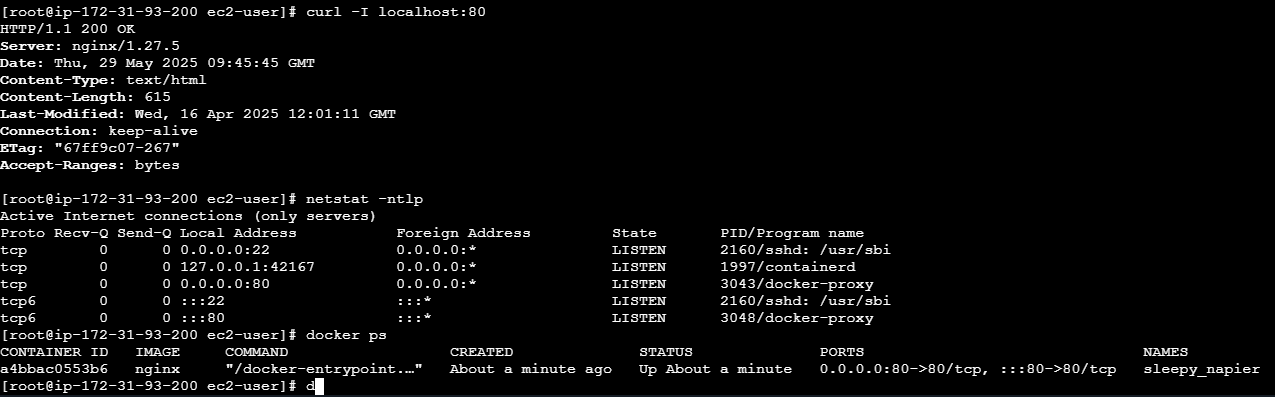
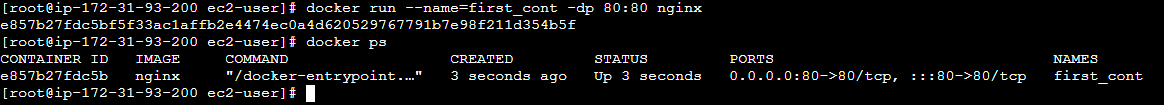
Docker Image is a file which contains all the necessary dependency and configurations which are required to run an application.

Docker Containers is basically a running instance an image.



Container identification:



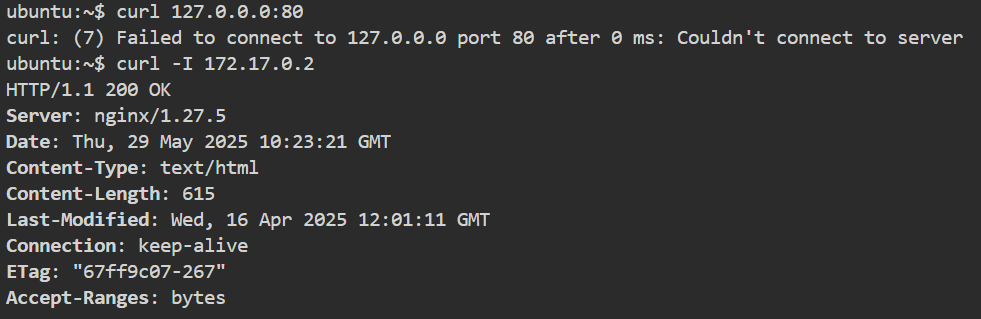
Uuid, by default name and custom container name by –name flag

Port binding:

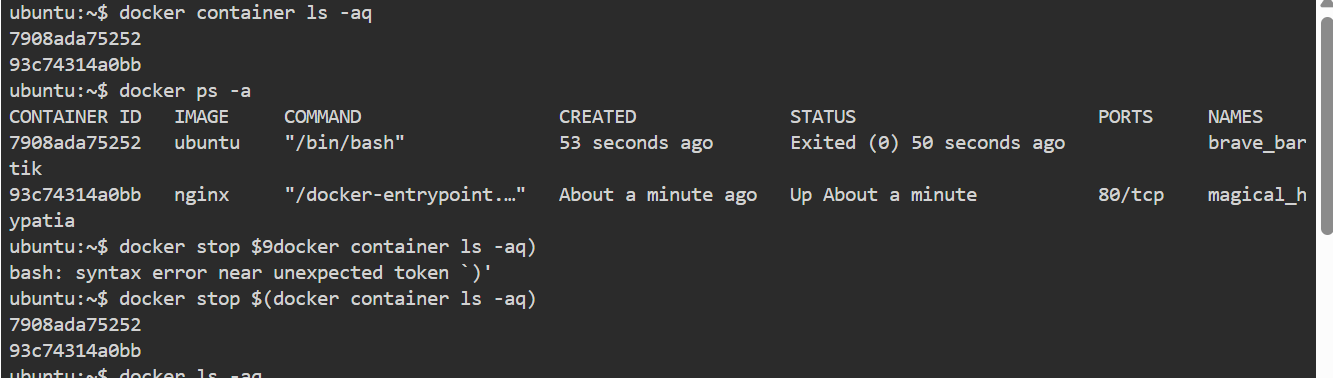
By default Docker containers can make connections to the outside world, but the outside world cannot connect to containers.

If we want to containers to accept incoming connection from the world, you will have to bind it to a host port.

Host ip and container ip are different so we are mapping the ports and accessing container with host ip

Without port mapping externally not connecting but internally with container ip it is connecting  


-d – detached mode to run container and associated logs in background

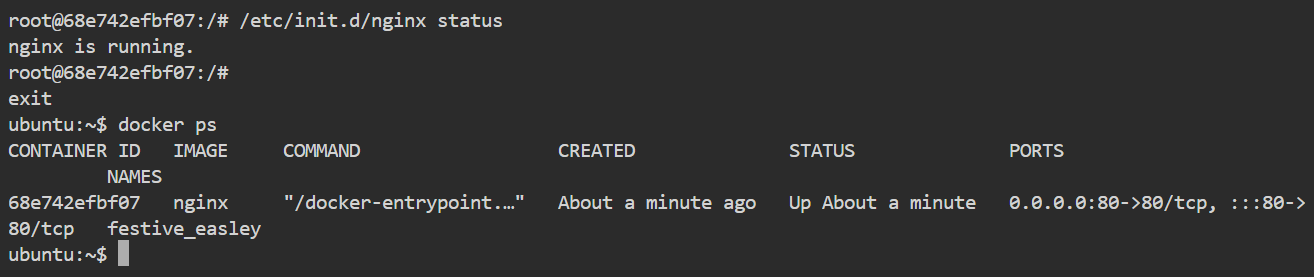


Exec- is to enter inside container and run commands.

-it flag: -I is interactive mode to provide an stdin to container and -t is tty to provide a terminal

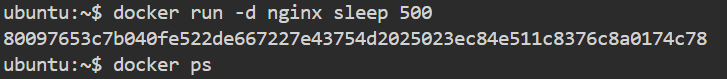
Default container command:

By default there would be command defined when starting container, which process should be started or not.



Nginx is running while we starting container by default.

Overriding default command:



Normally the default command will replace

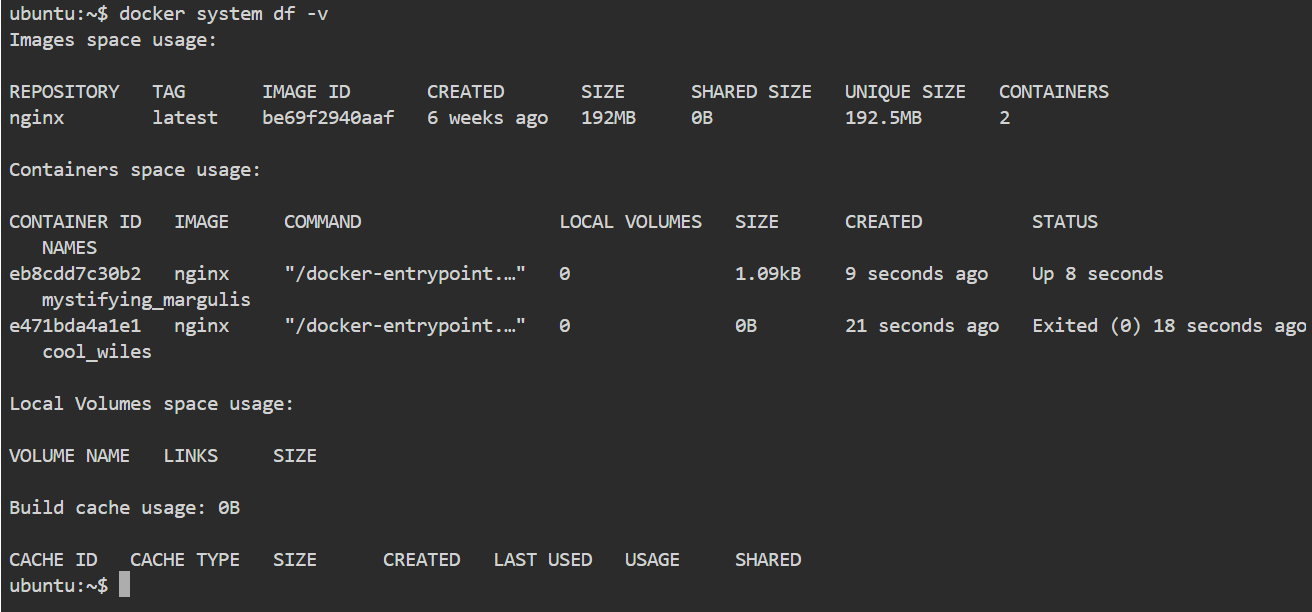
Since this is primary process PID 1 after the command process is done container will get into exited state.

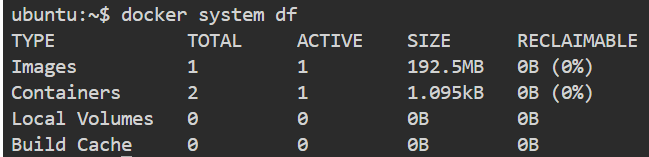
Docker restarts policy:

By default, Docker containers will not start when they exit or when docker daemon is restarted.

--restart flag used.

Docker system df -v : container wise volume usage





--rm flag : to auto remove container after exiting container.

**Section 2: Images**

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

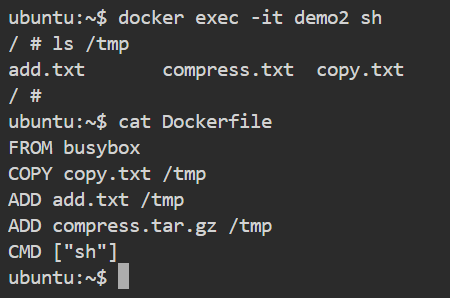
ADD vs COPY

COPY and ADD are both Dockerfile instructions that serve similar purposes. They let you copy files from a specific location into a Docker image.

ADD lets you do that too, but it also supports 2 other sources.

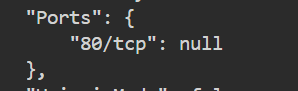
● First, you can use a URL instead of a local file / directory.

● Secondly, you can extract a tar file from the source directly into the destination



Expose:

Inform the application port listening not publishing. For publishing we should use -p. Exposing is mainly for understanding.



Health check:

he health check will first run interval seconds after the container is started, and then again interval seconds after each previous check completes.

1. docker run -dt --name tmp --health-cmd "curl --fail http://localhost" --health-interval=5s busybox sh
3. docker run -dt --name tmp2 --health-cmd "curl -f http://localhost" --health-interval=5s --health-retries=1 busybox sh

ENTRYPOINT:

Set main command, cannot override the command. We can append but cannot overrid

Workdir:

The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile

ENV:

The ENV instruction sets the environment variable to the value .

ENV NGINX 1.2

RUN curl -SL <http://example.com/web-$NGINX.tar.xz>

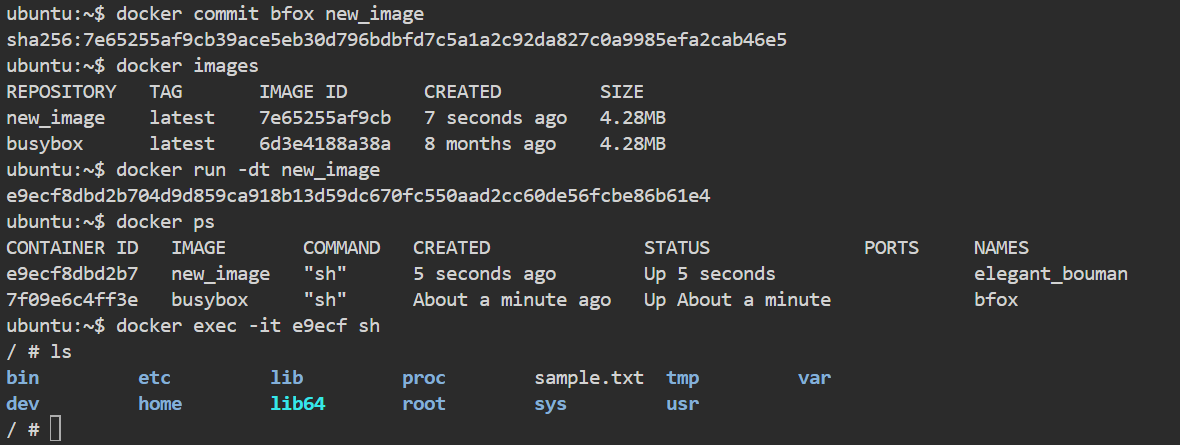
RUN tar -xzvf web-$NGINX.tar.xz



Tags: to represent images in different version

Docker commit: creating a image from a running container.

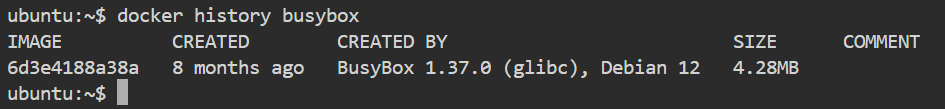
Container created and created a file inside the container. Now we are creating a image from that



--change flag can be used to change some parameters from base image like cmd env etc..

Layesr of image:

Refer ppt



Through this command we can identify the layers

Image inspect: docker inspect <image name> command

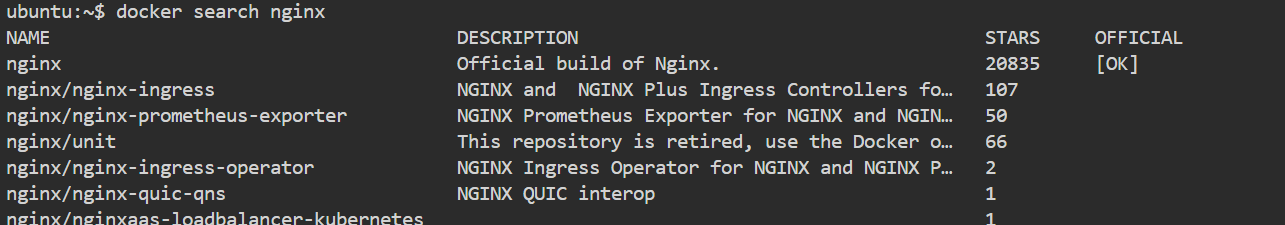
Image prune: help to clean unused images ( images not with containers and tags associated with him)

* prune can work with n/w volume container etc..

modifying image to single layer by export into tar and then import to docker image.

Docker registries: please check ppt

Filtering images in registry



Sending images between hosts:

The docker save command will save one or more images to a tar archive

Example Snippet: docker save busybox > busybox.tar

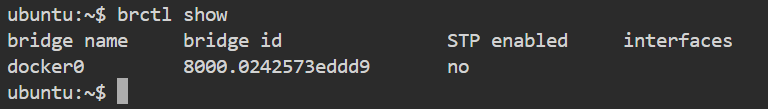
Now to back from tar

The docker load command will load an image from a tar archive

Example Snippet: docker load < busybox.tar

**Networking**

A bridge network uses a software bridge which allows containers connected to the same bridge network to communicate, while providing isolation from containers which are not connected to that bridge network.



This driver removes the network isolation between the docker host and the docker containers to use the host’s networking directly. For instance, if you run a container which binds to port 80 and you use host networking, the container’s application will be available on port 80 on the host’s IP address.

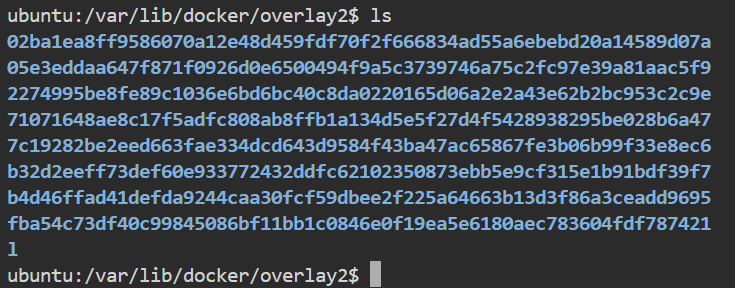
If you want to completely disable the networking stack on a container, you can use the none network.

**Volume**

A Docker image is built up from a series of layers.

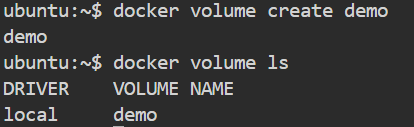
Storage Driver piece all these things together for you

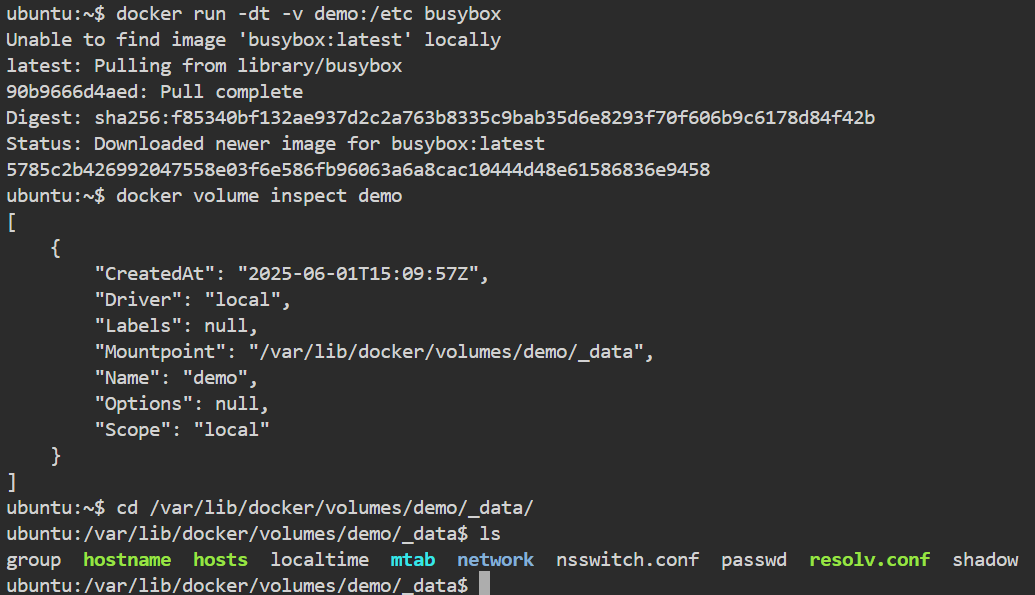
/etc/docker will have the storage driver details all image layers details inside the storage driver folder



Bind mount and docker volume

Create a docker volume area in host





/etc of container mounted to that specific path in host

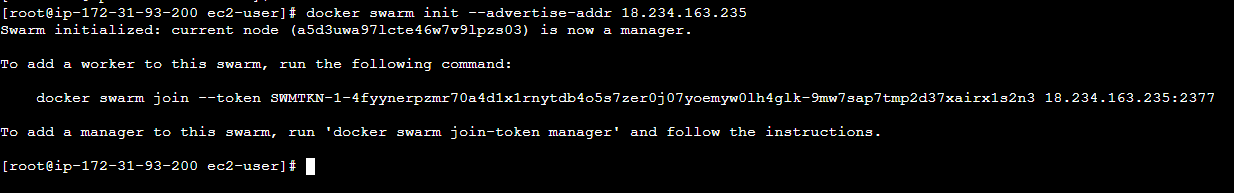
Bind mounts are using new or existing directory in host while docker volume are specifically from docker volume folder

-v /hostpath:/container path

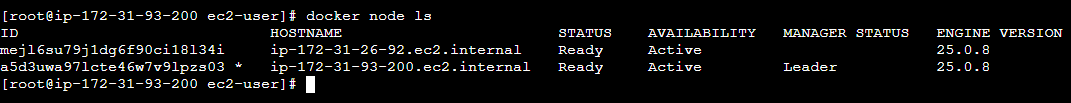
**Orchestration:**

Managing container lifecycle

Docker swarm: a manager node and worker nodes are there

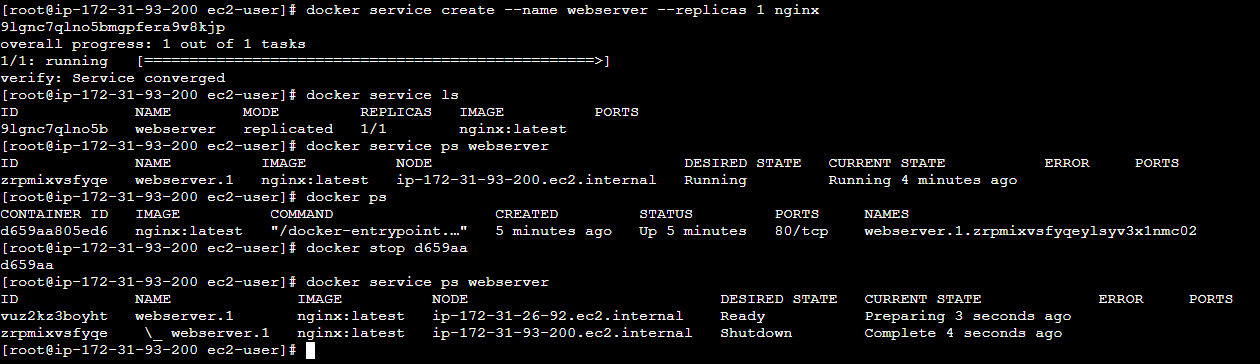


Join worker node with join command

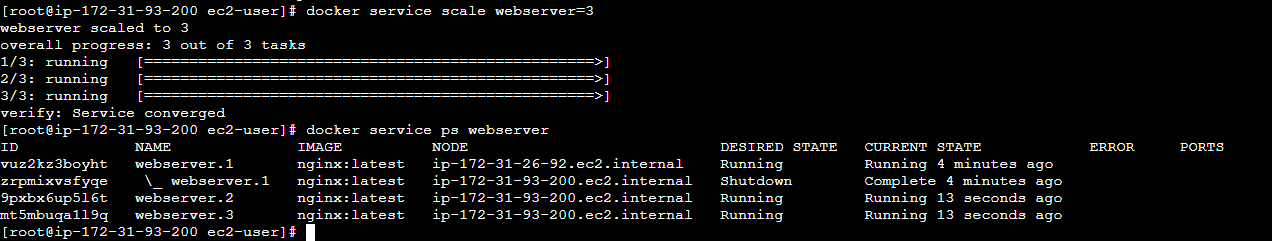


Services:

A service is the definition of the tasks to execute on the manager or worker nodes. docker service create --name webserver --replicas 1 nginx



Containers running in a service are called “tasks.”



A global service is a service that runs one task on every node.like daemon set

Drain node: docker node update --availability drain swarm03

Publishing ports:

docker service create --name mywebserver --replicas 1 --publish 8080:80 nginx

Docker compose:

Compose is a tool for defining and running multi-container Docker applications.

1. version: '3.3'
3. services:
4. db:
5. image: mysql:5.7
6. volumes:
7. - db\_data:/var/lib/mysql
8. restart: always
9. environment:
10. MYSQL\_ROOT\_PASSWORD: somewordpress
11. MYSQL\_DATABASE: wordpress
12. MYSQL\_USER: wordpress
13. MYSQL\_PASSWORD: wordpress
15. wordpress:
16. depends\_on:
17. - db
18. image: wordpress:latest
19. ports:
20. - "8000:80"
21. restart: always
22. environment:
23. WORDPRESS\_DB\_HOST: db:3306
24. WORDPRESS\_DB\_USER: wordpress
25. WORDPRESS\_DB\_PASSWORD: wordpress
26. WORDPRESS\_DB\_NAME: wordpress
27. volumes:
28. db\_data: {}

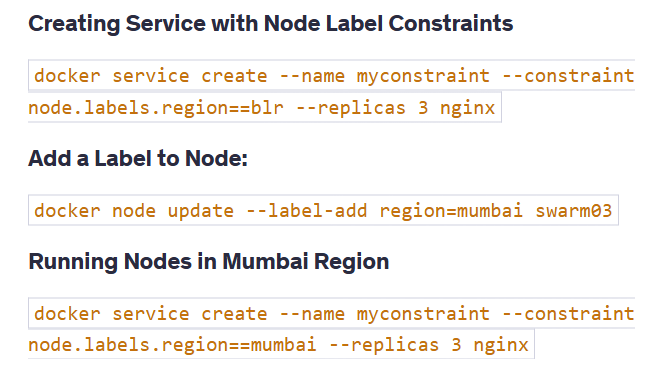
docker compose up -d

docker stack: to deploy in swarm mode compose wont allow to orchestrate ie swarm mode

swarm mount: docker service create --name myservice --mount type=volume,source=myvolume,target=/mypath nginx

Placement constraints

Swarm services provide a few different ways for you to control scale and placement of services on different nodes.



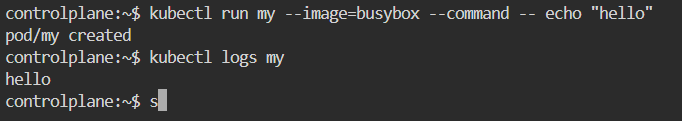
In swarm overlay network is used to communicate between containers in nodes

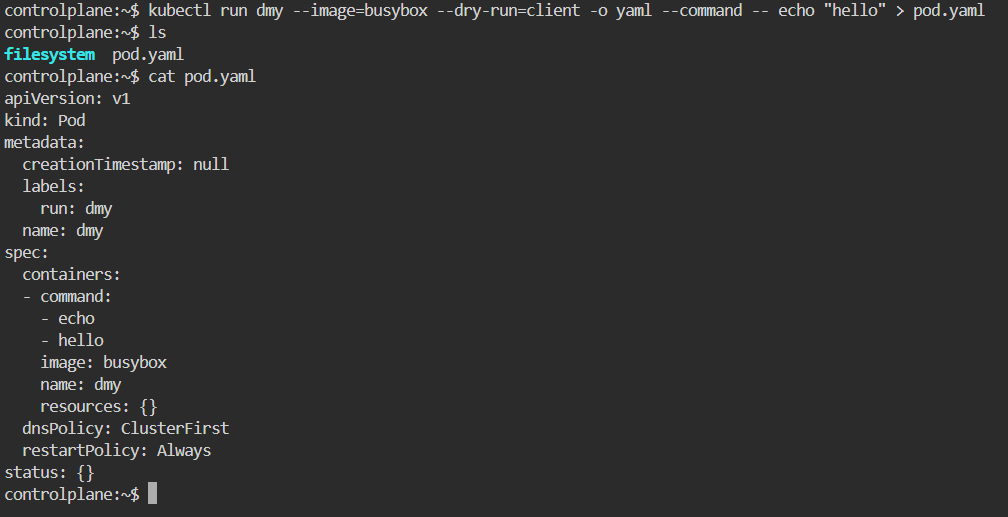
docker service create --name myoverlay --network mynetwork --replicas 3 nginx

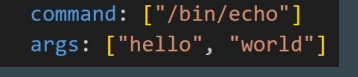
**Kubernetes**

Command and argument:

If we define cmd and arg in pod definition it will override the default/defined in dockerfile of the image used.







2 ways of representing like above  
there are two primary ways to specify them:

● Array (JSON array notation, square brackets []):

● Multi-Line YAML List (- for each new item)

The separation of command and args in Kubernetes is a design choice that provides flexibility and clarity when working with containerized applications.

This means it is not required to define as cmd and arg single definition like above also will work

Ports in manifest is same as expose we are providing info on ports

Node affinity: Node Selector primarily makes use of a method of strict equality, Node Affinity is similar to the older Node Selector, but it is more flexible and expressive.

Eg: run pod in any node except small one

Request: The minimum amount of CPU or memory a container is guaranteed to get.

Limit: The maximum amount of CPU or memory a container is allowed to use.

Priority class: By defining priority class scheduler can schedule In order, and in case high priority pod there scheduler can evict low priority pod if no space in node

Service:

Pods will be creating with a private ip in nodes and it used to communicate within node

Endpoints contain the address of underlying Pods to which the service will route the traffic to.

Cluster IP: A Kubernetes Service of type ClusterIP provides an internal, stable IP address to expose your application only within the Kubernetes cluster.

Service acts as a gateway since it internal no external communication is not possible to endpoint

In manifest port means port associated with service so when we hit svcip:port it will hit target port defined

Node Port: A NodePort is one of the service types used to expose your application to the external world.

LB: This type creates an External Load Balancer in a Cloud Provider and routes the request received in Load Balancer to underlying NodePort.

An lb will be created automatically in respective cloud provider once we define this service and a default node port is created as underlying service to LB

External ip will be public ip of LB created

Ingress:

When we use a LoadBalancer Service Type, the Load balancer forwards traffic to a NodePort associated with a single service.

In an ideal approach, you want a single load balancer to handle requests for multiple services and a logic that can route traffic accordingly.

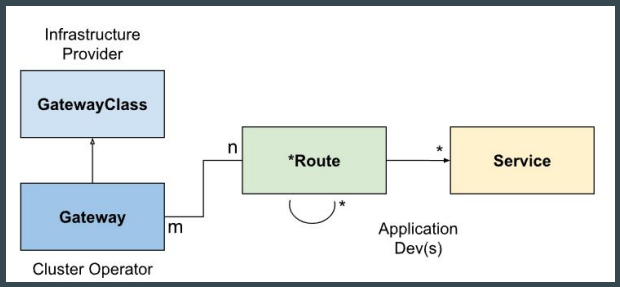
Service account: specifically for applications to perform some tasks in k8s cluster

HPA: Horizontal Pod AutoScaler (HPA) automatically adjusts the number of running pods in a deployment based on observed CPU utilization, memory usage, or custom metrics.

Gateway API:

Same as ingress concept, but support more protocols that http/https

Workflow:



GatewayClass specifies the controller used to implement the Gateway API resources

A Gateway describes an instance of traffic handling infrastructure.

The HTTPRoute kind specifies routing behavior of HTTP requests from a Gateway listener to backend network endpoints.(other than httproute multiple route are there)

CRD:

A Custom Resource Definition allows you to extend the Kubernetes API by defining your own custom resource types.

RBAC:

Role: set of permission subject: user, sa etc..

Rolebinding: linking roles and subject and its namespace scoped

Kubeconfig: refer ppt and github

In volume mounts we are defining the container path

Persistent Volume is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes.

PersistentVolumeClaim (PVC) is a request for storage by a user (e.g., a developer). It specifies size, access modes, and other requirements.