mytommy           1                   308255179973        16 hours ago        703MB
ubuntu                    latest              9873176a8ff5        6 days ago          72.7MB
centos                    latest              300e315adb2f        6 months ago        209MB
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

ls –lrt

docker-compose -f docker-compose.yaml up --scale webserver=2 --scale appserver=3 –d

```
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker-compose -f docker-compose.yaml up --scale webserver=2 --scale appserver=3 -d
Pulling webserver (nginx:)...
latest: Pulling from library/nginx
b4d181a07f80: Pull complete
edb81c9bc1f5: Pull complete
b21fed559b9f: Pull complete
03e6a2452751: Pull complete
b82f7f888feb: Pull complete
5430e98eba64: Pull complete
Digest: sha256:47ae43cdfc7064d28800bc42e79a429540c7c80168e8c8952778c0d5af1c09db
Status: Downloaded newer image for nginx:latest
Pulling appserver (tomcat:)...
latest: Pulling from library/tomcat
d960726af2be: Pull complete
e8d62473a22d: Pull complete
8962bc0fad55: Pull complete
65d943ee54c1: Pull complete
da20b77f10ac: Pull complete
8669a096f083: Pull complete
e0c0a5e9ce88: Pull complete
f7f46169d747: Pull complete
215575e3a745: Pull complete
6b282851d654: Pull complete
Digest: sha256:618fd3c32b26e89a74edb26d1438a262d0320370051d4430a733d536b0f3d497
Status: Downloaded newer image for tomcat:latest
Creating docker_compose_demo_webserver_1 ... done
Creating docker_compose_demo_webserver_2 ... done
Creating docker_compose_demo_appserver_1 ... done
Creating docker_compose_demo_appserver_2 ... done
Creating docker_compose_demo_appserver_3 ... done
[ec2-user@ip-172-31-42-105 docker_compose_demo]$
```

docker images

docker ps

docker ps –a

```
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker images
REPOSITORY               TAG            IMAGE ID       CREATED        SIZE
centostomcatimage        1              159638f65ba1   16 hours ago   479MB
ppreddy/myapplications   centos         159638f65ba1   16 hours ago   479MB
ppreddy/myapplications   mytommyimage   308255179973   17 hours ago   703MB
ppreddy/mytommy          1              308255179973   17 hours ago   703MB
mytommyimage             1.0            308255179973   17 hours ago   703MB
nginx                    latest         4f380adfc10f   23 hours ago   133MB
ubuntu                   latest         9873176a8ff5   6 days ago     72.7MB
tomcat                   latest         5505f7218e4d   7 days ago     667MB
centos                   latest         300e315adb2f   6 months ago   209MB
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                     NAMES
d1f820b2983d   tomcat    "catalina.sh run"        5 minutes ago   Up 5 minutes   0.0.0.0:49154->8080/tcp   docker_compose_demo_appserver_3
eff4a68e8cd7   tomcat    "catalina.sh run"        5 minutes ago   Up 5 minutes   0.0.0.0:49153->8080/tcp   docker_compose_demo_appserver_1
e9f1b60a164d   tomcat    "catalina.sh run"        5 minutes ago   Up 5 minutes   0.0.0.0:49155->8080/tcp   docker_compose_demo_appserver_2
f7673d8a87f0   nginx     "/docker-entrypoint.…"   5 minutes ago   Up 5 minutes   0.0.0.0:49157->80/tcp     docker_compose_demo_webserver_1
9b915a97371d   nginx     "/docker-entrypoint.…"   5 minutes ago   Up 5 minutes   0.0.0.0:49156->80/tcp     docker_compose_demo_webserver_2
[ec2-user@ip-172-31-42-105 docker_compose_demo]$
```

Checking volumes

```
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ cd
[ec2-user@ip-172-31-42-105 ~]$ ll
total 73436
drwxrwxr-x 2 ec2-user ec2-user       33 Jun 24 05:38 docker_compose_demo
-rw-rw-r-- 1 ec2-user ec2-user      364 Jun 21 14:11 Dockerfile
-rw-rw-r-- 1 ec2-user ec2-user      359 Jun 21 14:08 Dockerfile_1
drwxrwxr-x 2 ec2-user ec2-user      196 Jun 23 13:29 dockerfiledemos
-rw-rw-r-- 1 ec2-user ec2-user    21674 Jun 21 02:46 mydata.json
-rw-r--r-- 1 ec2-user ec2-user    21674 Jun 21 02:50 mydata.json_21062021
-rw-rw-r-- 1 ec2-user ec2-user      256 Jun 19 02:11 mypage.html
drwxrwxr-x 2 ec2-user ec2-user       43 Jun 21 06:55 psddevops_volume
drwxrwxr-x 2 ec2-user ec2-user        6 Jun 24 05:37 psdvolume
-rw------- 1 ec2-user ec2-user 75135488 Jun 23 15:57 ubuntu.tgz
[ec2-user@ip-172-31-42-105 ~]$ cd psdvolume
[ec2-user@ip-172-31-42-105 psdvolume]$ ll
total 0
[ec2-user@ip-172-31-42-105 psdvolume]$ docker exec -it f7673d8a87f0 bash
root@f7673d8a87f0:/# cd /usr/share/nginx/html
root@f7673d8a87f0:/usr/share/nginx/html# ll
bash: ll: command not found
root@f7673d8a87f0:/usr/share/nginx/html# ls
root@f7673d8a87f0:/usr/share/nginx/html# touch myfile.txt
root@f7673d8a87f0:/usr/share/nginx/html# read escape sequence
[ec2-user@ip-172-31-42-105 psdvolume]$ ls
myfile.txt
[ec2-user@ip-172-31-42-105 psdvolume]$
```

docker-compose -f docker-compose.yaml down

```
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS         PORTS                     NAMES
cac2fb616840   tomcat    "catalina.sh run"        11 seconds ago   Up 8 seconds   0.0.0.0:49162->8080/tcp   docker_compose_demo_appserver_2
c11742e110ec   tomcat    "catalina.sh run"        11 seconds ago   Up 8 seconds   0.0.0.0:49161->8080/tcp   docker_compose_demo_appserver_1
3558a56cb385   nginx     "/docker-entrypoint.…"   11 seconds ago   Up 9 seconds   0.0.0.0:49159->80/tcp     docker_compose_demo_webserver_1
4a7cb2354daa   nginx     "/docker-entrypoint.…"   11 seconds ago   Up 8 seconds   0.0.0.0:49160->80/tcp     docker_compose_demo_webserver_2
462a343a6753   nginx     "/docker-entrypoint.…"   11 seconds ago   Up 9 seconds   0.0.0.0:49158->80/tcp     docker_compose_demo_webserver_3
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker-compose -f docker-compose.yaml down
Stopping docker_compose_demo_appserver_2 ... done
Stopping docker_compose_demo_appserver_1 ... done
Stopping docker_compose_demo_webserver_1 ... done
Stopping docker_compose_demo_webserver_2 ... done
Stopping docker_compose_demo_webserver_3 ... done
Removing docker_compose_demo_appserver_2 ... done
Removing docker_compose_demo_appserver_1 ... done
Removing docker_compose_demo_webserver_1 ... done
Removing docker_compose_demo_webserver_2 ... done
Removing docker_compose_demo_webserver_3 ... done
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker images
REPOSITORY               TAG            IMAGE ID       CREATED        SIZE
centostomcatimage        1              159638f65ba1   16 hours ago   479MB
ppreddy/myapplications   centos         159638f65ba1   16 hours ago   479MB
ppreddy/mytommy          1              308255179973   17 hours ago   703MB
mytommyimage             1.0            308255179973   17 hours ago   703MB
ppreddy/myapplications   mytommyimage   308255179973   17 hours ago   703MB
nginx                    latest         4f380adfc10f   23 hours ago   133MB
ubuntu                   latest         9873176a8ff5   6 days ago     72.7MB
tomcat                   latest         5505f7218e4d   7 days ago     667MB
centos                   latest         300e315adb2f   6 months ago   209MB
[ec2-user@ip-172-31-42-105 docker_compose_demo]$
```

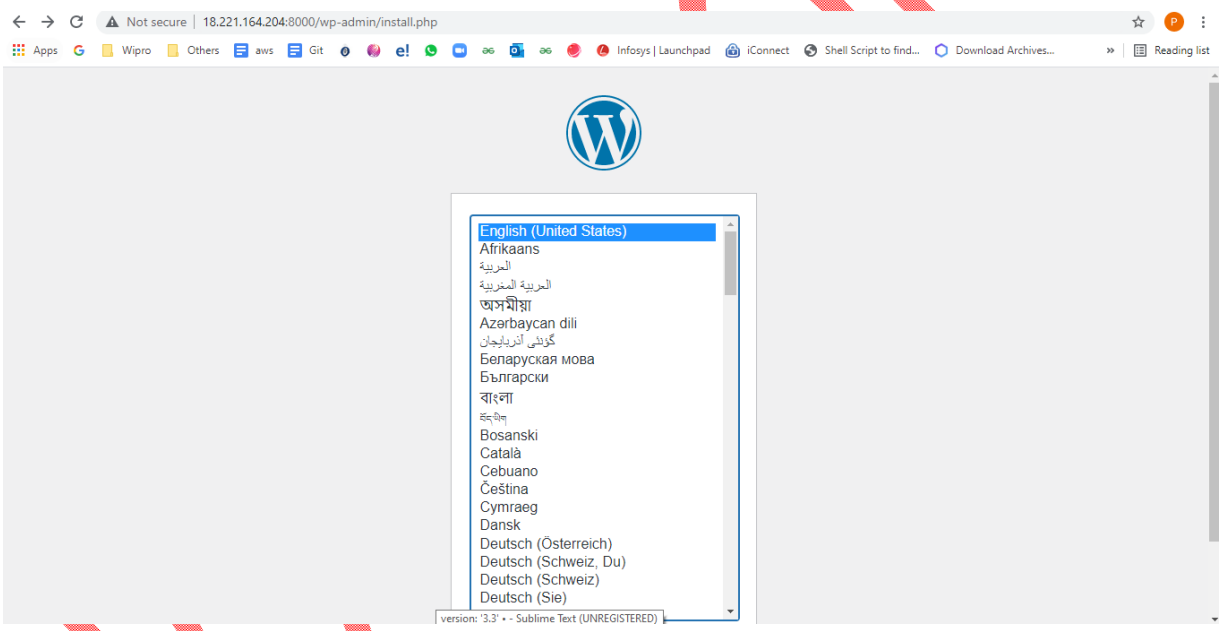
Demo-2

Copy the below content in docker-compose

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}
```

docker-compose up –d

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-42-105 ~]$ cd docker_compose_demo/
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ vi docker-compose.yaml
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker-compose up -d
Creating network "docker_compose_demo_default" with the default driver
Creating volume "docker_compose_demo_db_data" with default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
b4d181a07f80: Already exists
a462b60610f5: Pull complete
578fafb77ab8: Pull complete
524046006037: Pull complete
d0cbe54c8855: Pull complete
aa18e05cc46d: Pull complete
32ca814c833f: Pull complete
52645b4af634: Pull complete
bca6a5b14385: Pull complete
309f36297c75: Pull complete
7d75cacde0f8: Pull complete
Digest: sha256:1a2f9cd257e75cc80e9118b303d1648366bc2049101449bf2c8d82b022ea86b7
Status: Downloaded newer image for mysql:5.7
Creating docker_compose_demo_db_1 ... done
Creating docker_compose_demo_wordpress_1 ... done
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                    NAMES
aafc95b92f5d   wordpress:latest   "docker-entrypoint.s…"   12 seconds ago   Up 9 seconds    0.0.0.0:8000->80/tcp     docker_compose_
demo_wordpress_1
609287f9af4a   mysql:5.7          "docker-entrypoint.s…"   12 seconds ago   Up 11 seconds   3306/tcp, 33060/tcp      docker_compose_
demo_db_1
[ec2-user@ip-172-31-42-105 docker_compose_demo]$
```



docker-compose down

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-42-105 ~]$ cd docker_compose_demo/
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker-compose down
Stopping docker_compose_demo_wordpress_1 ... done
Stopping docker_compose_demo_db_1        ... done
Removing docker_compose_demo_wordpress_1 ... done
Removing docker_compose_demo_db_1        ... done
Removing network docker_compose_demo_default
[ec2-user@ip-172-31-42-105 docker_compose_demo]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-42-105 docker_compose_demo]$
```

Docker networking

Docker networking enables a user to link a Docker container to as many networks as he/she requires. Docker Networks are used to provide complete isolation for Docker containers.

Note: A user can add containers to more than one network.

Advantages

- They share a single operating system and maintain containers in an isolated environment.
- It requires fewer OS instances to run the workload.
- It helps in the fast delivery of software.
- It helps in application portability.

Network Drivers

Docker supports networking for its containers via network drivers. These drivers have several network drivers.

- Bridge
- Host
- None
- Overlay
- Macvlan

Bridge

- It is a private default network created on the host.
- Containers linked to this network have an internal IP address through which they communicate with each other easily.
- The Docker server (daemon) creates a virtual ethernet bridge docker0 that operates automatically, by delivering packets among various network interfaces.
- These are widely used when applications are executed in a standalone container.

Host

- It is a public network.
- It utilizes the host's IP address and TCP port space to display the services running inside the container.

- It effectively disables network isolation between the docker host and the docker containers, which means using this network driver a user will be unable to run multiple containers on the same host.

None

- In this network driver, the Docker containers will neither have any access to external networks nor will it be able to communicate with other containers.
- This option is used when a user wants to disable the networking access to a container.
- In simple terms, None is called a loopback interface, which means it has no external network interfaces

Overlay

- This is utilized for creating an internal private network to the Docker nodes in the Docker swarm cluster.
- Docker Swarm is a service for containers which facilitates developer teams to build and manage a cluster of swarm nodes within the Docker platform
- It is an important network driver in Docker networking. It helps in providing the interaction between the stand-alone container and the Docker swarm service.

Macvlan

- It simplifies the communication process between containers.
- This network assigns a MAC address to the Docker container. With this Mac address, the Docker server (daemon) routes the network traffic to a router.
- Docker Daemon is a server which interacts with the operating system and performs all kind of services.
- It is suitable when a user wants to directly connect the container to the physical network rather than the Docker host.

List out the networks

docker network ls

```
[ec2-user@ip-172-31-42-105 ~]$ docker network ls
NETWORK ID     NAME                          DRIVER    SCOPE
752d9c7459cc   bridge                        bridge    local
28b40cf987ae   docker_compose_demo_default   bridge    local
b554b45ea40e   host                          host      local
17e24e13317b   none                          null      local
[ec2-user@ip-172-31-42-105 ~]$
```

Remove all Unused Networks

docker network prune

```
[ec2-user@ip-172-31-42-105 ~]$ docker network prune
WARNING! This will remove all custom networks not used by at least one container.
Are you sure you want to continue? [y/N] y
[ec2-user@ip-172-31-42-105 ~]$ docker network ls
NETWORK ID     NAME                           DRIVER    SCOPE
752d9c7459cc   bridge                         bridge    local
28b40cf987ae   docker_compose_demo_default    bridge    local
b554b45ea40e   host                           host      local
17e24e13317b   none                           null      local
[ec2-user@ip-172-31-42-105 ~]$
```

Bridge-Demo

Creating the bridge network

docker network –help

```
[ec2-user@ip-172-31-42-105 ~]$ docker network --help

Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.
[ec2-user@ip-172-31-42-105 ~]$
```

docker network create --driver bridge my-bridge-network

docker network ls

```
[ec2-user@ip-172-31-42-105 ~]$ docker network create --driver bridge my-bridge-network
175cc43a8d67f4978f868f37818362d4a38840c6f3ecd39d0bf91eb09b7f1890
[ec2-user@ip-172-31-42-105 ~]$ docker network ls
NETWORK ID     NAME                           DRIVER    SCOPE
752d9c7459cc   bridge                         bridge    local
28b40cf987ae   docker_compose_demo_default    bridge    local
b554b45ea40e   host                           host      local
175cc43a8d67   my-bridge-network              bridge    local
17e24e13317b   none                           null      local
[ec2-user@ip-172-31-42-105 ~]$
```

Create two ubuntu containers and attach to the custom network

docker run -d -it --name c1 --net my-bridge-network ubuntu:latest

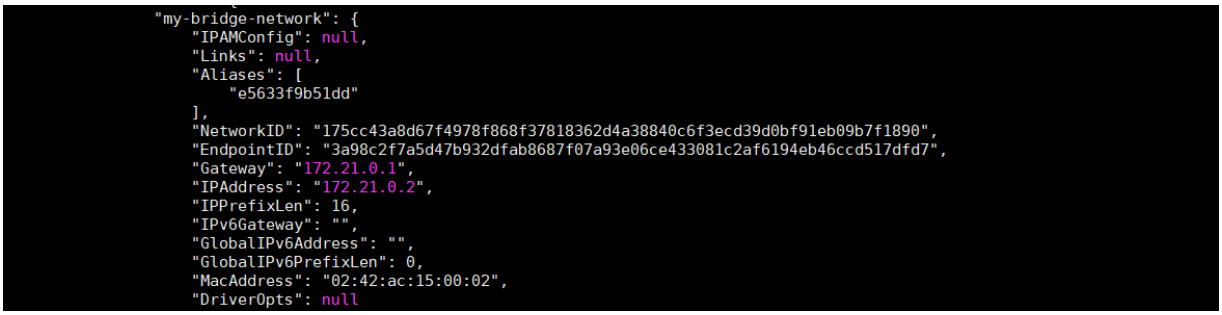
docker run -d -it --name c2 --net my-bridge-network ubuntu:latest

docker ps

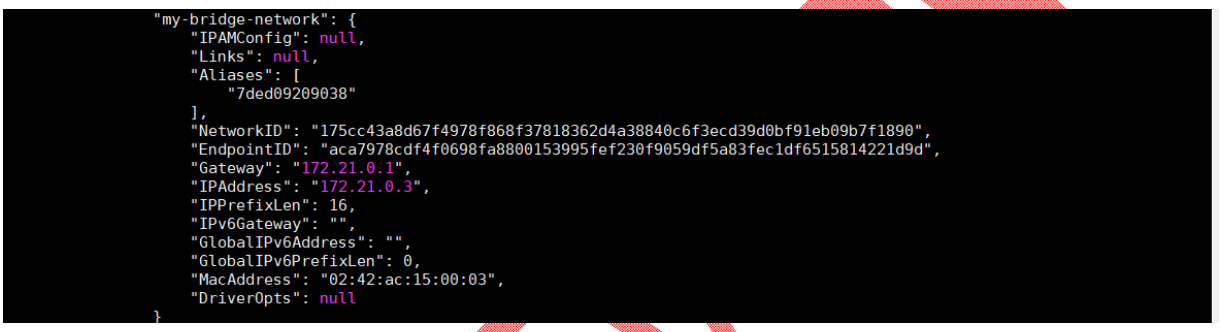
```
[ec2-user@ip-172-31-42-105 ~]$ docker run -d -it --name c1 --net my-bridge-network ubuntu:latest
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
c549ccf8d472: Pull complete
Digest: sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
Status: Downloaded newer image for ubuntu:latest
e5633f9b51dd149fbde07d9222be6e6c0a9c50db3a52693e3f5b0690bb9b2f50
[ec2-user@ip-172-31-42-105 ~]$ docker run -d -it --name c2 --net my-bridge-network ubuntu:latest
7ded09209038d1fbac1b8eee84c1a2b7636ad0e562624c4124d842a106782268
```

```
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS              PORTS
          NAMES
7ded09209038   ubuntu:latest          "bash"                   36 seconds ago       Up 35 seconds
          c2
e5633f9b51dd   ubuntu:latest          "bash"                   50 seconds ago       Up 48 seconds
          c1
```

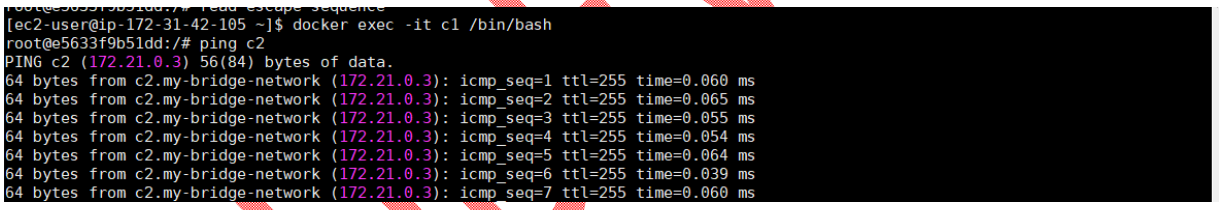
docker inspect c1



docker inspect c2



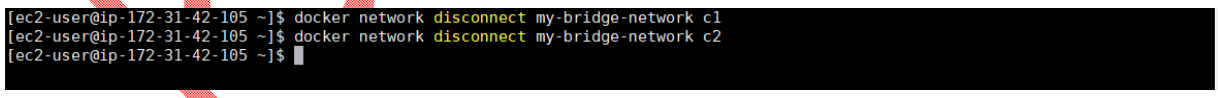
Pinging one container another container



Disconnect container from network

docker network disconnect my-bridge-network c1

docker network disconnect my-bridge-network c2



Remove a Network

docker network rm my-bridge-network

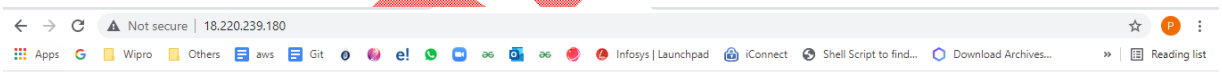
```
[ec2-user@ip-172-31-42-105 ~]$ docker network ls
NETWORK ID     NAME                          DRIVER    SCOPE
752d9c7459cc   bridge                        bridge    local
28b40cf987ae   docker_compose_demo_default   bridge    local
b554b45ea40e   host                          host      local
175cc43a8d67   my-bridge-network             bridge    local
17e24e13317b   none                          null      local
[ec2-user@ip-172-31-42-105 ~]$ docker network rm my-bridge-network
my-bridge-network
[ec2-user@ip-172-31-42-105 ~]$ docker network ls
NETWORK ID     NAME                          DRIVER    SCOPE
752d9c7459cc   bridge                        bridge    local
28b40cf987ae   docker_compose_demo_default   bridge    local
b554b45ea40e   host                          host      local
17e24e13317b   none                          null      local
[ec2-user@ip-172-31-42-105 ~]$
```

Host-Demo

Create the container with host network

docker run -d -it --name c1 --net=host nginx:latest

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -d -it --name c1 --net=host nginx:latest
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
b4d181a07f80: Pull complete
edb81c9bc1f5: Pull complete
b21fed559b9f: Pull complete
03e6a2452751: Pull complete
b82f7f888feb: Pull complete
5430e98eba64: Pull complete
Digest: sha256:47ae43cdfc7064d28800bc42e79a429540c7c80168e8c8952778c0d5af1c09db
Status: Downloaded newer image for nginx:latest
5697e9104becf82d20950084e646bfd0c37c85747454dbb90cda83d8fc2818eb
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
5697e9104bec   nginx:latest   "/docker-entrypoint.…"   48 seconds ago   Up 47 seconds             c1
[ec2-user@ip-172-31-42-105 ~]$
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

docker exec -it 5697e9104bec /bin/bash

```
[ec2-user@ip-172-31-42-105 ~]$ docker exec -it 5697e9104bec /bin/bash
root@ip-172-31-42-105:/# ifconfig
bash: ifconfig: command not found
root@ip-172-31-42-105:/# apt update
Get:1 http://security.debian.org/debian-security buster/updates InRelease [65.4 kB]
Get:2 http://deb.debian.org/debian buster InRelease [122 kB]
Get:3 http://deb.debian.org/debian buster-updates InRelease [51.9 kB]
Get:4 http://security.debian.org/debian-security buster/updates/main amd64 Packages [293 kB]
Get:5 http://deb.debian.org/debian buster/main amd64 Packages [7907 kB]
Get:6 http://deb.debian.org/debian buster-updates/main amd64 Packages [15.2 kB]
Fetched 8453 kB in 2s (4310 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
root@ip-172-31-42-105:/#
```

root@ip-172-31-42-105:/# apt install net-tools

```
root@ip-172-31-42-105:/# apt install net-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  net-tools
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 248 kB of archives.
After this operation, 1002 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main amd64 net-tools amd64 1.60+git20180626.aebd88e-1 [248 kB]
Fetched 248 kB in 0s (4142 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package net-tools.
(Reading database ... 7638 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20180626.aebd88e-1_amd64.deb ...
Unpacking net-tools (1.60+git20180626.aebd88e-1) ...
Setting up net-tools (1.60+git20180626.aebd88e-1) ...
root@ip-172-31-42-105:/#
```

root@ip-172-31-42-105:/# ifconfig

```
root@ip-172-31-42-105:/# ifconfig
br-28b40cf987ae: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.20.0.1  netmask 255.255.0.0  broadcast 172.20.255.255
        ether 02:42:e1:0b:35:91  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:2e:99:35:86  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 172.31.42.105  netmask 255.255.240.0  broadcast 172.31.47.255
        inet6 fe80::8bf:a6ff:fea2:d9b0  prefixlen 64  scopeid 0x20<link>
        ether 0a:bf:a6:a2:d9:b0  txqueuelen 1000  (Ethernet)
        RX packets 85424  bytes 123432758 (117.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28439  bytes 2131044 (2.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@ip-172-31-42-105:/#
```

Now, you should exit the container and run the same command on the host

```
[ec2-user@ip-172-31-42-105 ~]$ ifconfig
br-28b40cf987ae: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.20.0.1  netmask 255.255.0.0  broadcast 172.20.255.255
        ether 02:42:e1:0b:35:91  txqueuelen 0  (Ethernet)
        RX packets 85518  bytes 123438058 (117.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28516  bytes 2137196 (2.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:2e:99:35:86  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 172.31.42.105  netmask 255.255.240.0  broadcast 172.31.47.255
        inet6 fe80::8bf:a6ff:fea2:d9b0  prefixlen 64  scopeid 0x20<link>
        ether 0a:bf:a6:a2:d9:b0  txqueuelen 1000  (Ethernet)
        RX packets 85518  bytes 123438058 (117.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28525  bytes 2138038 (2.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[ec2-user@ip-172-31-42-105 ~]$
```

It can be seen that the output of the command is exactly the same. This means you should be able to access the container using host ip.

Now let's try another thing. We will try to run another container of the same service using host mode networking on this host and see what happens

```
[ec2-user@ip-172-31-42-105 ~]$ clear
[ec2-user@ip-172-31-42-105 ~]$ docker run -d -it --name c2 --net=host nginx:latest
b246e6af2f2981a2692eef0a52eebbf644e92f6b1de99328b8eb5128b3564614
[ec2-user@ip-172-31-42-105 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
5697e9104bec   nginx:latest   "/docker-entrypoint.…"   12 minutes ago   Up 12 minutes             c1
[ec2-user@ip-172-31-42-105 ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                      PORTS     NAMES
b246e6af2f29   nginx:latest   "/docker-entrypoint.…"   25 seconds ago   Exited (1) 21 seconds ago             c2
5697e9104bec   nginx:latest   "/docker-entrypoint.…"   12 minutes ago   Up 12 minutes                         c1
```

If you notice carefully, the container did not even start. Let's look at the logs.

docker logs c2

```
[ec2-user@ip-172-31-42-105 ~]$ docker logs c2
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2021/06/25 05:06:48 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
2021/06/25 05:06:48 [emerg] 1#1: bind() to [::]:80 failed (98: Address already in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
2021/06/25 05:06:48 [notice] 1#1: try again to bind() after 500ms
2021/06/25 05:06:48 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
2021/06/25 05:06:48 [emerg] 1#1: bind() to [::]:80 failed (98: Address already in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
2021/06/25 05:06:48 [notice] 1#1: try again to bind() after 500ms
2021/06/25 05:06:48 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
2021/06/25 05:06:48 [emerg] 1#1: bind() to [::]:80 failed (98: Address already in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
2021/06/25 05:06:48 [notice] 1#1: try again to bind() after 500ms
2021/06/25 05:06:48 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
2021/06/25 05:06:48 [emerg] 1#1: bind() to [::]:80 failed (98: Address already in use)
nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
2021/06/25 05:06:48 [notice] 1#1: try again to bind() after 500ms
2021/06/25 05:06:48 [emerg] 1#1: bind() to 0.0.0.0:80 failed (98: Address already in use)
nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
```

None-Demo

A none network doesn't provide any networking capability to the container which means the container is like a black box to the host. The host or any other container won't be able to communicate with the container.

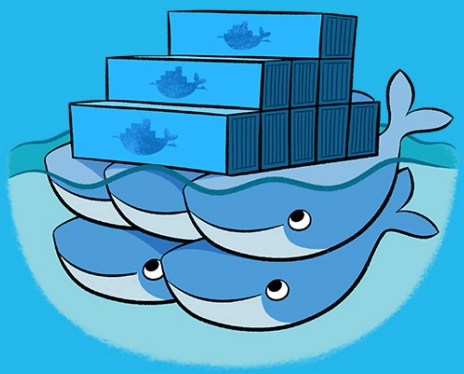
docker network ls

```
[ec2-user@ip-172-31-42-105 ~]$ docker network ls
NETWORK ID     NAME                          DRIVER    SCOPE
0e8ea5f150ef   bridge                        bridge    local
28b40cf987ae   docker_compose_demo_default   bridge    local
b554b45ea40e   host                          host      local
17e24e13317b   none                          null      local
[ec2-user@ip-172-31-42-105 ~]$
```

docker run -it --name c1 --net none ubuntu:latest

```
[ec2-user@ip-172-31-42-105 ~]$ docker run -it --name c1 --net none ubuntu:latest
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
c549ccf8d472: Pull complete
Digest: sha256:aba80b77e27148d99c034a987e7da3a287ed455390352663418c0f2ed40417fe
Status: Downloaded newer image for ubuntu:latest
root@3024ae899aaf:/# apt-get update
Err:1 http://security.ubuntu.com/ubuntu focal-security InRelease
  Temporary failure resolving 'security.ubuntu.com'
Err:2 http://archive.ubuntu.com/ubuntu focal InRelease
  Temporary failure resolving 'archive.ubuntu.com'
Err:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease
  Temporary failure resolving 'archive.ubuntu.com'
Err:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease
  Temporary failure resolving 'archive.ubuntu.com'
Reading package lists... Done
W: Failed to fetch http://archive.ubuntu.com/ubuntu/dists/focal/InRelease  Temporary failure resolving 'archive.ubuntu.com'
W: Failed to fetch http://archive.ubuntu.com/ubuntu/dists/focal-updates/InRelease  Temporary failure resolving 'archive.ubuntu.com
'
W: Failed to fetch http://archive.ubuntu.com/ubuntu/dists/focal-backports/InRelease  Temporary failure resolving 'archive.ubuntu.c
om'
W: Failed to fetch http://security.ubuntu.com/ubuntu/dists/focal-security/InRelease  Temporary failure resolving 'security.ubuntu.
com'
W: Some index files failed to download. They have been ignored, or old ones used instead.
root@3024ae899aaf:/#
```

There aren't many use cases of a black-box container but it could be used to isolate itself from the host and other containers. It is best for running non-networking tasks and while avoiding any communication with the outside work.



Docker Swarm

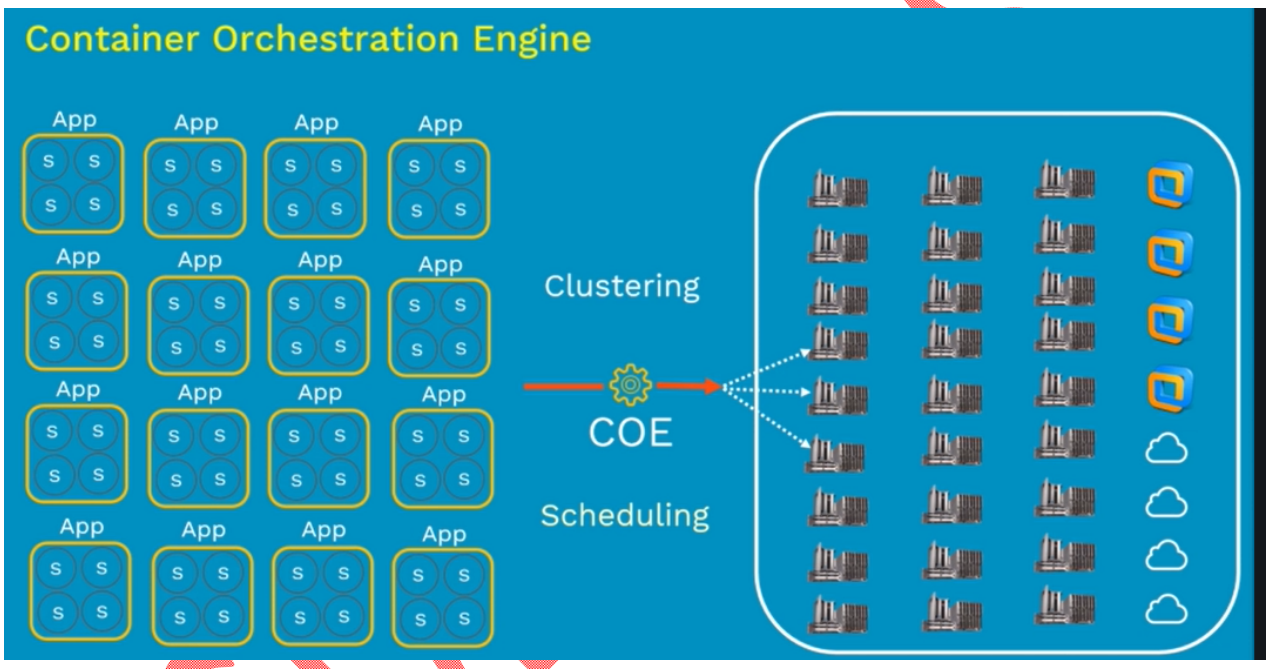
..... Prepared by PPNREDDY

Docker Swarm

Container Orchestration Engine (COE)

Container Orchestration Engine automates deploying, scaling, and managing containerized applications on a group of servers.

Ex: Docker Swarm
Kubernetes
Marathon from Apache Mesos …. etc.



COE Features

- Clustering (Master and Worker nodes)
- Scheduling (Deploy the applications specific worker nodes having more features)
- Scalability (Increase/Decrease the worker nodes OR containers)
- Load balancing
- Fault tolerance (If any container die/exit it will create new one)
- Deployment (Supports rolling and canary deployments)

Docker Swarm

Docker Swarm is a clustering and scheduling tool for **Docker** containers. With **Swarm**, IT administrators and developers can establish and manage a cluster of Docker nodes as a single virtual system.

- Docker Swarm Mode enables
  - To have multiple Docker Engines / Nodes as a single cluster
  - Provides orchestration features
- One manager, and rest workers (More than one manager can also be present, and recommended manager is three).
- Docker Swarm features are comes with Docker Engine – we do not need to install it separately like docker compose. We can just turn our Docker Engine into a Swarm mode.
- Written in Golang
- Easy to understand and get started
- Simple architecture than Kubernetes and Mesos
- Typical users of Docker Swarm are:
  - Smaller teams (startup and medium-size companies)
  - Typically greenfield projects (A software project that is developed from scratch)
  - Developer-driven teams who need to deploy products.

Advantages

Cluster management integrated with Docker Engine:

Use the Docker Engine CLI to create a swarm of Docker Engines where you can deploy application services. You don't need additional orchestration software to create or manage a swarm.

Decentralized design:

Instead of handling differentiation between node roles at deployment time, the Docker Engine handles any specialization at runtime. You can deploy both kinds of nodes, managers and workers, using the Docker Engine.

Scaling:

For each service, you can declare the number of tasks you want to run. When you scale up or down, the swarm manager automatically adapts by adding or removing tasks to maintain the desired state.

Desired state reconciliation:

The swarm manager node constantly monitors the cluster state and reconciles any differences between the actual state and your expressed desired state. For example, if you set up a service to run 10 replicas of a container, and a worker machine hosting two of those replicas' crashes, the manager creates two new replicas to replace the replicas that crashed. The swarm manager assigns the new replicas to workers that are running and available.

Multi-host networking:

You can specify an overlay network for your services. The swarm manager automatically assigns addresses to the containers on the overlay network when it initializes or updates the application.

Service discovery:

Swarm manager nodes assign each service in the swarm a unique DNS name and load balances running containers. You can query every container running in the swarm through a DNS server embedded in the swarm.

Load balancing:

You can expose the ports for services to an external load balancer. Internally, the swarm lets you specify how to distribute service containers between nodes.

Secure by default:

Each node in the swarm enforces TLS mutual authentication and encryption to secure communications between itself and all other nodes. You have the option to use self-signed root certificates or certificates from a custom root CA.

Rolling updates:

At rollout time you can apply service updates to nodes incrementally. The swarm manager lets you control the delay between service deployment to different sets of nodes. If anything goes wrong, you can roll-back a task to a previous version of the service.

How Docker Swarm works

- Docker swarm is a container orchestration tool which let us to create a cluster of docker nodes.
- In cluster we makes one node as manager and remaining nodes will be as worker, we can alsomake more than one manager as per our choice, manager node should manages the container tasks which is running in the worker nodes, but by default when we runs any container in swarm, the container gets distributed to master node as well, this is not the best practice, we should make the master node to manage the manager related task only to get the high performance.
- We can use below command to stop restrict services to run any container on manager nodes
  docker node update –availability drain
  This will drain all container from master node, and again if we want to make master node to run the container we need to use active
  docker node update –availability active
  We can use all options available by using the command: - docker node update --help

Workflow of Docker Swarm

- With docker swarm we can distribute our different services(container) in different hosts to achieve the high availability and load balancing, docker swarm comes automatically installed with docker engine, but by default the swarm service remains inactive, we need to initiate the swarm service in order to use swarm.

```
[ec2-user@ip-172-31-12-36 ~]$ docker info | grep swarm
[ec2-user@ip-172-31-12-36 ~]$ docker info | grep -i swarm
 Swarm: inactive
[ec2-user@ip-172-31-12-36 ~]$ docker swarm init
Swarm initialized: current node (6prs676umbk1l8zbst12g81yp) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-34do05uutovuvr7wduzbrpibl 172.31.12.3
6:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[ec2-user@ip-172-31-12-36 ~]$ docker info | grep -i swarm
 Swarm: active
[ec2-user@ip-172-31-12-36 ~]$
```

Now the time comes to make a server as manager/master server for swarm and remaining as slave/worker node, so we need to run command, "***docker swarm init***" on the master server so that swarm will activate in this server along with the token, using this token worker can get join the swarm cluster.

- Manager node is the master node where docker swarm is initiated, the manager node is responsible for maintaining the cluster state, managing the workers, adding and removing workers, creating distributing and ensuring the state of container and services across all workers.
- We can have a single manager node, but it is not recommended, because if it fails there will be no manager nodes to manage the cluster so for fault tolerance we can have multiple manager node in a single cluster. However when we talks of multiple manager nodes then there comes a question which manager nodes will act always and takes decision for any changes in worker node, so to prevent any conflict only one manager nodes is allowed to make decision and that node we will call it as leader and other manager node will be present for backup if in the case the leader node goes down other manager node will come into action, we can identify other manager node by running the command "***docker node ls***", the node with status of reachable are the secondary manager node. However leader node can't make any decision on its own, all decision needs to be mutually agreed upon by all the managers or majority of the managers in the cluster,This is important because if a leader was to make a decision and in between it fails before updating/informing the other managers about the decision the cluster will go in an inconsistent state.

  Ex: If a new worker was to be added to the cluster by the leader without updating the other managers and if the leader fails then other manager will not be aware of the newly created worker node, and future cluster operation will ignore the newly added worker and services running in that worker will result in an inconsistent applications, this is known as problem of distribution consensus, So docker follows **RAFT algorithm**to ensure that all managers have same information about the cluster all the time .

RAFT Algorithm

To explain RAFT, let us take one example, suppose we have 3 manager nodes then RAFT algorithm will decide who is going to be leader among the three, if the leader has enough load available to make a decision and all decision are in consent with the other managers.

RAFT algorithm uses random timer for initiating requests, for example a random timer is kicked off on three managers, the first manager node which finish the timers sends the requests to other two managers to seek the permission to be the leader, and if both other manager votes in favour of 1st manager then 1st manager becomes the leader, after 1st node becomes leader, it will be duty of 1st node to send out the notification in regular interval to other two master informing them it is continuing to assume the role of the leader, in case other node don't receives a notification from the leader at any point of time which can either be due to reason leader went down or due to network issue leader server went down then 2 nodes initiates the re-election process between them and the new leader is identified.

Every manager has its own copy of RAFT database that stores all the information about the entire cluster process and they always remains in sync (synchronized), if leader need to add any worker node to cluster or do any modification in cluster then first leader need to send the notification to other two manager node and need to get response from at least (n/2+1) manager to make the changes, means if 3 manager is present then he need to take permission from at least 2 manager which confirms that any changes in the cluster took place with consent of majority of the managers in the cluster.

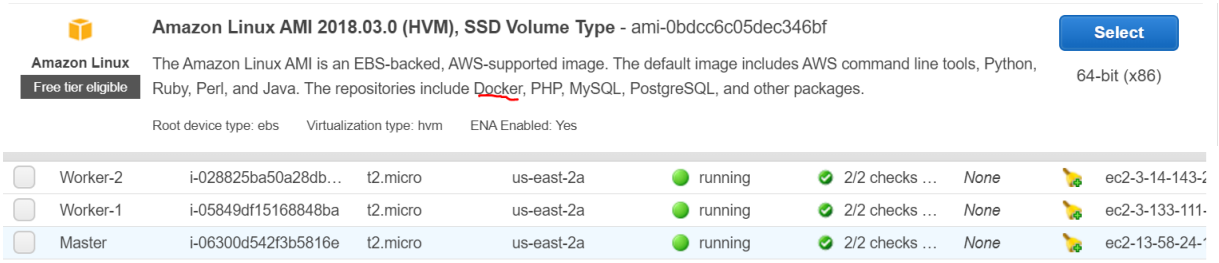
This process of making changes in cluster is known as Quorum → Quorum is defined as minimum number of persons in an assembly that must be present in any of the meeting to make the proceedings of the meeting valid.

Suppose 5 managers present then quorum will be 3 managers, if total 7 managers then quorum will be 4. We can add as many as manager we want in the cluster but docker suggests us to make maximum of 7 manager as adding more manager doesn't increase scalability or performance.

Always any modification we can made in the cluster if majority of the managers gives vote in favour, if suppose we were having 3 manager node and now 2 manager went down now in this case we can't make any modification in the cluster, we have only two option, *Either bring at least one manager node up or force recreate the cluster using the command "docker swarm init –force-new-cluster".* By default all managers node are also worker nodes, means whatever service we runs in manager nodes it runs gets distributed among manager node as well, we can disable this functionality and dedicate manager node for only management tasks, to do so we have to run the command " docker node update –availability drain node_name", this step is most recommended in production environment.

Swarm Setup

1. Launch Three Amazon Linux VM's.



2. Install docker on all three VM's.

```
[ec2-user@ip-172-31-4-211 ~]$ sudo yum install docker
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main                                                                 | 2.1 kB  00:00:00
amzn-updates                                                              | 3.8 kB  00:00:00
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:19.03.6ce-4.58.amzn1 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Processing Dependency: xfsprogs for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Processing Dependency: pigz for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Processing Dependency: libseccomp.so.2()(64bit) for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Running transaction check
---> Package containerd.x86_64 0:1.3.2-1.3.amzn1 will be installed
---> Package libseccomp.x86_64 0:2.3.1-2.4.amzn1 will be installed
---> Package pigz.x86_64 0:2.3.3-1.6.amzn1 will be installed
---> Package runc.x86_64 0:1.0.0-0.1.20200204.gitdc9208a.1.amzn1 will be installed
---> Package xfsprogs.x86_64 0:4.5.0-18.23.amzn1 will be installed
```

```
[ec2-user@ip-172-31-8-4 ~]$ docker --version
Docker version 19.03.6-ce, build 369ce74
[ec2-user@ip-172-31-8-4 ~]$
```

3. Docker is not working run below commands

```
sudo groupadd docker
sudo usermod -aG docker ec2-user
newgrp docker

sudo service docker start
```

4. To initiate the docker swarm in manager server we use the command

   docker swarm init

   Before initiating the swarm if we run "docker info | grep -i swarm" we can see the output that swarm is inactive mode.

```
[ec2-user@ip-172-31-12-36 ~]$ docker info | grep swarm
[ec2-user@ip-172-31-12-36 ~]$ docker info | grep -i swarm
 Swarm: inactive
[ec2-user@ip-172-31-12-36 ~]$ docker swarm init
Swarm initialized: current node (6prs676umbk1l8zbst12g81yp) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-34do05uutovuvr7wduzbrpibl 172.31.12.3
6:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[ec2-user@ip-172-31-12-36 ~]$ docker info | grep -i swarm
 Swarm: active
[ec2-user@ip-172-31-12-36 ~]$
```


..... Prepared by PPNREDDY

5. Add below command in worker nodes

```
docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-34do05uutovuvr7wduzbrpibl 172.31.12.36:2377
```

```
[root@ip-172-31-4-211 docker]# docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-34do05uuto
vuvr7wduzbrpibl 172.31.12.36:2377
This node joined a swarm as a worker.
[root@ip-172-31-4-211 docker]# exit
```

```
[ec2-user@ip-172-31-8-4 ~]$ docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-34do05uutovuv
r7wduzbrpibl 172.31.12.36:2377
This node joined a swarm as a worker.
[ec2-user@ip-172-31-8-4 ~]$
```

6. Verify the nodes in cluster using below command

docker node ls

```
[ec2-user@ip-172-31-12-36 etc]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
[ec2-user@ip-172-31-12-36 etc]$
```

Note:

If manager status will be Leader means it is primary manager node, if status will be reachable means they are secondary manager node which could come up in case the primary manager node goes down, if manager status will be nothing means they are worker node.

Adding New manager and workers to the existing swarm

After running "docker swarm init" in the manager node we will get the join token as output, running that token we can add any number of nodes into this cluster. Now we can get the token for worker node as well as manager node using the commands.

docker swarm join-token manager

docker swarm join-token worker

1. Adding manager using below command

   docker swarm join-token manager

```
[ec2-user@master ~]$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-dt9on3i9xkmxkcxb4n6jsmijc 172.31.12.3
6:2377

[ec2-user@master ~]$
```

Copy the command in newly created node

```
docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-dt9on3i9xkmxkcxb4n6jsmijc 172.31.12.36:2377
```

```
[ec2-user@manager2 ~]$ docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-dt9on3i9xkmxkcxb4n
6jsmijc 172.31.12.36:2377
This node joined a swarm as a manager.
[ec2-user@manager2 ~]$
```

Verify the cluster status

```
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
56vn8w113p3tt4ju2ew12xl7j     manager2            Ready               Active              Reachable           19.03.6-ce
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
[ec2-user@master ~]$
```

2. Adding worker using below command

docker swarm join-token worker

```
[ec2-user@master ~]$ docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-34do05uutovuvr7wduzbrpibl 172.31.12.3
6:2377

[ec2-user@master ~]$
```

Copy the command in newly created node

docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-34do05uutovuvr7wduzbrpibl 172.31.12.36:2377

```
[ec2-user@worker3 ~]$ docker swarm join --token SWMTKN-1-1c06v3vcamzzo7uuw4woscs01p4amcncxkq5da0m2aaciti44q-34do05uutovuvr7wduz
brpibl 172.31.12.36:2377
This node joined a swarm as a worker.
[ec2-user@worker3 ~]$
```

Verify the cluster status

```
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
56vn8w113p3tt4ju2ew12xl7j     manager2            Ready               Active              Reachable           19.03.6-ce
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
ohj6fjtgcr2rlq1rs9i8ezvml     worker3             Ready               Active                                  19.03.6-ce
[ec2-user@master ~]$
```

Promote and demote commands in docker swarm

Promote: Converting existing worker node to master node in the swarm

Demote: Converting existing master node to worker node in the swarm

Cluster current status

```
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
56vn8w113p3tt4ju2ew12xl7j     manager2            Ready               Active              Reachable           19.03.6-ce
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
ohj6fjtgcr2rlq1rs9i8ezvml     worker3             Ready               Active                                  19.03.6-ce
[ec2-user@master ~]$
```

After promoting worker3 node using below command

docker node promote worker3

```
[ec2-user@master ~]$ docker node promote worker3
Node worker3 promoted to a manager in the swarm.
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
56vn8w113p3tt4ju2ew12xl7j     manager2            Ready               Active              Reachable           19.03.6-ce
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
ohj6fjtgcr2rlq1rs9i8ezvml     worker3             Ready               Active              Reachable           19.03.6-ce
[ec2-user@master ~]$
```

Demoting the manager2 node using below command

docker node demote manager2

```
[ec2-user@master ~]$ docker node demote manager2
Manager manager2 demoted in the swarm.
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
56vn8w113p3tt4ju2ew12xl7j     manager2            Ready               Active                                  19.03.6-ce
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
ohj6fjtgcr2rlq1rs9i8ezvml     worker3             Ready               Active              Reachable           19.03.6-ce
[ec2-user@master ~]$
```

Removing the nodes from cluster

Removing the worker nodes from cluster

1. Go to the worker node and run the below command.

   docker swarm leave

```
[ec2-user@manager2 ~]$ docker swarm leave
Node left the swarm.
[ec2-user@manager2 ~]$
```

   Remove the node in the cluster displaying inactive/down

```
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
56vn8w113p3tt4ju2ew12xl7j     manager2            Down                Active                                  19.03.6-ce
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
ohj6fjtgcr2rlq1rs9i8ezvml     worker3             Ready               Active              Reachable           19.03.6-ce
[ec2-user@master ~]$ docker node rm manager2
manager2
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
ohj6fjtgcr2rlq1rs9i8ezvml     worker3             Ready               Active              Reachable           19.03.6-ce
[ec2-user@master ~]$
```

Removing the manager nodes from cluster

1. Demote the manger node in the swarm manager cluster

```
[ec2-user@worker3 ~]$ docker swarm leave
Error response from daemon: You are attempting to leave the swarm on a node that is participating as a manager. Removing this n
ode leaves 1 managers out of 2. Without a Raft quorum your swarm will be inaccessible. The only way to restore a swarm that has
 lost consensus is to reinitialize it with `--force-new-cluster`. Use `--force` to suppress this message.
[ec2-user@worker3 ~]$
```

```
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
ohj6fjtgcr2rlq1rs9i8ezvml     worker3             Ready               Active              Reachable           19.03.6-ce
[ec2-user@master ~]$ docker node demote worker3
Manager worker3 demoted in the swarm.
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
ohj6fjtgcr2rlq1rs9i8ezvml     worker3             Ready               Active                                  19.03.6-ce
```

2. Go to the worker node and run the below command.

   docker swarm leave

```
[ec2-user@worker3 ~]$ docker swarm leave
Node left the swarm.
[ec2-user@worker3 ~]$
```

```
[ec2-user@master ~]$ docker node rm worker3
worker3
[ec2-user@master ~]$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
6prs676umbk1l8zbst12g81yp *   master              Ready               Active              Leader              19.03.6-ce
xzwjmunjg4ijupv8hbb873b3f     worker1             Ready               Active                                  19.03.6-ce
htgk97g07zocd0jxiq5x6bbc5     worker2             Ready               Active                                  19.03.6-ce
[ec2-user@master ~]$
```

Creating the containers in the Swarm

In docker we use run command to create any container likewise we use service in docker swarm, all the options which we used in docker run command we can use it with docker service.

docker service create --name my-tomcat --replicas 10 -p 1234:8080 tomcat

In docker swarm we use –replicas to scale up out container to the desired count.

Once we run above command 10 tomcat containers will be created.

Verify the services in Swarm

docker service ls

```
[ec2-user@ip-172-31-24-221 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
j7jq12gpvskd        my-tomcat           replicated          10/10               tomcat:latest       *:1234->8080/tcp
[ec2-user@ip-172-31-24-221 ~]$
```

Container is running in which server

docker service ps my-tomcat

```
[ec2-user@ip-172-31-24-221 ~]$ docker service ps my-tomcat
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE
      ERROR               PORTS
wxzpjllw3z5o        my-tomcat.1         tomcat:latest       worker1             Running             Running 6 minutes
 ago
3s0u1xa4fmjt        my-tomcat.2         tomcat:latest       worker2             Running             Running 6 minutes
 ago
5shhkde0qnur        my-tomcat.3         tomcat:latest       worker2             Running             Running 6 minutes
 ago
aiwngzrhd0x8        my-tomcat.4         tomcat:latest       worker1             Running             Running 6 minutes
 ago
x9l9r4bckhjf        my-tomcat.5         tomcat:latest       manager             Running             Running 6 minutes
 ago
ywqttrt547kt        my-tomcat.6         tomcat:latest       manager             Running             Running 6 minutes
 ago
m2gpuo1rox3i        my-tomcat.7         tomcat:latest       worker2             Running             Running 6 minutes
 ago
kj5i0zklfv9h        my-tomcat.8         tomcat:latest       manager             Running             Running 6 minutes
 ago
lvjlfga4lnms        my-tomcat.9         tomcat:latest       manager             Running             Running 6 minutes
 ago
ljky2qdvvubg        my-tomcat.10        tomcat:latest       worker1             Running             Running 6 minutes
 ago
[ec2-user@ip-172-31-24-221 ~]$
```

Deleting the service

docker service rm my-tomcat

```
[ec2-user@ip-172-31-24-221 ~]$ docker service rm  my-tomcat
my-tomcat
[ec2-user@ip-172-31-24-221 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
[ec2-user@ip-172-31-24-221 ~]$
```

Updating the replicas

docker service update --replicas 15 my-nginx

```
[ec2-user@ip-172-31-24-221 ~]$ docker service update --replicas 15 my-nginx
my-nginx
overall progress: 15 out of 15 tasks
1/15: running   [==================================================>]
2/15: running   [==================================================>]
3/15: running   [==================================================>]
4/15: running   [==================================================>]
5/15: running   [==================================================>]
6/15: running   [==================================================>]
7/15: running   [==================================================>]
8/15: running   [==================================================>]
9/15: running   [==================================================>]
10/15: running   [==================================================>]
11/15: running   [==================================================>]
12/15: running   [==================================================>]
13/15: running   [==================================================>]
14/15: running   [==================================================>]
15/15: running   [==================================================>]
verify: Service converged
```

Rollback

docker service create --name my-nginx --replicas 10 -p 80:80 nginx:latest

```
[ec2-user@ip-172-31-24-221 ~]$ docker service create --name my-nginx --replicas 10 -p 80:80 nginx:latest
zkyijhrvby5xujvsfvwkizpr9
overall progress: 10 out of 10 tasks
1/10: running   [==================================================>]
2/10: running   [==================================================>]
3/10: running   [==================================================>]
4/10: running   [==================================================>]
5/10: running   [==================================================>]
6/10: running   [==================================================>]
7/10: running   [==================================================>]
8/10: running   [==================================================>]
9/10: running   [==================================================>]
10/10: running   [==================================================>]
verify: Service converged
[ec2-user@ip-172-31-24-221 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
zkyijhrvby5x        my-nginx            replicated          10/10               nginx:latest        *:80->80/tcp
```

docker service update --replicas 15 my-nginx

```
[ec2-user@ip-172-31-24-221 ~]$ docker service update --replicas 15 my-nginx
my-nginx
overall progress: 15 out of 15 tasks
1/15: running   [==================================================>]
2/15: running   [==================================================>]
3/15: running   [==================================================>]
4/15: running   [==================================================>]
5/15: running   [==================================================>]
6/15: running   [==================================================>]
7/15: running   [==================================================>]
8/15: running   [==================================================>]
9/15: running   [==================================================>]
10/15: running   [==================================================>]
11/15: running   [==================================================>]
12/15: running   [==================================================>]
13/15: running   [==================================================>]
14/15: running   [==================================================>]
15/15: running   [==================================================>]
verify: Service converged
[ec2-user@ip-172-31-24-221 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
zkyijhrvby5x        my-nginx            replicated          15/15               nginx:latest        *:80->80/tcp
```

……. Prepared by PPNREDDY

docker service rollback my-nginx

```
[ec2-user@ip-172-31-24-221 ~]$ docker service rollback my-nginx
my-nginx
rollback: manually requested rollback
overall progress: rolling back update: 15 out of 10 tasks
1/10: running   [>                                                  ]
2/10: running   [>                                                  ]
3/10: running   [>                                                  ]
4/10: running   [>                                                  ]
5/10: running   [>                                                  ]
6/10: running   [>                                                  ]
7/10: running   [>                                                  ]
8/10: running   [>                                                  ]
9/10: running   [>                                                  ]
10/10: running   [>                                                 ]
service rolled back: rollback completed
[ec2-user@ip-172-31-24-221 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
zkyijhrvby5x        my-nginx            replicated          10/10               nginx:latest        *:80->80/tcp
```

Scale vs Replicas

By using scale we can update multiple services but by using replicas we can update single service.

```
[ec2-user@ip-172-31-24-221 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
zkyijhrvby5x        my-nginx            replicated          10/10               nginx:latest        *:80->80/tcp
dsvyonvdge3p        mytommy             replicated          10/10               tomcat:latest       *:8080->8080/tcp
```

```
[ec2-user@ip-172-31-24-221 ~]$ docker service update --replicas 15 my-nginx
my-nginx
overall progress: 15 out of 15 tasks
1/15: running   [==================================================>]
2/15: running   [==================================================>]
3/15: running   [==================================================>]
4/15: running   [==================================================>]
5/15: running   [==================================================>]
6/15: running   [==================================================>]
7/15: running   [==================================================>]
8/15: running   [==================================================>]
9/15: running   [==================================================>]
10/15: running   [==================================================>]
11/15: running   [==================================================>]
12/15: running   [==================================================>]
13/15: running   [==================================================>]
14/15: running   [==================================================>]
15/15: running   [==================================================>]
verify: Service converged
```

```
[ec2-user@ip-172-31-24-221 ~]$ docker service update --replicas 5 mytommy
mytommy
overall progress: 5 out of 5 tasks
1/5: running   [==================================================>]
2/5: running   [==================================================>]
3/5: running   [==================================================>]
4/5: running   [==================================================>]
5/5: running   [==================================================>]
verify: Service converged
[ec2-user@ip-172-31-24-221 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
zkyijhrvby5x        my-nginx            replicated          15/15               nginx:latest        *:80->80/tcp
dsvyonvdge3p        mytommy             replicated          5/5                 tomcat:latest       *:8080->8080/tcp
[ec2-user@ip-172-31-24-221 ~]$
```

```
[ec2-user@ip-172-31-24-221 ~]$ docker service scale my-nginx=10 mytommy=10
my-nginx scaled to 10
mytommy scaled to 10
overall progress: 10 out of 10 tasks
1/10: running   [==================================================>]
2/10: running   [==================================================>]
3/10: running   [==================================================>]
4/10: running   [==================================================>]
5/10: running   [==================================================>]
6/10: running   [==================================================>]
7/10: running   [==================================================>]
8/10: running   [==================================================>]
9/10: running   [==================================================>]
10/10: running   [==================================================>]
verify: Service converged
overall progress: 10 out of 10 tasks
1/10: running   [==================================================>]
2/10: running   [==================================================>]
3/10: running   [==================================================>]
4/10: running   [==================================================>]
5/10: running   [==================================================>]
6/10: running   [==================================================>]
7/10: running   [==================================================>]
8/10: running   [==================================================>]
9/10: running   [==================================================>]
10/10: running   [==================================================>]
verify: Service converged
[ec2-user@ip-172-31-24-221 ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
zkyijhrvby5x        my-nginx            replicated          10/10               nginx:latest        *:80->80/tcp
dsvyonvdge3p        mytommy             replicated          10/10               tomcat:latest       *:8080->8080/tcp
[ec2-user@ip-172-31-24-221 ~]$
```

Global mode

In replicas we can use the mode as global also, if mode will be global then a single container will come up in all host server which are part of swarm cluster:

docker service create --name my-nginx --mode global -p 80:80 nginx:latest

```
[ec2-user@ip-172-31-24-221 ~]$ docker service create --name my-nginx --mode global -p 80:80 nginx:latest
ujo0c3ckus4wx0dy6ha0ayl59
overall progress: 3 out of 3 tasks
2xyj24vcd1le: running   [==================================================>]
xt3pxljy5tqp: running   [==================================================>]
ln0titr185co: running   [==================================================>]
verify: Service converged
[ec2-user@ip-172-31-24-221 ~]$ docker service ps my-nginx
ID                  NAME                                    IMAGE               NODE                DESIRED STATE
CURRENT STATE           ERROR               PORTS
dx52wwt5znwp        my-nginx.xt3pxljy5tqpa6qkb44iy3ock   nginx:latest        manager             Running
Running 21 seconds ago
smqc8az81mrg        my-nginx.ln0titr185coevwoh84dj1r7p   nginx:latest        worker2             Running
Running 21 seconds ago
7lwuy876ad10        my-nginx.2xyj24vcd1legz69v8b9awyg4   nginx:latest        worker1             Running
Running 21 seconds ago
[ec2-user@ip-172-31-24-221 ~]$
```

docker node update --availability drain manager

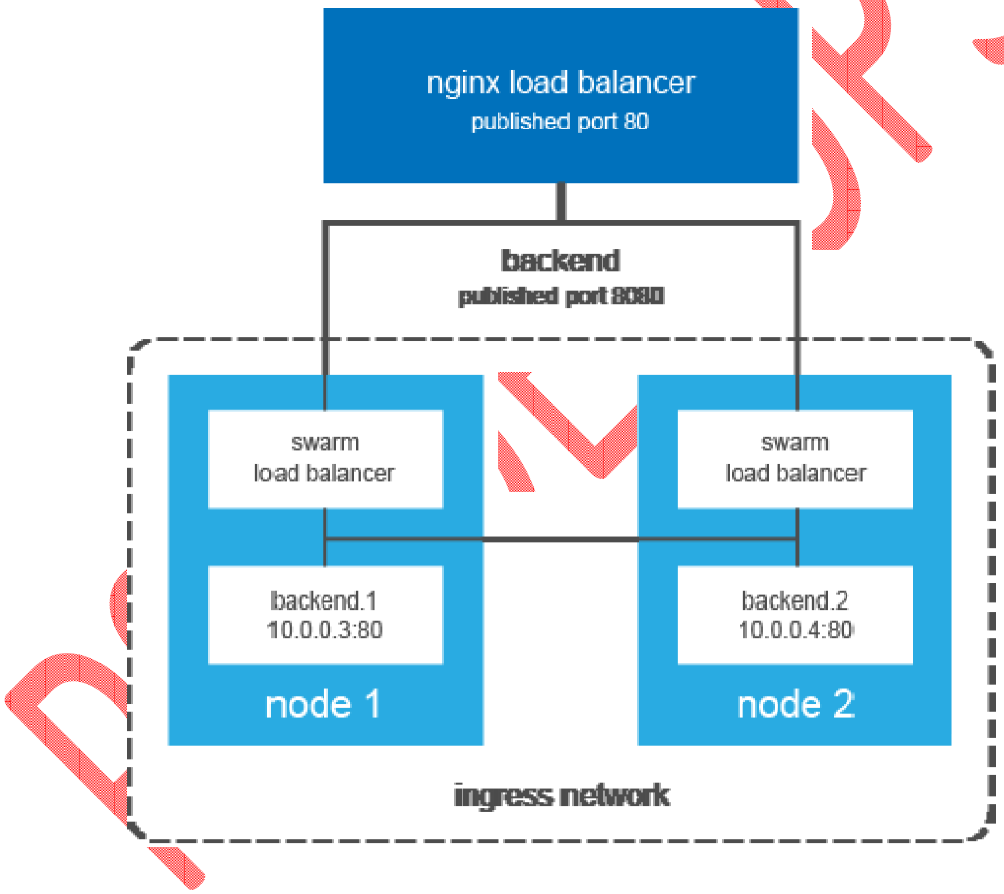
```
[ec2-user@ip-172-31-24-221 ~]$ docker node update --availability drain manager
manager
[ec2-user@ip-172-31-24-221 ~]$ docker service ps my-nginx
ID                  NAME                                    IMAGE               NODE                DESIRED STATE
CURRENT STATE           ERROR               PORTS
dx52wwt5znwp        my-nginx.xt3pxljy5tqpa6qkb44iy3ock   nginx:latest        manager             Shutdown
Shutdown 1 second ago
smqc8az81mrg        my-nginx.ln0titr185coevwoh84dj1r7p   nginx:latest        worker2             Running
Running 2 minutes ago
7lwuy876ad10        my-nginx.2xyj24vcd1legz69v8b9awyg4   nginx:latest        worker1             Running
Running 2 minutes ago
```

docker node update –availability active manager

```
[ec2-user@ip-172-31-24-221 ~]$ docker node update --availability active manager
manager
[ec2-user@ip-172-31-24-221 ~]$ docker service ps my-nginx
ID                  NAME                                    IMAGE               NODE                DESIRED STATE
    CURRENT STATE           ERROR               PORTS
hv1ww7ao9yc4        my-nginx.xt3pxljy5tqpa6qkb44iy3ock      nginx:latest        manager             Running
    Running 3 seconds ago
dx52wwt5znwp         \_ my-nginx.xt3pxljy5tqpa6qkb44iy3ock  nginx:latest        manager             Shutdown
    Shutdown 34 seconds ago
smqc8az81mrg        my-nginx.ln0titr185coevwoh84dj1r7p      nginx:latest        worker2             Running
    Running 3 minutes ago
7lwuy876ad10        my-nginx.2xyj24vcd1legz69v8b9awyg4      nginx:latest        worker1             Running
    Running 3 minutes ago
[ec2-user@ip-172-31-24-221 ~]$
```

Docker Swarm load balancer

The *Docker Swarm load balancer* runs on every node and can load balance requests across any of the containers on any of the hosts in the cluster.



Ex:

1. Create volume by using below command
   docker volume create psddevops

```
[ec2-user@manager ~]$ docker volume create psddevops
psddevops
[ec2-user@manager ~]$
```

```
[root@manager docker]# cd volumes
[root@manager volumes]# ll
total 28
-rw------- 1 root root 32768 Jul 25 01:33 metadata.db
drwxr-xr-x 3 root root  4096 Jul 25 01:33 psddevops
[root@manager volumes]# pwd
/var/lib/docker/volumes
[root@manager volumes]#
```

2. Create the service using below command
   docker service create --name my-nginx --replicas 10 --mount source=psddevops,target=/usr/share/nginx/html -p 80:80 nginx:latest

```
[root@manager volumes]# docker service create --name my-nginx --replicas 10 --mount source=psddevops,target=/usr/share/nginx/html -p 80:80 nginx:latest
y8iolk9qbk0ndvruhlo4tp2ap
overall progress: 10 out of 10 tasks
1/10: running   [==================================================>]
2/10: running   [==================================================>]
3/10: running   [==================================================>]
4/10: running   [==================================================>]
5/10: running   [==================================================>]
6/10: running   [==================================================>]
7/10: running   [==================================================>]
8/10: running   [==================================================>]
9/10: running   [==================================================>]
10/10: running   [==================================================>]
verify: Service converged
[root@manager volumes]#
```

3. Update the ***html***file in in all servers located in below path
   /var/lib/docker/volumes/psddevops/_data

```
<body>
<h1>Welcome to nginx! this is master server</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

```
<body>
<h1>Welcome to nginx! This is worker node1</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

```
<body>
<h1>Welcome to nginx! This is worker node2</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

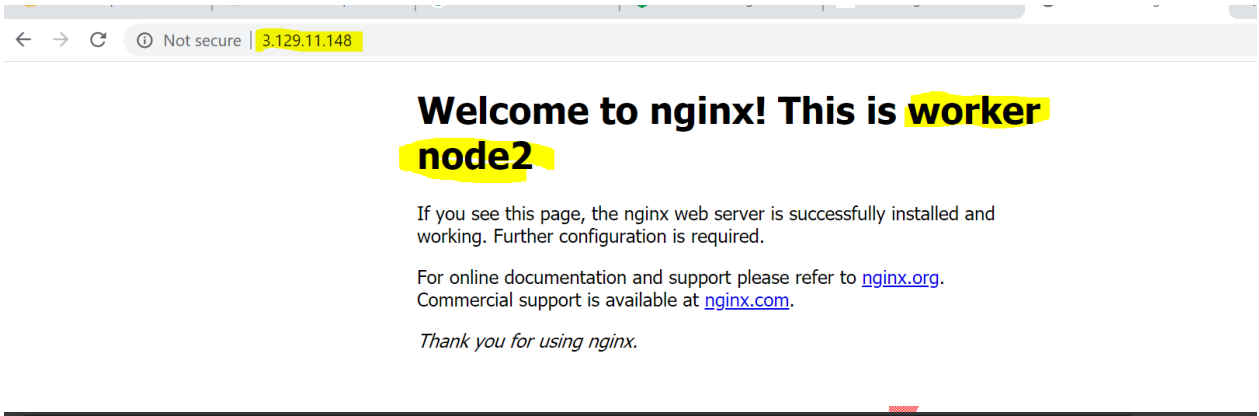
4. Verify the nginx server with same IP address.

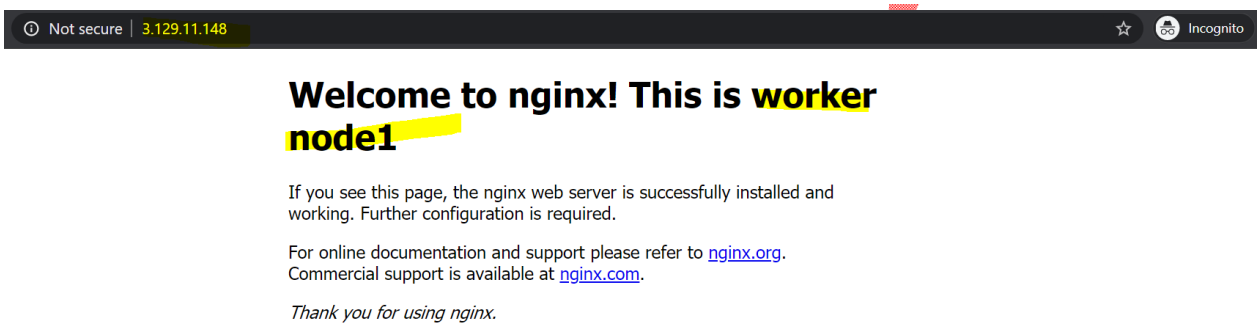
Welcome to nginx! this is master server

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.





Creating the services on particular node in the cluster

By using labels concept, we can achieve this type of scenario.

docker inspect manager

```
[ec2-user@manager ~]$ docker inspect manager
[
    {
        "ID": "xt3pxljy5tqpa6qkb44iy3ock",
        "Version": {
            "Index": 975
        },
        "CreatedAt": "2020-07-23T14:01:27.802472807Z",
        "UpdatedAt": "2020-07-25T01:38:45.350102409Z",
        "Spec": {
            "Labels": {},
            "Role": "manager",
            "Availability": "active"
        },
```

```
[ec2-user@manager ~]$ docker inspect worker1
[
    {
        "ID": "2xyj24vcd1legz69v8b9awyg4",
        "Version": {
            "Index": 941
        },
        "CreatedAt": "2020-07-23T14:06:47.262418629Z",
        "UpdatedAt": "2020-07-25T01:07:45.343261014Z",
        "Spec": {
            "Labels": {},
            "Role": "worker",
            "Availability": "active"
        },
```

```
[ec2-user@manager ~]$ docker inspect worker2
[
    {
        "ID": "ln0titr185coevwoh84dj1r7p",
        "Version": {
            "Index": 943
        },
        "CreatedAt": "2020-07-23T14:06:55.272000248Z",
        "UpdatedAt": "2020-07-25T01:08:05.191196467Z",
        "Spec": {
            "Labels": {},
            "Role": "worker",
            "Availability": "active"
        },
```

Adding the label

```
[ec2-user@manager ~]$ docker node update --help

Usage:  docker node update [OPTIONS] NODE

Update a node

Options:
      --availability string   Availability of the node ("active"|"pause"|"drain")
      --label-add list        Add or update a node label (key=value)
      --label-rm list         Remove a node label if exists
      --role string           Role of the node ("worker"|"manager")
[ec2-user@manager ~]$
```

docker node update --label-add region=east worker2

```
[ec2-user@manager ~]$ docker node update --label-add region=east worker2
worker2
[ec2-user@manager ~]$ docker inspect worker2
[
    {
        "ID": "ln0titr185coevwoh84dj1r7p",
        "Version": {
            "Index": 1170
        },
        "CreatedAt": "2020-07-23T14:06:55.272000248Z",
        "UpdatedAt": "2020-07-25T03:04:19.906367957Z",
        "Spec": {
            "Labels": {
                "region": "east"
            },
            "Role": "worker",
            "Availability": "active"
```

```
[ec2-user@manager ~]$ docker service create --name my-nginx --replicas 10 --constraint node.labels.region==east -p 80
:80 nginx:latest
w0j0yf25ljkawmffdznqawu50
overall progress: 10 out of 10 tasks
1/10: running   [==================================================>]
2/10: running   [==================================================>]
3/10: running   [==================================================>]
4/10: running   [==================================================>]
5/10: running   [==================================================>]
6/10: running   [==================================================>]
7/10: running   [==================================================>]
8/10: running   [==================================================>]
9/10: running   [==================================================>]
10/10: running   [==================================================>]
verify: Service converged
```

Docker Secrets

- A secret is a piece of data, such as a password, SSH private key, SSL certificate, or another piece of data that should not be transmitted over a network or stored unencrypted in a Dockerfile or in your application's source code.
- In Docker 1.13 and higher, you can use Docker secrets to centrally manage this data and securely transmit it to only those containers that need access to it.
- Docker Secrets is only available in the Swarm mode, so standalone containers can't use this feature.
- A given secret is only accessible to those services which have been granted explicit access to it, and only while those service tasks are running.

How Docker Swarm manage the secrets

- When a user adds a new secret to a Swarm cluster, this secret is sent to a manager using a TLS connection. TLS is a cryptographic protocol that provides communications security over a network by providing communication encryption, privacy and data integrity.
- When we have multiple Managers, RAFT algorithm manages the secrets on all the managers. Containers work on mounted decrypted secrets, which store at /run/secrets/<secret_name> in containers.
- User can update a service to grant it access to additional secrets or revoke its access to a given secret at any time.
- When container task stops running, the decrypted secrets shared to it are unmounted from the in-memory filesystem for that container and flushed from the node's memory.

Secrets creation

There are 2-ways to create the secrets

1. Using file
2. Using CLI command

Using file

Create a folder and add a file with secret data

```
[ec2-user@manager ~]$ mkdir secrets
[ec2-user@manager ~]$ cd secrets
[ec2-user@manager secrets]$ vi cred.txt
[ec2-user@manager secrets]$ cat cred.txt
India@123
[ec2-user@manager secrets]$ pwd
/home/ec2-user/secrets
[ec2-user@manager secrets]$
```

docker secret –help

docker secret create db_pass cred.txt

docker secret ls

docker secret inspect db_pass

```
[ec2-user@manager secrets]$ docker secret --help

Usage:  docker secret COMMAND

Manage Docker secrets

Commands:
  create      Create a secret from a file or STDIN as content
  inspect     Display detailed information on one or more secrets
  ls          List secrets
  rm          Remove one or more secrets

Run 'docker secret COMMAND --help' for more information on a command.
[ec2-user@manager secrets]$ docker secret create db_pass cred.txt
sktlz15if5v8gumbeps4p3o6f
[ec2-user@manager secrets]$ docker secret ls
ID                          NAME                DRIVER              CREATED             UPDATED
sktlz15if5v8gumbeps4p3o6f   db_pass                                 17 seconds ago      17 seconds ago
[ec2-user@manager secrets]$ docker secret inspect db_pass
[
    {
        "ID": "sktlz15if5v8gumbeps4p3o6f",
        "Version": {
            "Index": 1187
        },
        "CreatedAt": "2020-07-25T08:12:57.706184598Z",
        "UpdatedAt": "2020-07-25T08:12:57.706184598Z",
        "Spec": {
            "Name": "db_pass",
            "Labels": {}
        }
    }
]
[ec2-user@manager secrets]$
```

Using CLI command

```
[ec2-user@manager secrets]$ echo "scc-db" | docker secret create db_user -
znoqdjspe8vcquqwfs8ltueki
[ec2-user@manager secrets]$ docker secret ls
ID                          NAME                DRIVER              CREATED             UPDATED
sktlz15if5v8gumbeps4p3o6f   db_pass                                 6 minutes ago       6 minutes ago
znoqdjspe8vcquqwfs8ltueki   db_user                                 10 seconds ago      10 seconds ago
[ec2-user@manager secrets]$ docker secret inspect db_user
[
    {
        "ID": "znoqdjspe8vcquqwfs8ltueki",
        "Version": {
            "Index": 1188
        },
        "CreatedAt": "2020-07-25T08:19:42.40257181Z",
        "UpdatedAt": "2020-07-25T08:19:42.40257181Z",
        "Spec": {
            "Name": "db_user",
            "Labels": {}
        }
    }
]
[ec2-user@manager secrets]$
```

docker service create --name my-nginx --secret db_pass --secret db_user -e DATABASE_USER_FILE=/run/secrets/db_user -e DATABASE_PASSWORD_FILE=/run/secrets/db_pass --replicas 5 -p 80:80 nginx:latest

……. Prepared by PPNREDDY

```
[ec2-user@manager ~]$ docker service create --name my-nginx --secret db_pass --secret db_user -e DATABASE_USER_FILE=/run/secret
s/db_user -e DATABASE_PASSWORD_FILE=/run/secrets/db_pass --replicas 5 -p 80:80 nginx:latest
4x68czfjzt51nhpx9zf551x85
overall progress: 5 out of 5 tasks
1/5: running   [==================================================>]
2/5: running   [==================================================>]
3/5: running   [==================================================>]
4/5: running   [==================================================>]
5/5: running   [==================================================>]
verify: Service converged
[ec2-user@manager ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NA
MES
b6b92a039032        nginx:latest        "/docker-entrypoint.…"   2 minutes ago       Up 2 minutes        80/tcp              my
-nginx.5.znl3dgkcb9fw9pshce1q9nxe3
f129e93cbeb9        nginx:latest        "/docker-entrypoint.…"   2 minutes ago       Up 2 minutes        80/tcp              my
-nginx.4.ggw9dmundyea9784azgirgwwj
[ec2-user@manager ~]$ docker exec -it b6b92a039032 bash
root@b6b92a039032:/# cd /run/secrets
root@b6b92a039032:/run/secrets# ll
bash: ll: command not found
root@b6b92a039032:/run/secrets# ls
db_pass  db_user
root@b6b92a039032:/run/secrets# cat db_pass
India@123
root@b6b92a039032:/run/secrets# cat db_user
scc-db
root@b6b92a039032:/run/secrets#
```

Removing the docker secrets

docker secret rm db_pass

docker secret rm db_user

```
[ec2-user@manager ~]$ docker secret rm db_pass
db_pass
[ec2-user@manager ~]$ docker secret rm db_user
db_user
[ec2-user@manager ~]$ docker secret ls
ID                  NAME                DRIVER              CREATED             UPDATED
[ec2-user@manager ~]$
```

Docker Stack

By using docker we can create multiple services at a time in the docker swarm. docker compose is the prerequisite for this.

docker-compose.yml file

```
version: "3"
services:
  web:
    image: nginx
    ports:
      - "80:8080"
    deploy:
      replicas: 3
      restart_policy:
        condition: on-failure
  database:
    image: redis
```

```
[ec2-user@manager ~]$ docker-compose up -d
WARNING: Some services (web) use the 'deploy' key, which will be ignored. Compose does not support 'deploy' configuration - use
 `docker stack deploy` to deploy to a swarm.
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the curren
t node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "ec2-user_default" with the default driver
Pulling web (nginx:)...
latest: Pulling from library/nginx
Digest: sha256:0e188877aa60537d1a1c6484b8c3929cfe09988145327ee47e8e91ddf6f76f5c
Status: Downloaded newer image for nginx:latest
Pulling database (redis:)...
latest: Pulling from library/redis
6ec8c9369e08: Already exists
efe6cceb88f8: Pull complete
cdb6bd1ce7c5: Pull complete
9d80498f79fe: Pull complete
b7cd40c9247b: Pull complete
96403647fb55: Pull complete
Digest: sha256:d86d6739fab2eaf590cfa51eccf1e9779677bd2502894579bcf3f80cb37b18d4
Status: Downloaded newer image for redis:latest
Creating ec2-user_web_1      ... done
Creating ec2-user_database_1 ... done
[ec2-user@manager ~]$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
[ec2-user@manager ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
          NAMES
b079b1532ff3        redis               "docker-entrypoint.s…"   22 seconds ago      Up 21 seconds       6379/tcp
          ec2-user_database_1
fa5c4b6978a5        nginx               "/docker-entrypoint.…"   22 seconds ago      Up 21 seconds       80/tcp, 0.0.0.0:80->80
80/tcp    ec2-user_web_1
[ec2-user@manager ~]$
```

docker stack --help

```
[ec2-user@manager ~]$ docker stack --help

Usage:  docker stack [OPTIONS] COMMAND

Manage Docker stacks

Options:
      --orchestrator string   Orchestrator to use (swarm|kubernetes|all)

Commands:
  deploy      Deploy a new stack or update an existing stack
  ls          List stacks
  ps          List the tasks in the stack
  rm          Remove one or more stacks
  services    List the services in the stack

Run 'docker stack COMMAND --help' for more information on a command.
[ec2-user@manager ~]$
```

Create the Docker Stack: docker stack deploy -c docker-compose.yml myStack

List of Stack:docker stack ls

List of service:docker stack services myStack

Verify the services:docker stack ps myStack

```
[ec2-user@manager ~]$ docker stack deploy -c docker-compose.yml myStack
Creating network myStack_default
Creating service myStack_database
Creating service myStack_web
[ec2-user@manager ~]$
[ec2-user@manager ~]$ docker stack ls
NAME                SERVICES            ORCHESTRATOR
myStack             2                   Swarm
[ec2-user@manager ~]$ docker stack services ls
Nothing found in stack: ls
[ec2-user@manager ~]$ docker stack services myStack
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
cdvfarhpqva3        myStack_database    replicated          1/1                 redis:latest
rt5e1o0o8i0gt       myStack_web         replicated          3/3                 nginx:latest        *:80->8080/tcp
[ec2-user@manager ~]$ docker stack ps myStack
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE           ERROR
       PORTS
7628utnjsus6        myStack_web.1       nginx:latest        worker1             Running             Running 2 minutes ago

lqlhp2hrwjt5        myStack_database.1  redis:latest        worker1             Running             Running 2 minutes ago

zhc747gmm6kx        myStack_web.2       nginx:latest        manager             Running             Running 2 minutes ago

2p829fm97rx4        myStack_web.3       nginx:latest        worker2             Running             Running 2 minutes ago

[ec2-user@manager ~]$
```

..... Prepared by PPNREDDY

Remove the stack:docker stack rm myStack

```
[ec2-user@manager ~]$ docker stack ls
NAME                SERVICES            ORCHESTRATOR
myStack             2                   Swarm
[ec2-user@manager ~]$ docker stack rm myStack
Removing service myStack_database
Removing service myStack_web
Removing network myStack_default
[ec2-user@manager ~]$ docker stack ls
NAME                SERVICES            ORCHESTRATOR
[ec2-user@manager ~]$
```

Passing Secrets to the Docker stack

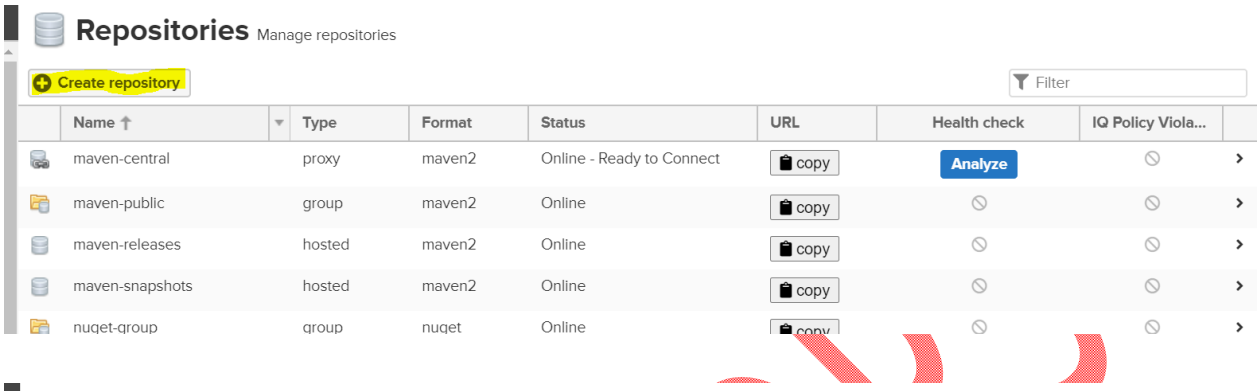
```
version: "3.1"
services:
  web:
    image: nginx
    secrets:
      - source: db_user
      - source: db_pass
    ports:
      - "80:8080"
    deploy:
      replicas: 3
      restart_policy:
        condition: on-failure
  database:
    image: redis
secrets:                    # top level secrets block
  db_user:
    external: true
  db_pass:
    external: true
```

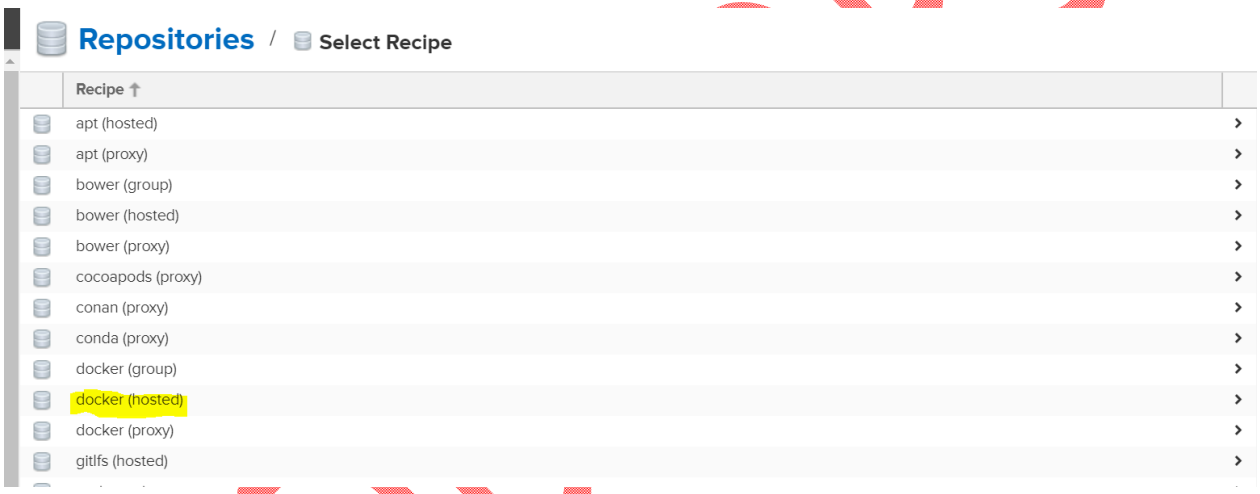


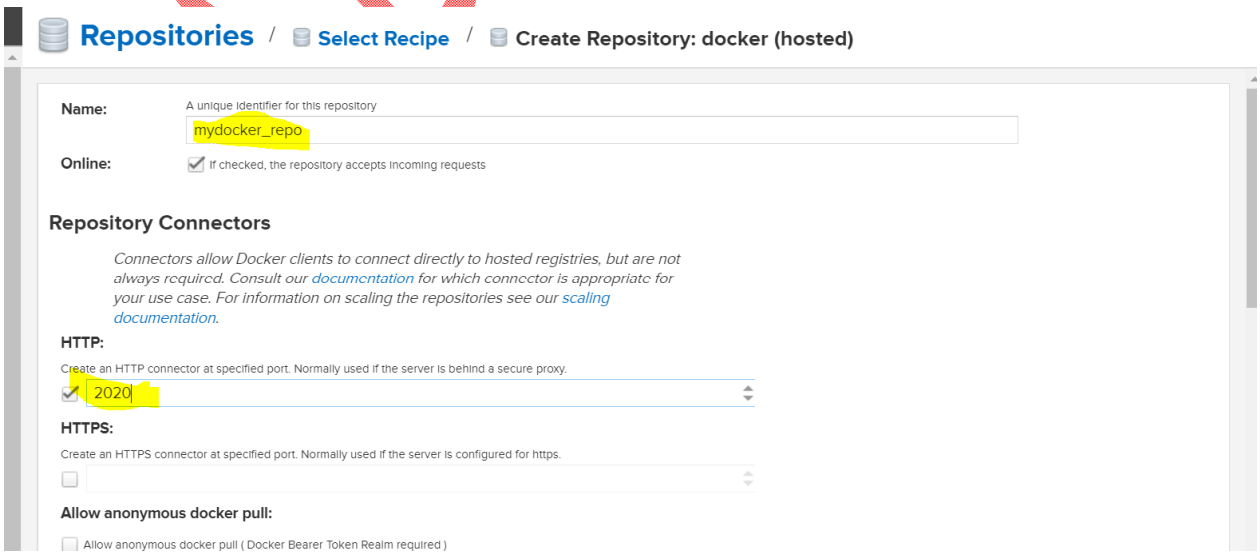
```
[ec2-user@ip-172-31-24-221 stack_demo]$ docker stack deploy -c docker-compose.yml myStack
Creating service myStack_database
Creating service myStack_web
[ec2-user@ip-172-31-24-221 stack_demo]$ docker stack ps myStack
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE           E
RROR                PORTS
snndysa7qaie        myStack_web.1       nginx:latest        ip-172-31-16-25     Running             Running 46 seconds ago

bvyfm0rosqxj        myStack_database.1  redis:latest        ip-172-31-25-33     Running             Running 45 seconds ago

pyiy2gk6q7qo        myStack_web.2       nginx:latest        ip-172-31-24-221    Running             Running 46 seconds ago

3lofvu4g14a7        myStack_web.3       nginx:latest        ip-172-31-25-33     Running             Running 45 seconds ago

[ec2-user@ip-172-31-24-221 stack_demo]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               N
AMES
df817d144e27        nginx:latest        "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp              m
yStack_web.2.pyiy2gk6q7qou8ekp54qluu8g
[ec2-user@ip-172-31-24-221 stack_demo]$ docker exec -it snndysa7qaie bash
Error: No such container: snndysa7qaie
[ec2-user@ip-172-31-24-221 stack_demo]$ docker exec -it df817d144e27 bash
root@df817d144e27:/# cd /run/secrets
root@df817d144e27:/run/secrets# ls
db_pass  db_user
root@df817d144e27:/run/secrets# cat db_pass
India@123
root@df817d144e27:/run/secrets# cat db_user
scc-db
root@df817d144e27:/run/secrets#
```

Uploading images to the nexus server

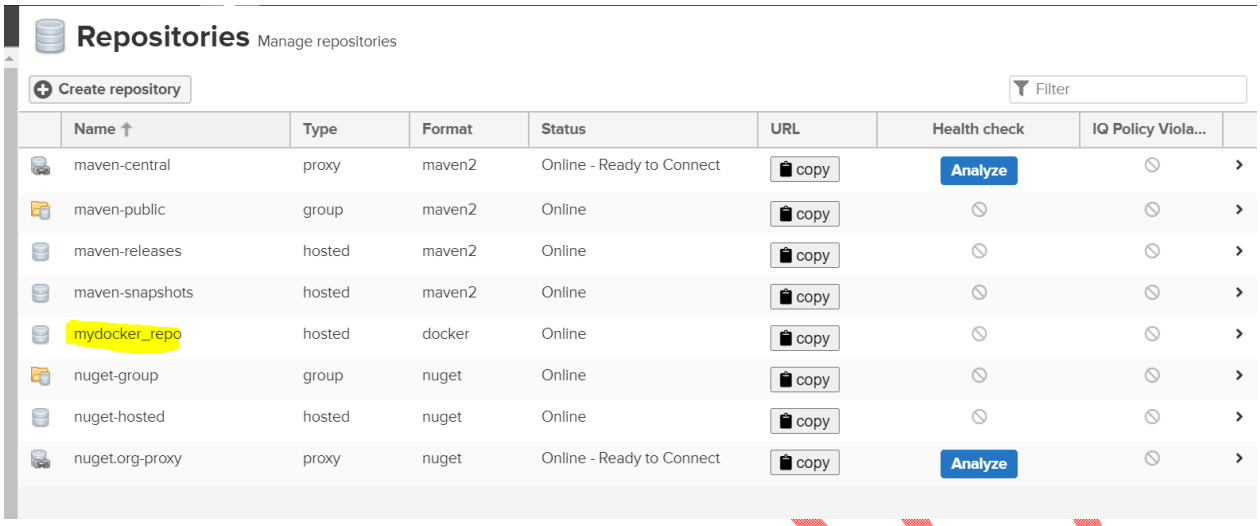
1. Create docker hosted repository

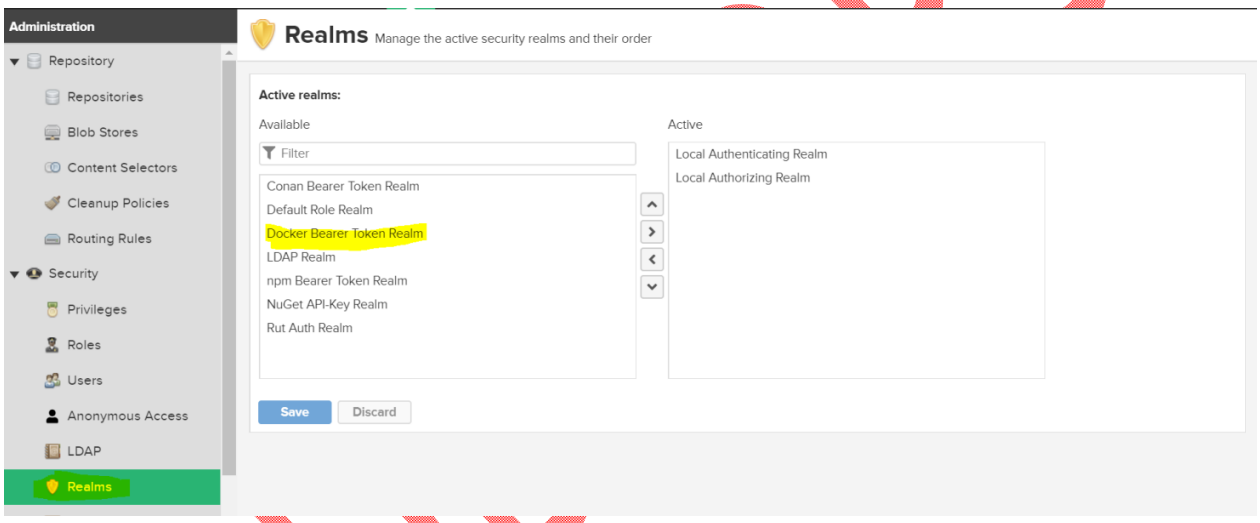






..... Prepared by PPNREDDY





2. Upload images to the nexus

Login to the docker and download any sample image

docker pull redis

```
[ec2-user@ip-172-31-24-221 ~]$ docker pull redis
Using default tag: latest
latest: Pulling from library/redis
6ec8c9369e08: Already exists
efe6cceb88f8: Pull complete
cdb6bd1ce7c5: Pull complete
9d80498f79fe: Pull complete
b7cd40c9247b: Pull complete
96403647fb55: Pull complete
Digest: sha256:d86d6739fab2eaf590cfa51eccf1e9779677bd2502894579bcf3f80cb37b18d4
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest
[ec2-user@ip-172-31-24-221 ~]$
```

Add insecure repos in daemon.json and restart docker service

cd /etc/docker

vi daemon.json

..... Prepared by PPNREDDY

```
{
  "insecure-registries" : ["3.129.25.247:2020"]
}
```

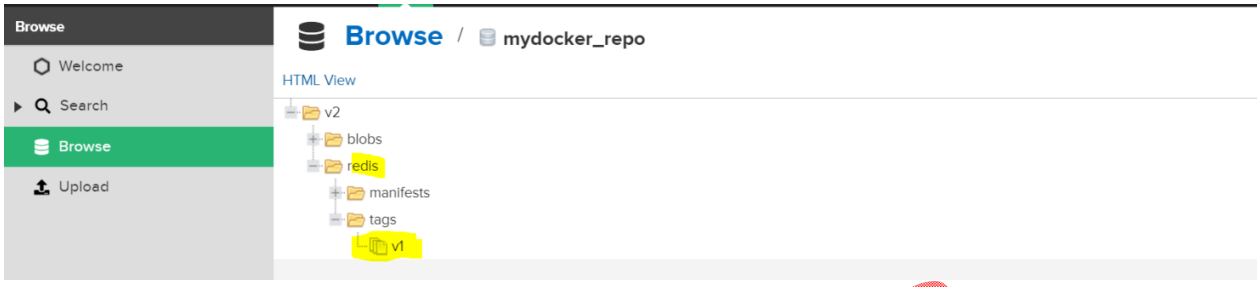
```
[ec2-user@ip-172-31-24-221 ~]$ sudo su
[root@ip-172-31-24-221 ec2-user]# cd /etc/docker
[root@ip-172-31-24-221 docker]# ll
total 4
-rw------- 1 root root 244 Jul 23 13:59 key.json
[root@ip-172-31-24-221 docker]# vi daemon.json
[root@ip-172-31-24-221 docker]# cat daemon.json
{
  "insecure-registries" : ["3.129.25.247:2020"]
}
[root@ip-172-31-24-221 docker]# systemctl restart docker
bash: systemctl: command not found
[root@ip-172-31-24-221 docker]# service docker restart
Stopping docker:                                           [  OK  ]
Starting docker:        .                                  [  OK  ]
[root@ip-172-31-24-221 docker]#
```

3. Create tag and upload the docker image

docker tag redis 3.129.25.247:2020/redis:v1
docker login -u admin -p admin 3.129.25.247:2020
docker push 3.129.25.247:2020/redis:v1

```
[ec2-user@ip-172-31-24-221 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
tomcat              <none>              b4f6a90c69a4        3 days ago          647MB
nginx               latest              8cf1bfb43ff5        5 days ago          132MB
redis               latest              50541622f4f1        5 days ago          104MB
hello-world         latest              bf756fb1ae65        6 months ago        13.3kB
[ec2-user@ip-172-31-24-221 ~]$ docker tag redis 3.129.25.247:2020/redis:v1
[ec2-user@ip-172-31-24-221 ~]$ docker push 3.129.25.247:2020/redis:v1
The push refers to repository [3.129.25.247:2020/redis]
ed8f8f354de8: Preparing
e7b10f84d45b: Preparing
ce89ae6e6358: Preparing
03f8dae99b4d: Preparing
ad9080bec957: Preparing
95ef25a32043: Preparing
no basic auth credentials
[ec2-user@ip-172-31-24-221 ~]$ docker login -u admin -p admin 3.129.25.247:2020
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-24-221 ~]$ docker push 3.129.25.247:2020/redis:v1
The push refers to repository [3.129.25.247:2020/redis]
ed8f8f354de8: Pushed
e7b10f84d45b: Pushed
ce89ae6e6358: Pushed
03f8dae99b4d: Pushed
ad9080bec957: Pushed
95ef25a32043: Pushed
v1: digest: sha256:7d1ebef417593273b2f7b137f615393b21723d93568868c7fe5aad0401469152 size: 1572
[ec2-user@ip-172-31-24-221 ~]$
```

4. Verify the nexus server



Docker Logs

To troubleshoot or debug we need the logs of docker container. docker logs command is used to print the logs of any container.

find Docker Container Logs file Location

docker inspect --format='{{.LogPath}}' container_id

```
[ec2-user@ip-172-31-24-221 ~]$ docker inspect --format='{{.LogPath}}' 7e3123320183
/var/lib/docker/containers/7e3123320183ad20b1b127c738b2ca482c13283e9a50184a39d4369fb747028c/7e3123320183ad20b1b127c738b2ca482c1
3283e9a50184a39d4369fb747028c-json.log
[ec2-user@ip-172-31-24-221 ~]$
```

```
[ec2-user@ip-172-31-24-221 ~]$ sudo cat /var/lib/docker/containers/7e3123320183ad20b1b127c738b2ca482c13283e9a50184a39d4369fb747
028c/7e3123320183ad20b1b127c738b2ca482c13283e9a50184a39d4369fb747028c-json.log
{"log":"/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration\r\n","stream":"stdout"
,"time":"2020-07-27T03:47:21.818413792Z"}
{"log":"/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/\r\n","stream":"stdout","time":"2020-07-27T03:
47:21.818727703Z"}
{"log":"/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh\r\n","stream":"stdout","time":"20
20-07-27T03:47:21.823087448Z"}
{"log":"10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf\r\n","stream":"stdout","time":"
2020-07-27T03:47:21.829981347Z"}
{"log":"10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf\r\n","stream":"stdout","time"
:"2020-07-27T03:47:21.838747918Z"}
{"log":"/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh\r\n","stream":"stdout","time":"2020-0
7-27T03:47:21.839101579Z"}
{"log":"/docker-entrypoint.sh: Configuration complete; ready for start up\r\n","stream":"stdout","time":"2020-07-27T03:47:21.84
1814705Z"}
{"log":"27.59.137.112 - - [27/Jul/2020:03:48:01 +0000] \"GET / HTTP/1.1\" 200 612 \"-\" \"Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36\" \"-\"\r\n","stream":"stdout","time":"2020-07-27
T03:48:01.948935907Z"}
{"log":"2020/07/27 03:48:05 [error] 28#28: *1 open() \"/usr/share/nginx/html/favicon.ico\" failed (2: No such file or directory
), client: 27.59.137.112, server: localhost, request: \"GET /favicon.ico HTTP/1.1\", host: \"3.129.65.69\", referrer: \"http://
3.129.65.69/\"\r\n","stream":"stdout","time":"2020-07-27T03:48:05.05870787Z"}
{"log":"27.59.137.112 - - [27/Jul/2020:03:48:05 +0000] \"GET /favicon.ico HTTP/1.1\" 404 555 \"http://3.129.65.69/\" \"Mozilla/
5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36\" \"-\"\r\n","stream
":"stdout","time":"2020-07-27T03:48:05.058731584Z"}
{"log":"209.17.96.122 - - [27/Jul/2020:03:50:17 +0000] \"GET / HTTP/1.1\" 200 612 \"-\" \"Mozilla/5.0 (compatible; Nimbostratus
-Bot/v1.3.2; http://cloudsystemnetworks.com)\" \"-\"\r\n","stream":"stdout","time":"2020-07-27T03:50:17.166802686Z"}
[ec2-user@ip-172-31-24-221 ~]$
```

find the logs of docker container

docker logs container_id

```
[ec2-user@ip-172-31-24-221 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               N
AMES
7e3123320183        nginx:latest        "/docker-entrypoint.…"   9 minutes ago       Up 9 minutes        0.0.0.0:80->80/tcp   m
y-nginx
[ec2-user@ip-172-31-24-221 ~]$ docker logs 7e3123320183
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
27.59.137.112 - - [27/Jul/2020:03:48:01 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
2020/07/27 03:48:05 [error] 28#28: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client:
 27.59.137.112, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "3.129.65.69", referrer: "http://3.129.65.69/"
27.59.137.112 - - [27/Jul/2020:03:48:05 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://3.129.65.69/" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
209.17.96.122 - - [27/Jul/2020:03:50:17 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (compatible; Nimbostratus-Bot/v1.3.2;
http://cloudsystemnetworks.com)" "-"
[ec2-user@ip-172-31-24-221 ~]$
```

docker container logs with timestamp

docker logs container_id --timestamps

```
[ec2-user@ip-172-31-24-221 ~]$ docker logs 7e3123320183 --timestamps
2020-07-27T03:47:21.818413792Z /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
2020-07-27T03:47:21.818727703Z /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
2020-07-27T03:47:21.823087448Z /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
2020-07-27T03:47:21.829981347Z 10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
2020-07-27T03:47:21.838747918Z 10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
2020-07-27T03:47:21.839101579Z /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
2020-07-27T03:47:21.841814705Z /docker-entrypoint.sh: Configuration complete; ready for start up
2020-07-27T03:48:01.948935907Z 27.59.137.112 - - [27/Jul/2020:03:48:01 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Window
s NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
2020-07-27T03:48:05.058707870Z 2020/07/27 03:48:05 [error] 28#28: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No s
uch file or directory), client: 27.59.137.112, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "3.129.65.69", re
ferrer: "http://3.129.65.69/"
2020-07-27T03:48:05.058731584Z 27.59.137.112 - - [27/Jul/2020:03:48:05 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://3.129
.65.69/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "
-"
2020-07-27T03:50:17.166802686Z 209.17.96.122 - - [27/Jul/2020:03:50:17 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (compat
ible; Nimbostratus-Bot/v1.3.2; http://cloudsystemnetworks.com)" "-"
[ec2-user@ip-172-31-24-221 ~]$
```

find docker container logs since particular date

docker logs container_id --since YYYY-MM-DD

```
[ec2-user@ip-172-31-24-221 ~]$ docker logs 7e3123320183 --since 2020-07-27
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
27.59.137.112 - - [27/Jul/2020:03:48:01 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
2020/07/27 03:48:05 [error] 28#28: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client:
 27.59.137.112, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "3.129.65.69", referrer: "http://3.129.65.69/"
27.59.137.112 - - [27/Jul/2020:03:48:05 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://3.129.65.69/" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
209.17.96.122 - - [27/Jul/2020:03:50:17 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (compatible; Nimbostratus-Bot/v1.3.2;
http://cloudsystemnetworks.com)" "-"
```

find docker container logs since particular date

docker logs container_id --since YYYY-MM-DDTHH:MM

```
[ec2-user@ip-172-31-24-221 ~]$ docker logs 7e3123320183 --since 2020-07-27T01:00
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
27.59.137.112 - - [27/Jul/2020:03:48:01 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
2020/07/27 03:48:05 [error] 28#28: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client:
 27.59.137.112, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "3.129.65.69", referrer: "http://3.129.65.69/"
27.59.137.112 - - [27/Jul/2020:03:48:05 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://3.129.65.69/" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
209.17.96.122 - - [27/Jul/2020:03:50:17 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (compatible; Nimbostratus-Bot/v1.3.2;
http://cloudsystemnetworks.com)" "-"
[ec2-user@ip-172-31-24-221 ~]$
```

Tail the last N lines of logs

docker logs container_id --tail N

```
[ec2-user@ip-172-31-24-221 ~]$ docker logs 7e3123320183 --tail 10
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
27.59.137.112 - - [27/Jul/2020:03:48:01 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
2020/07/27 03:48:05 [error] 28#28: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client:
 27.59.137.112, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "3.129.65.69", referrer: "http://3.129.65.69/"
27.59.137.112 - - [27/Jul/2020:03:48:05 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://3.129.65.69/" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
209.17.96.122 - - [27/Jul/2020:03:50:17 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (compatible; Nimbostratus-Bot/v1.3.2;
http://cloudsystemnetworks.com)" "-"
[ec2-user@ip-172-31-24-221 ~]$
```

To see live logs

docker logs container_id --follow

```
[ec2-user@ip-172-31-24-221 ~]$ docker logs 7e3123320183 --follow
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
27.59.137.112 - - [27/Jul/2020:03:48:01 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
2020/07/27 03:48:05 [error] 28#28: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client:
 27.59.137.112, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "3.129.65.69", referrer: "http://3.129.65.69/"
27.59.137.112 - - [27/Jul/2020:03:48:05 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://3.129.65.69/" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
209.17.96.122 - - [27/Jul/2020:03:50:17 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (compatible; Nimbostratus-Bot/v1.3.2;
http://cloudsystemnetworks.com)" "-"
```

..... Prepared by PPNREDDY

docker logs to stdout

docker logs container > /tmp/stdout.log

```
[ec2-user@ip-172-31-24-221 ~]$ docker logs 7e3123320183 > 7e3123320183.log
[ec2-user@ip-172-31-24-221 ~]$
```

```
[ec2-user@ip-172-31-24-221 ~]$ docker logs 7e3123320183 > 7e3123320183.log
[ec2-user@ip-172-31-24-221 ~]$ cat 7e3123320183.log
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
27.59.137.112 - - [27/Jul/2020:03:48:01 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
2020/07/27 03:48:05 [error] 28#28: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client:
 27.59.137.112, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "3.129.65.69", referrer: "http://3.129.65.69/"
27.59.137.112 - - [27/Jul/2020:03:48:05 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://3.129.65.69/" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
209.17.96.122 - - [27/Jul/2020:03:50:17 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (compatible; Nimbostratus-Bot/v1.3.2;
http://cloudsystemnetworks.com)" "-"
27.59.137.112 - - [27/Jul/2020:04:07:00 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKi
t/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
27.59.137.112 - - [27/Jul/2020:04:07:11 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKi
t/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36" "-"
[ec2-user@ip-172-31-24-221 ~]$
```

..... Prepared by PPNREDDY