Dockerfile for maven

FROM openjdk

Run yum install maven

RUN echo “installed maven”

docker logs

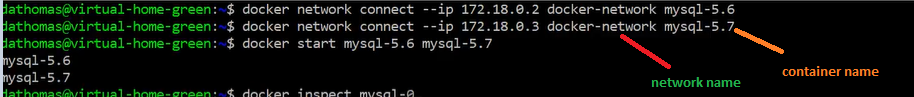
---------------------------------------------------------------Docker-------------------------------------------------------------------

**What is containerization?**

Containerization is a process of packaging an application code along with its required libraries,dependencies,frameworks and configuration files so that it can run efficiently and seamlessly in any environments.

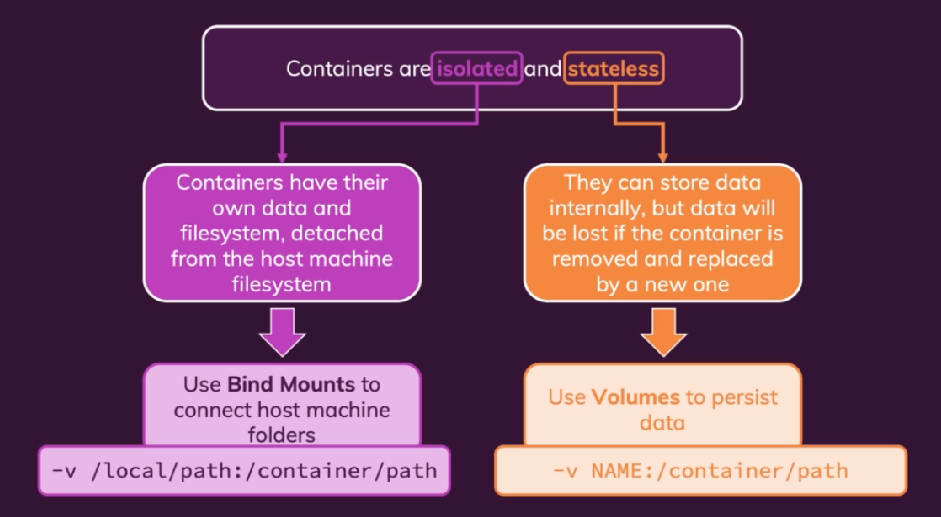


**How to assing IP address to containers?**

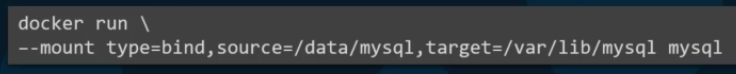


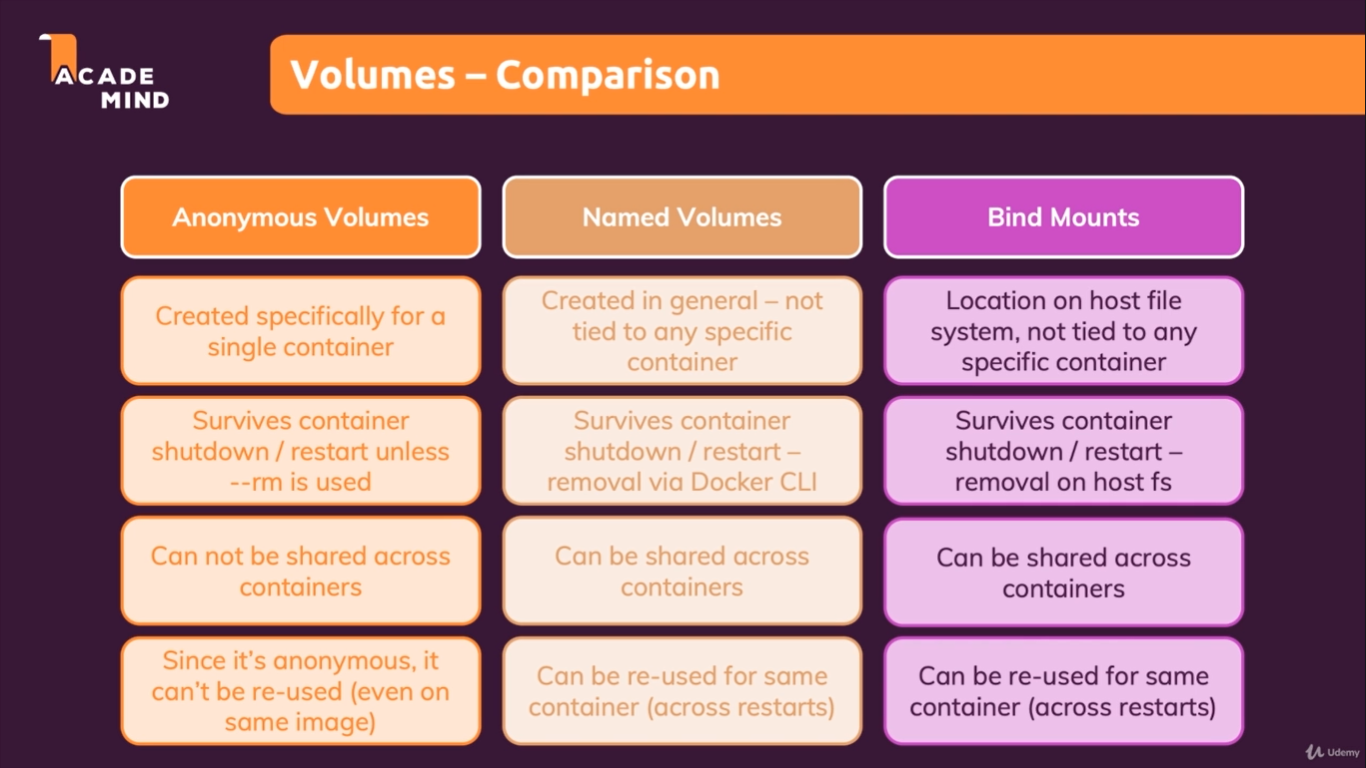
**Difference b/w volume mount and bind mount?**

* Volume mount mounts volume from the location (var/lib/docker/volumes), whereas bind mount mounts the directory from any location on the docker host.
* When you use a bind mount, a file or directory on the *host machine* is mounted into a container. The file or directory is referenced by its absolute path on the host machine.  By contrast, when you use a volume, a new directory is created within Docker’s storage directory on the host machine, and Docker manages that directory’s contents.
* Containers have their own data and filesystems,detached from the host machine filesystem,use bind mounts to connect host machine folders.When we start container docker stores the data in the anonymous volumes created under /var/lib/docker , concern with this volume is data under it will be vanished when the container removed. So to persist the data we need to attach Volume.
* If the attached volume is not present docker will create the volume and then it will mount that to specific path in-case of volume mount.
* Volumes are managed by docker whereas bind mounts are managed by us.
* If in case container removed or exited we can re-use(attach) the volume/mount to other containers ,so it can have the previous data.



Note :: And one important point to note here is using –v for volume is the old style , --mount is the latest way as it is more verbose.





**Diff b/w ARG and ENV?**

### What is ARG ?

* ARG instruction defines a variable that can be used to build a Docker image.
* ARG values are not available after the image is built.
* In the running container you can’t access the ARG variables.
* Once ARG variable is defined in the Dockerfile, you can always override the values by passing command line argument as shown below
* We can have multiple **ARG** instruction in the **Dockerfile**.
* **ARG** is the only instruction that can precede the FROM instruction in the **Dockerfile**.

docker build -t test **--build-arg VARIABLE\_1=7** **--build-arg VAR\_2=8 .**

#### Example of using ARG argument in the Dockerfile

FROM alpine:3.7  
ARG VARIABLE\_1=5  
RUN echo $VARIABLE\_1

#### Run the Dockerfile

docker build .

#### Run the Dockerfile by passing arguments

docker build **--build-arg VARIABLE\_1=7 .**

**Output:** We can see the value is override with the command line argument

### What is ENV?

* ENV is to provide default values for your future environment variables inside the container
* We can’t change the ENV variable using command line argument directly
* If we need to change the ENV variable using command line then we have to use ARG and place ARG variable in ENV variable
* In the below Dockerfile, we have created two variables , one as ARG and one as ENV
* Using ARG Value in ENV Variable

FROM alpine:3.7

ARG VARIABLE\_1=5

ENV VARIABLE\_2=$VARIABLE\_1

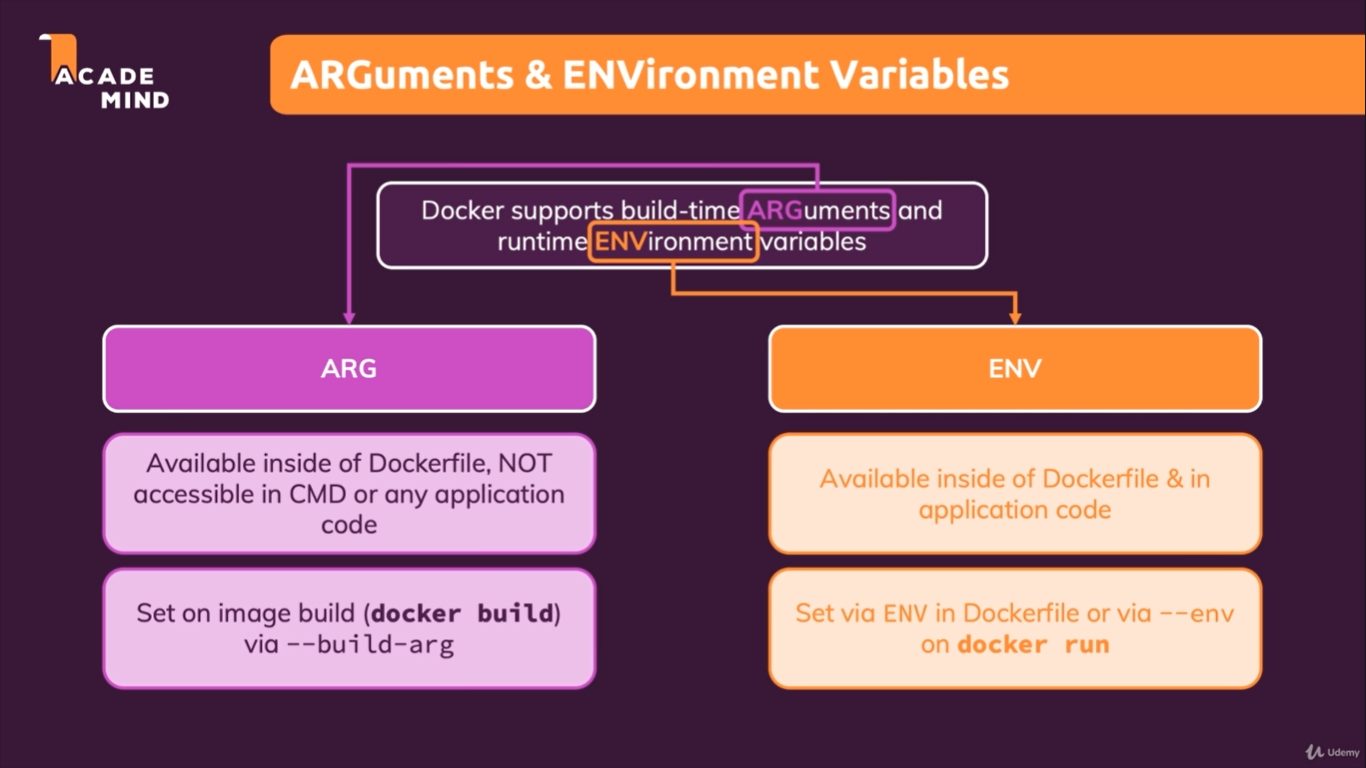
RUN echo "print variable value:" $VARIABLE\_1

RUN echo " print ENV variable : " $VARIABLE\_2

#### Build the docker image directly

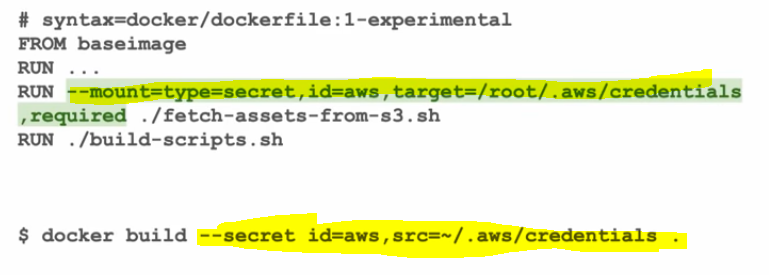
* both variable having same values as defined for ARG variable

docker build .



**how to deal with sensitive data in docker ?**

Secrets are used to store the username and password for the security purpose.



**how to give access to container/pod to access s3 bucket?---IRSA**

<https://www.bluematador.com/blog/iam-access-in-kubernetes-the-aws-security-problem>

**How to move/copy files from one container to another?**

docker cp [OPTIONS] CONTAINER:SRC\_PATH DEST\_PATH|-

docker cp [OPTIONS] SRC\_PATH|- CONTAINER:DEST\_PATH

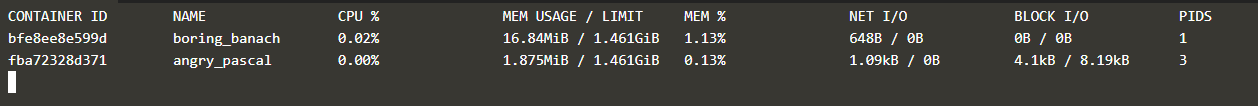
$ **docker cp . 7b7:/tmp** ----copy contents from Host to container

$**docker cp 7b7:/tmp/file2.txt .** -----To copy file2.txt from container to host

$**docker cp 7b7:/tmp/file2.txt 6br:/tmp .** ---Copy file2.txt to 6br container

**Use of docker stats command?**

It Display a live stream of container(s) resource usage statistics



**What is multistage builds?**

With multi-stage builds, you use multiple FROM statements in your Dockerfile. Each FROM instruction can use a different base, and each of them begins a new stage of the build. You can selectively copy artifacts from one stage to another, leaving behind everything you don't want in the final image.

FROM golang:1.7.3

WORKDIR /go/src/github.com/alexellis/href-counter/

RUN go get -d -v golang.org/x/net/html

COPY app.go .

RUN CGO\_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest

RUN apk --no-cache add ca-certificates

WORKDIR /root/

COPY --from=0 /go/src/github.com/alexellis/href-counter/app .

CMD ["./app"]

You only need the single Dockerfile. You don’t need a separate build script, either. Just run docker build.

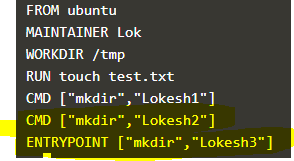
$ docker build -t alexellis2/href-counter:latest .

**What happens if i include below instructions in the dockerfile ?**

CMD [“mkdir”,” Lokesh1”]

CMD [“mkdir”, “Lokesh2”]

ENTRYPOINT [“mkdir”,” Lokesh3”]

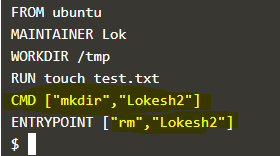


**Output:** the folders Lokesh3 ,mkdir and Lokesh2 will be created under /tmp

----------------------

CMD [“mkdir”,” Lokesh2”]

ENTRYPOINT [“rm” ,“Lokesh2”]



**Output:**

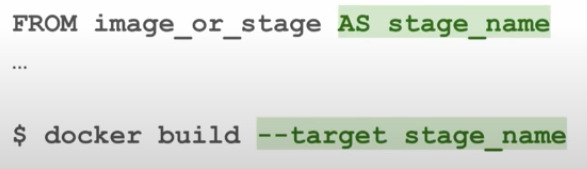
rm: cannot remove 'Lokesh2': No such file or directory

rm: cannot remove 'mkdir': No such file or directory

rm: cannot remove 'Lokesh2': No such file or directory

**What is the use of “—target” in multi-stage dockerfile?**

Using which we can build specific stages



**What is the use of “set” command?**

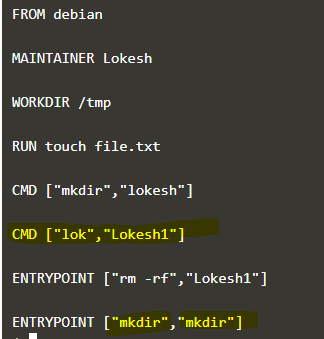
It is used to see all the environment variables.

**How will you get to know the container path to attach volume?—**WORKDIR

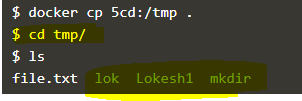
**Can we have multiple CMD and ENTRYPOINT instructions in dockerfile?**

Yes we can specify multiple ENTRYPOINT and CMD in a Dockerfile.

Note :: If a Dockerfile has multiple CMDs,ENTRYPOINT it only applies the instructions from the last one

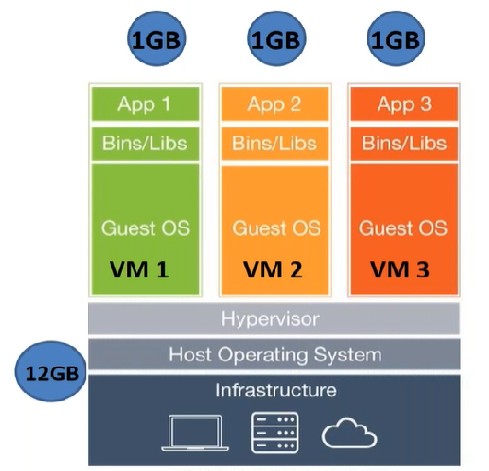


**Output::**



**What was the seen before Docker and what are the advantages of it?**

1. With Virtualization feature we can run multiple vm’s on top of physical hardware (Hypervisor), however each vm needs diff OS/necessary software’s and configurations .So it also leads to more costs and time wastage.



To run the same application in docker, all you need to do is deploy that application in Docker. There is no need for additional OS layer. You just deploy the application with its dependent libraries, docker engine (kernel, etc.) provides the rest..

1. **To make a replica of the application:**

In virtualization world what we would usually do, is:

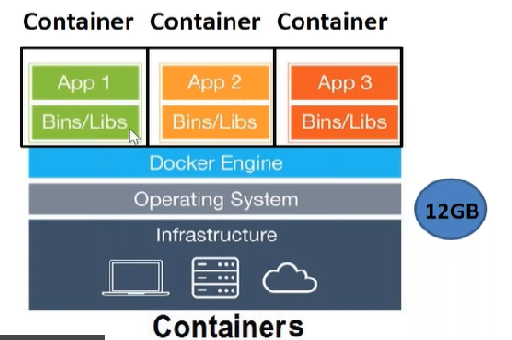
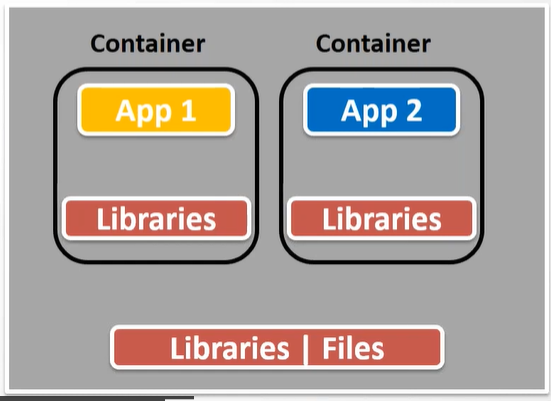
* Deploy OS in a VM running on vSphere.
* Deploy an application inside OS.
* Create a template.
* Redeploy the template in case of need. Time of redeployment around 5-10 minutes.
* Sounds great! Having app up and running in an hour and then being able to redeploy it in 5 minutes.

**But using docker all you need to do is build an image for the application and push it on to the hub and create a containers as many as you want.**

1. Compatibility issue b/w Developers and QA teams usually developers do unit testing and provides code/app to the QA for functional testing ,this app/code might or might not work properly due to /version of OS/library/dependancy so it may lead to dispute b/w developers and QA. Docker avoids this problem.

Using docker we first write a dockerfile with all the dependancies/libraries that are needed to run the applications and then will build an image out of it and we push that onto the repo ,and we can create a containers as many as we want depending on the requirements.

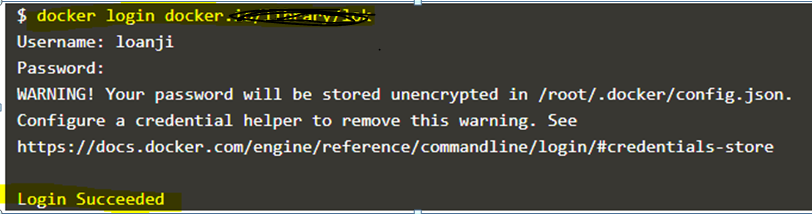
Containers look like a fully-fledged OS to the applications which are running on it ,and the containers are process that are created virtually on the Host OS.

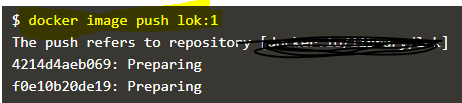


**How to push docker image to private repo?**

Login to the repo using(**docker login repo name)** the username and password

**Docker push** image name





**Dockerfile for java application?**

FROM openjdk:8-jre-alpine3.9

# copy the packaged jar file into our docker image

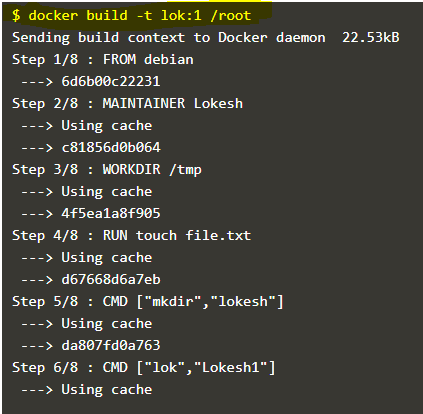
COPY target/demo-0.0.1-SNAPSHOT.jar /demo.jar

# set the startup command to execute the jar

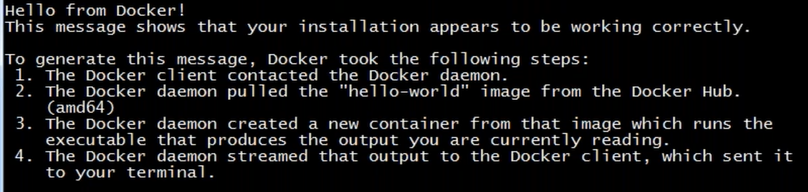
CMD ["java", "-jar", "/demo.jar"]

**What if I’m not giving(dot) . while building dockerimage?** (docker build –t tagname .)

If the dockerfile is present in present working directory we need to give “**.**” else need to provide the location of the file.



**What will happen if we pull the image for the first time?**



**What is “From scratch “ indicates in the Dockerfile?**

**FROM scrath**

Will get basic linux file system,if we provide scratch.

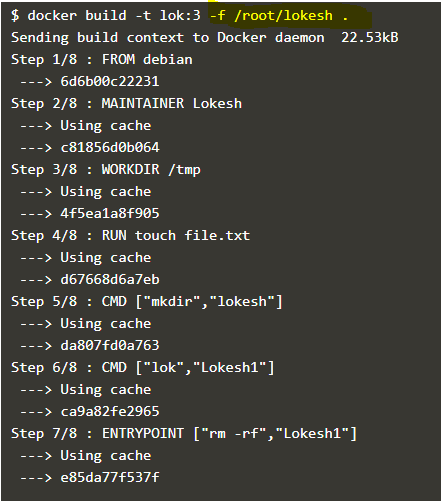
**What is “it” ?**

A pseudo terminal (also known as a **tty (tty is short of teletype**) connects a user's "terminal" with the stdin and stdout stream, commonly through a shell such as bash . ... In the case of docker, you'll often use **-t** and -**i** together when you run processes in **interactive mode**, such as when starting a bash shell

**Can we build an image with different filename (Other than “Dockerfile”)**

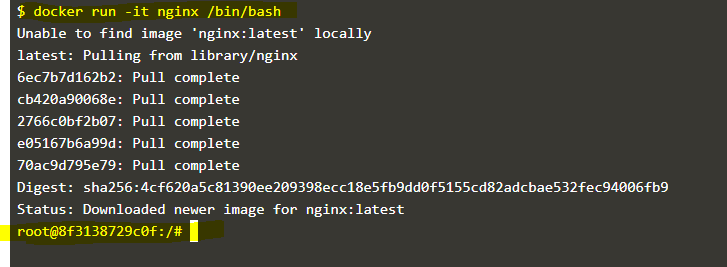
Yes we can have ,but while building u need to specify the path of the file by “-f”

Docker build -t tagname –f /path/filename .



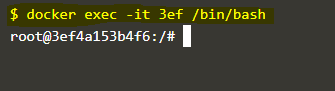
**Commands to enter into the container?**

**docker run –it nginx /bin/bash** -------------while creating container



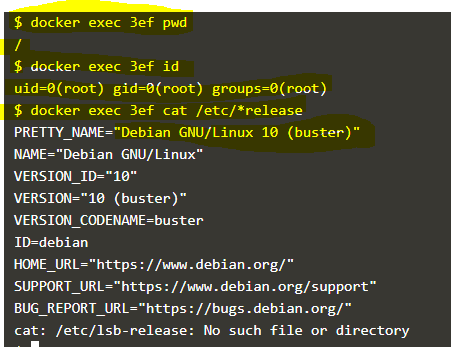
Or

**docker exec –it (container ID or container name) /bin/bash**------when container is already created



Or

**docker exec (container ID or container name) pwd ---** when container is already created and if you want to execute command without login to the container.



**How to exit from the docker container without killing it?**

ctrl pq

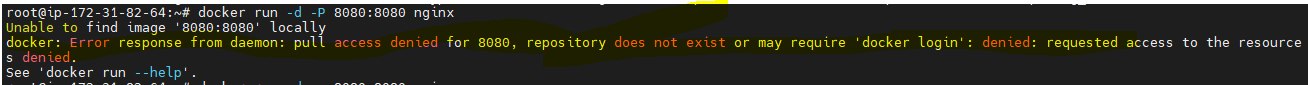
**What is “it” and “p” flags in Docker?**

It ---Interactive tty(terminal)

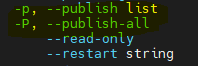
p---publish

**What will happen if I execute below command?**

docker run -d –P 8080:8080 nginx



It’ll give an error (Bcz “-P” is to publish all )



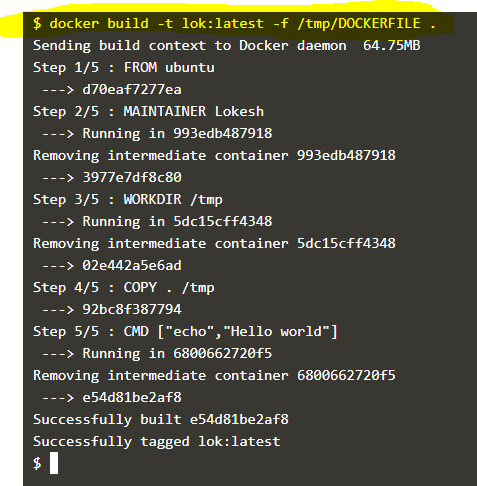
So below command works fine



**Lets say I want to keep my dockerfile name as “DOCKERFILE” and want to build the image,Can we do it?**

Yes we can build the image ,provided we mention –f and the location,name

**Note:: But it is recommended to use “Dockerfile” naming convention**

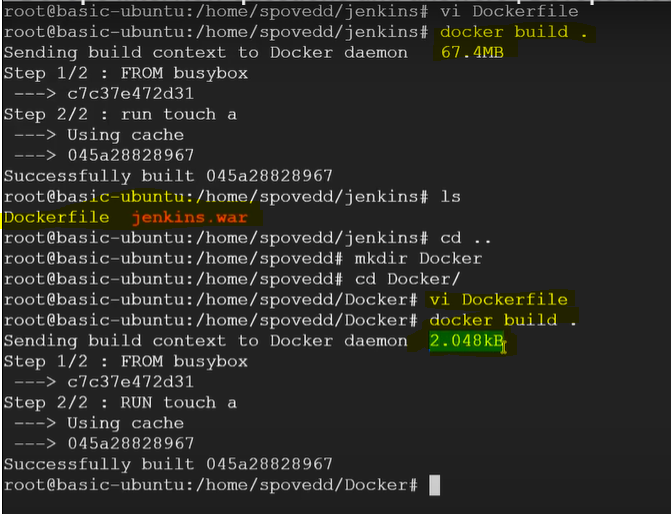


**What is meant by docker in docker ?**

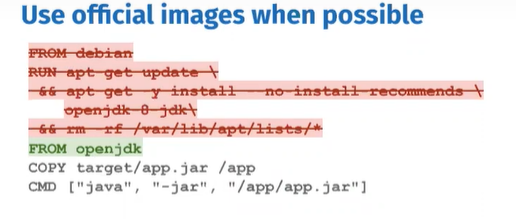
 If you run a Docker container that has itself Docker installed, can you then run Docker inside that Docker container (for example, to pull and build images, or to run other containers) with the two Docker instances being completely independent from each other.

https://itnext.io/docker-in-docker-521958d34efd

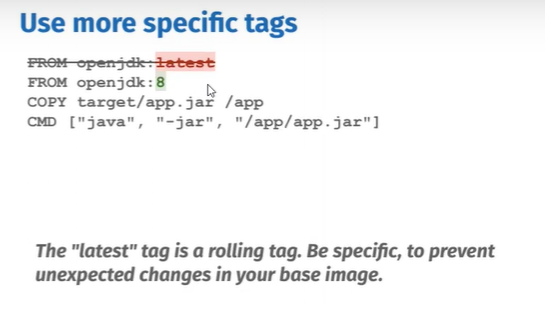
**Any 3 best practices while writing dockerfile?**

1) Always keep dockerfile in a empty directory. It saves time and data.

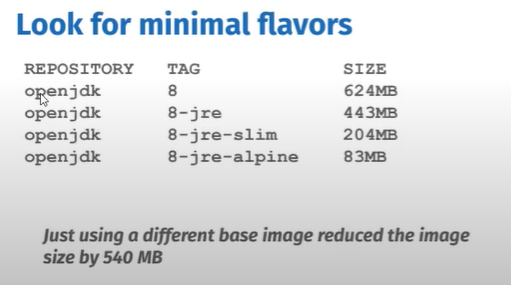
2. Use official images if possible, for security purpose.



3. Use more specific tags.



4) Look for minimal flavours



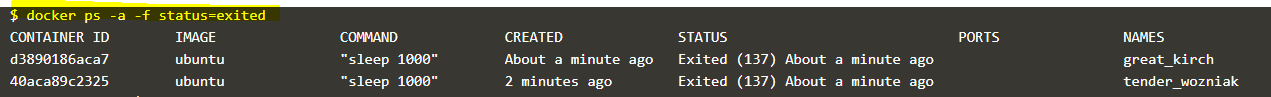
**Difference between docker stop and docker kill?**

**Docker stop** –It send s SIGTERM (Termination signal) and then the Kill signal(SIGKILL) after a grace period.

**Docker Kill**---It sends a kill signal (SIGKILL) directly

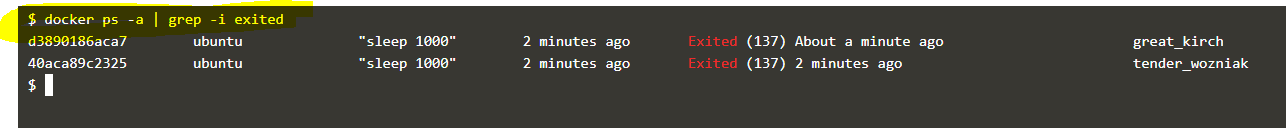
**Command to list containers which state is exited?**

docker ps –a –f status=exited (where -f is filter)



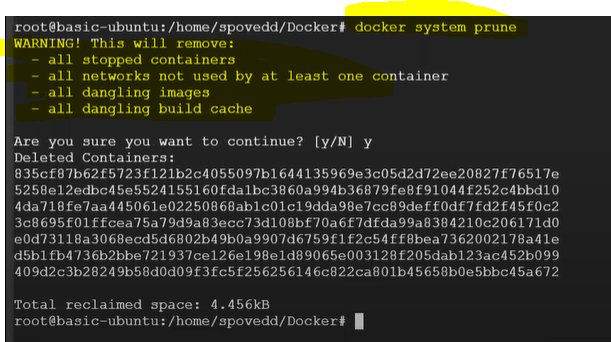
**Or**

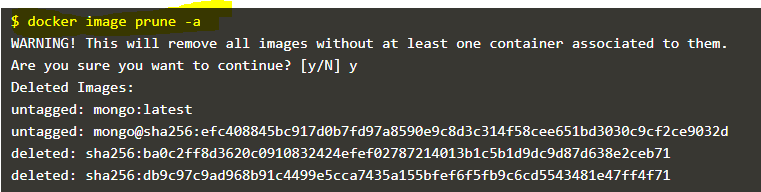
$docker ps –a | grep –I exited

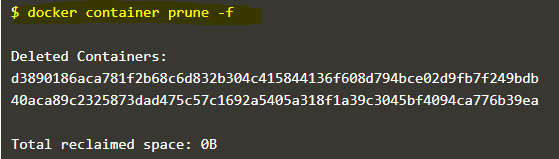


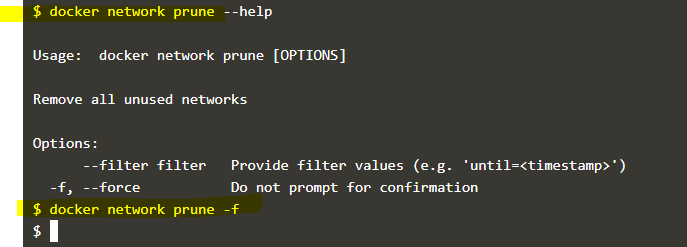
**Command to clean up docker host (deleting stopped containers, dangling images and unused networks,dangling build cache)? Or is there any method to reclaim storage in docker?**

Docker system prune









**What version of docker you have used? Specific reason to use that particular version?**

Version: 19.03.8

<https://docs.docker.com/release-notes/>

**Can we have multiple CMD in Dockerfile?**

There is no use of writing multiple CMD’s bcz last CMD will be taken into consideration.

**Have you worked on docker swarm and docker compose?**

**Docker swarm** is a container orchestration tool ,meaning that it allows the user to manage multiple containers deployed across multiple host machine.

**Docker Compose** :: Compose is a tool for defining and running multi container docker applications .

If there are dependent containers it will be difficult to start one by one and it’ll take time,By using docker compose up command you can run all the containers at a time.

**What are dangling images ?**

Dangling images are unused images, and that are not used in any container.

We can delete those using docker system prune command.

**Have you worked on multistage dockerfile and why we need that?**

FROM MavenasMaven

|  |
| --- |
|  |
|  | RUN mkdir /usr/src/mymaven |
|  | WORKDIR /usr/src/mymaven |
|  | COPY . . |
|  | RUN mvn install –DskipTests |
|  |  |
|  | FROM tomcat |
|  | WORKDIR webapps |
|  | COPY --from=maven /usr/src/mymaven/target/java-tomcat-maven-example.war . |
|  | RUN rm -rf ROOT && mv java-tomcat-maven-example.war ROOT.war |

We use multi-stage dockerfiles mainly to save storage (to reduce image size)

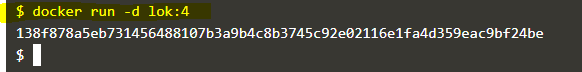
Note :: While the second FROM is executing docker will move previously executed steps to the cache.We can have multiple FROM statements in the dockerfile the last one will be taken into consideration for image size.

**Lets say I have a container in my machine, by login to that container I have installed many packages now I want to replicate the same in another machine ,how to achieve the same?**

Docker commit containerID –t tagname **--🡪 it’ll create an image**

Push the image to repo and using that image create a container





**Can you copy file from local to running container?**

Ans :: Yes,Using docker cp cmd



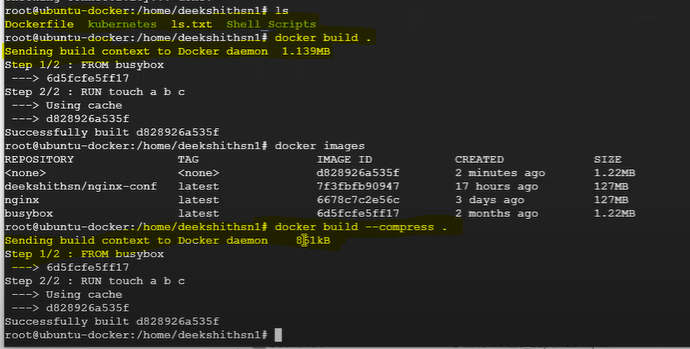
**What is the difference between ADD and COPY docker instructions in dockerfile?**

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

COPY takes in a *src* and *destination*. It lets you copy file or directory from your host into the Docker image itself.

ADD also do the same , 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.

**Lets say I have 1GB file that has to be sent to docker daemon,as its 1GB it will take much time and network too. By which option while building dockerfile we can send the file in better manner?**

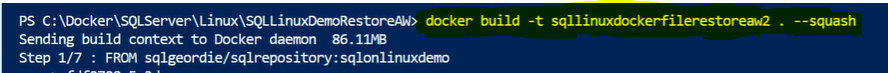


**Command to remove ,stopped and running containers?**

Ans ::: docker rm $(docker ps –a q)

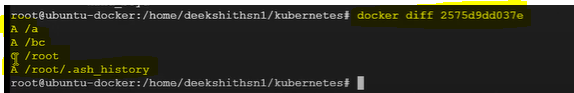
**What is docker squash?**

Docker-Squash Is a utility To Squash Multiple Docker Layers Into One To Create An Image.

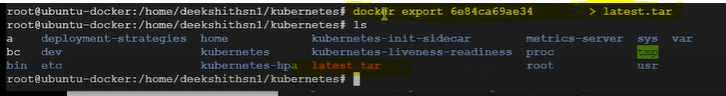


**Inside the container I did many changes like creating,modifying and deleting file but I wanted to check which file has been changed and what action has been taken what is the command we need to use?**

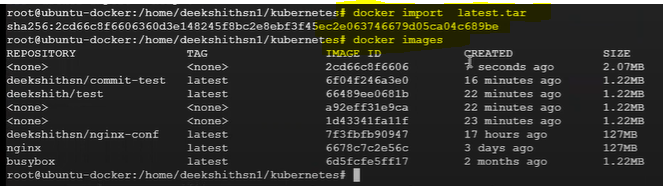
Ans :: Docker diff (It basically shows added, changed and deleted files)



**Can we export filesystem ,structure in docker ?**



**And to create a container out of it**



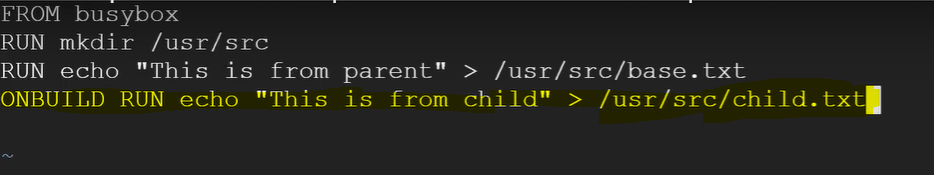
**What does ONBUILD instruction do in Dockerfile?**

It basically sets a trigger instruction that will not be executed when the image is built ,however it is going to be executed when a child image is built.

This is useful when you have two images ,the first one is the base that just install the packages and prepares the necessary environment ,while the second is based on it and provides the actual implementation of the application.

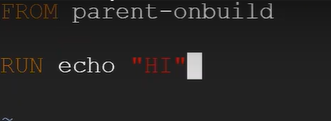
ONBUILD instruction accepts the following Dockerfile instructions as triggers:

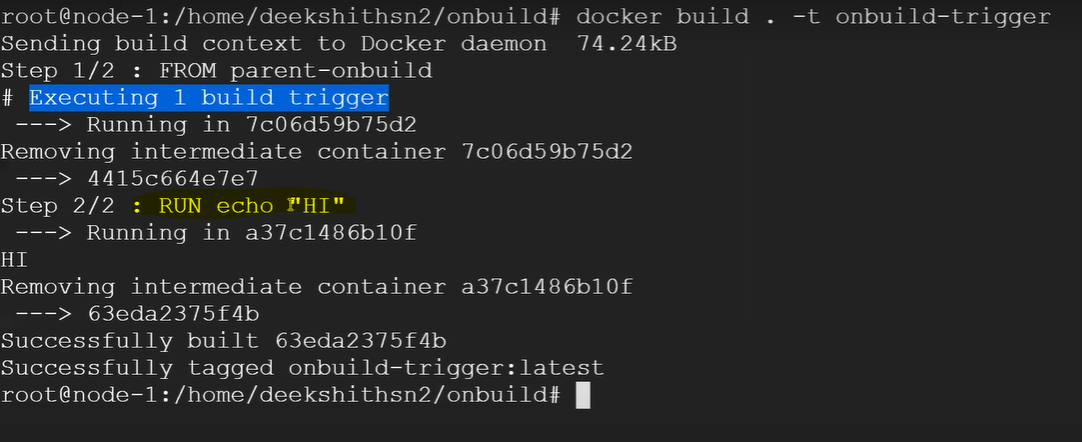
ADD,COPY,ENV,EXPOSE,LABEL,USER,WORKDIR,VOLUME,STOPSIGNAL ,,RUN,,ECHO

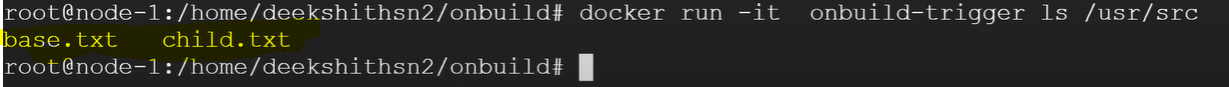


When you build an image (let’s say parent image) using this dockerfile,then u can only see base.txt under /usr/src/





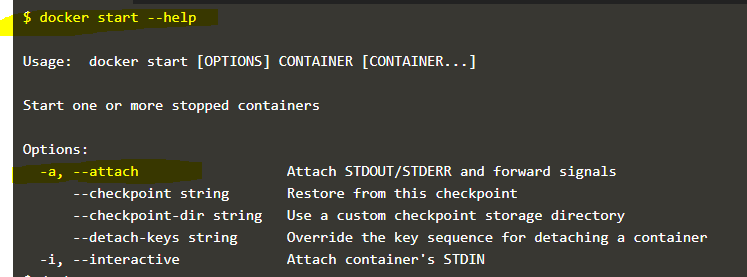


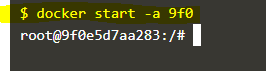


**What does “-a” indicates in the below command?**

Docker start –a container ID or name

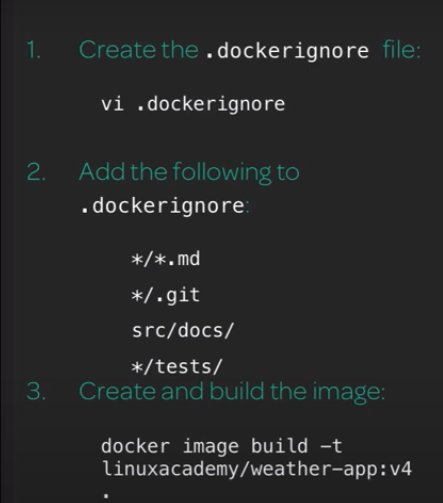
-a indicates that ,container will start the stopped container and it’ll attach the interactive terminal.





**What is the use of .Dockerignore file?**

The . dockerignore file allows you to exclude some files.so It helps to make build faster and lighter the image by excluding big files or repository that are not used wile build.



**I have a dockerfile that accepts arguments,if I supply value as “1” then it should use maven 2.x version for base image and if I supply “2” then it should take maven latest as base image.**

***ARG Instruction ---***ARG instruction defines a variable that can be passed at build time. Once it is defined in the Dockerfile you can pass with this flag --build-arg while building the image. We can have multiple ARG instruction in the Dockerfile. ARG is the only instruction that can precede the FROM instruction in the Dockerfile.

FROM centos:7

ARG arg

RUN if [ $1 -eq 1 ] ; then

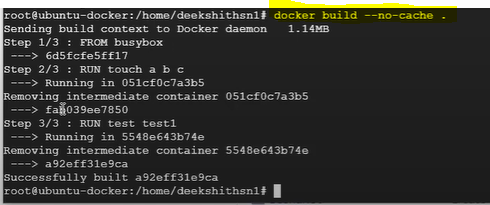
RUN echo use maven2.x;

RUN else echo use maven:latest

RUN fi

**What to do if I don’t want to use build cache while building images?**

If we mention –no-cache docker daemon will download the base image from the dockerhub instead of using the cache (To avoid cached insecure images)



**What if I don’t want to use the cached image ,is there any option to build image from docker hub always?**

Docker daemon (Always attempt to pull a newer version of the image)



**How to get outside files inside my container?**

**Use docker cp or bind mount**

Docker run –v /opt:/containerpath

**To delete all containers,images ,volumes etc at a shot?**

Ans :: Docker system prune

**write a docker file for to install apache,nginx,tomcat?**

FROM centos

RUN yum install httpd -y

COPY index.html /var/www/html/

CMD [“/usr/sbin/httpd”,” -D”,” FOREGROUND”]

EXPOSE 80

FROM ubuntu

MAINTAINER user@gmail.com

RUN apt-get update

RUN apt-get install –y nginx

CMD [“echo”,”Image created”]

FROM centos

MAINTAINER aksarav@middlewareinventory.com

RUN mkdir /opt/tomcat/

WORKDIR /opt/tomcat

RUN curl -O https://www-eu.apache.org/dist/tomcat/tomcat-8/v8.5.40/bin/apache-tomcat-8.5.40.tar.gz

RUN tar xvfz apache\*.tar.gz

RUN mv apache-tomcat-8.5.40/\* /opt/tomcat/.

RUN yum -y install java

RUN java -version

WORKDIR /opt/tomcat/webapps

RUN curl -O -L https://github.com/AKSarav/SampleWebApp/raw/master/dist/SampleWebApp.war

EXPOSE 8080

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

**How to merge two containers?** ---docker compose

**3. What is disadvantages of docker?**

**Containers don't run at bare-metal speeds.**

**2. How can connect the two containers? for example mangodb and apache servers i have .I need to communicate both containers?**

Keep both containers on the same n/w.

**Architecture of docker?**

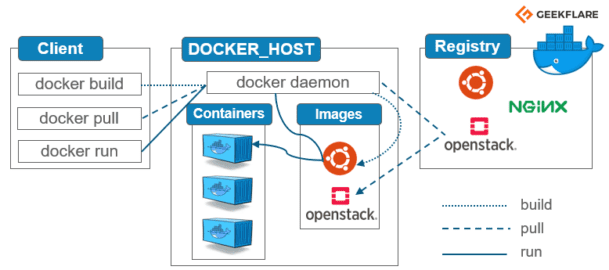
Docker has a client server architecture in that will have Docker client and Daemon.

**Docker host :** The machine on which docker is installed.

**Docker daemon**:: It’s a continuous running process which runs in the background,it does most of the heavy lifting like maintaining images,containers,network,storage and other things along with this docker daemon always listens for commands from docker client.

**Docker Client**: it’s a command line interface which interacts with daemon.This will connect with Daemon using below three ways

1. API 2) Unix socket 3)Network socket.

****

**What is Docker registry?**

It’s a remote repository where the docker images can be stored

1) **Public**: Example is docker hub,where we can pull/store images,but it is not secure

2) **Private:** we can create a private repo to store the images securely.

|  |  |
| --- | --- |
|  |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Limitations of docker compose version1?**

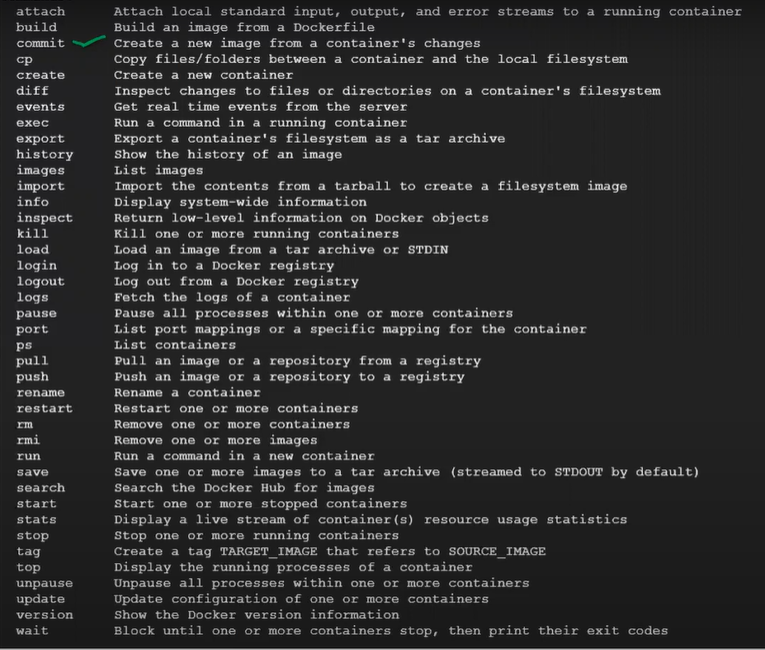
We can’t deploy the containers on other networks other than “Bridge”.

**Advntages of version2 over version 1 compose file?**

It’ll create default network and creates the container,so that they can communicate with each other without LINKS

But the only limitation is we need to mention the DEPENDS\_ON field (for the images which are dependent and need to make sure first image shd be up)

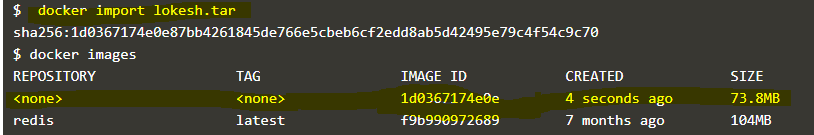
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------Docker basic commands



1. Docker commit : (to create an image out of container)

Docker commit “container ID” “name of the image”

1. Docker diff container ID
2. Docker events –realtime events
3. docker export bdf > lokesh.tar
4. docker import to create an image

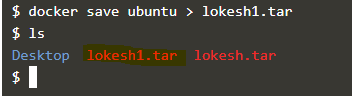


1. **docker info –info about the docker**

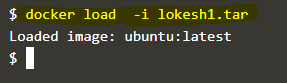
**docker export and import ----containers**

**docker save and load ----images**

**docker save**



**Docker load**



**Docker update---**used to update the configurations like cpu,memory for a containers

**Docker wait** ----to block container until it stops ,when there is no use of that container.

Alternatives for Docker?

## [containerd](https://containerd.io/)

## [Mesos Containerizer](http://mesos.apache.org/documentation/latest/mesos-containerizer/)

## [LXC Linux Containers](https://linuxcontainers.org/)

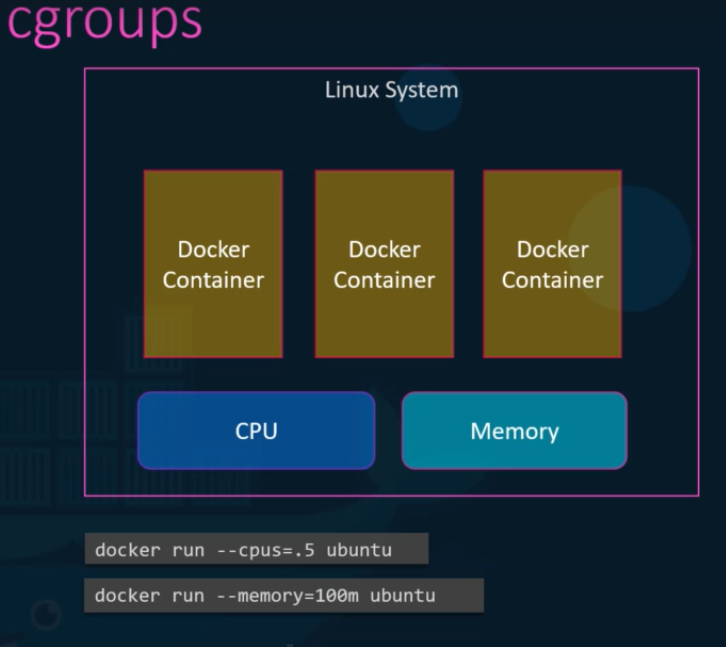
## [CoreOS rkt](https://coreos.com/rkt/docs/latest/)

**What happens to the container before its running?**

When we run “docker run image name” daemon will check in the cache if the image is present then it will run a container out of it ,if not it pull the image from docker hub ,and then it will create.

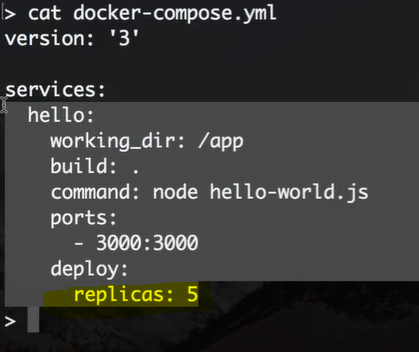
**Cgroups** to limit the hardaware resources.

A cgroup limits an application to a specific set of resources. For example, you can limit the memory available to a specific container.



**Services:**

Services allow you to scale containers across multiple Docker daemons, A service allows you to define the desired state, such as the number of replicas of the service that must be available at any given time. By default, the service is load-balanced across all worker nodes.



**Q. Docker Vs VM (Virtual Machine)**

|  |  |
| --- | --- |
| **Docker Containers** | **Virtual Machine** |
| Lightweight and performance will be fast | Heavyweight and performance is slower compared to containers |
| Virtualization at OS level | Virtualization at hardware level |
| Requires less memory | Requires more memory |
| Startup time will be in milliseconds | Startup time will be minutes |
| Ex: LXC.LXD,Docker | Ex:: Hyper V,Vsphere,Virtual box |
| No need of licences,cost less | Licences are needed |
| Can create containers within a matter of seconds ,using docker images | VM creation may takes a hour |
| Operating System resources can be shared within Docker | Separate Operating System for each VM |

**FQDN in Docker ?**

Embedded DNS on the docker (127.0,0,11) will maintain data of each containers.

Container name /ip

**How will you make sure that the containers are in same n/w?**

While running a container u can specify on which network u want to deploy it “—driver bridge”

**How to restrict resources in docker.**

By using Cgroups we can restrict the resources.

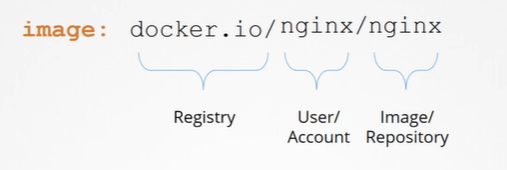
**What is port forwarding?**

Port forwarding or port mapping is a technique of forwarding a data from the container port to the host port.

**How docker image is pulled from private repo?**

Whenever we simply mention image name as “nginx” without a registry and account name it is assumed to be a docker hub,,if we really need to tell docker to pull image from the private repo we need to explicitly mention “REGISTRY”(dns name),,account and the image name. as shown below.





**What is Docker store ?**

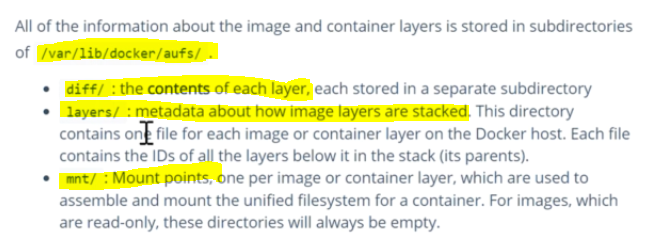
Docker store is part of the Docker hub, enterprises “with compliant, commercially supported software from trusted and verified publishers, that is packaged as Docker images.

**Who is responsible for maintaining layered architecture (Union File system),creating writable & readable layers ?**

Its storage drivers, and selection of the storage drivers is depends on the use of underlying OS

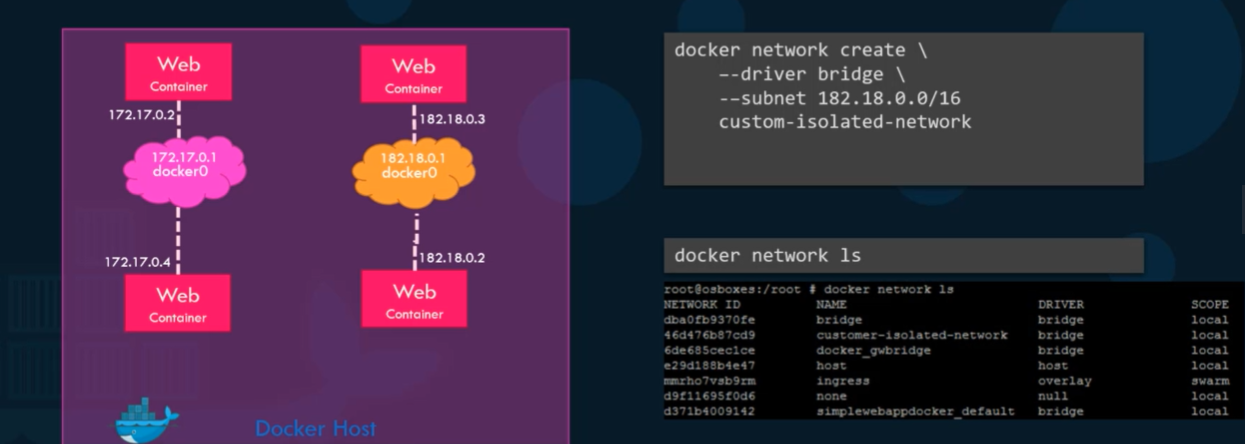
Ubuntu—AUFS (default)

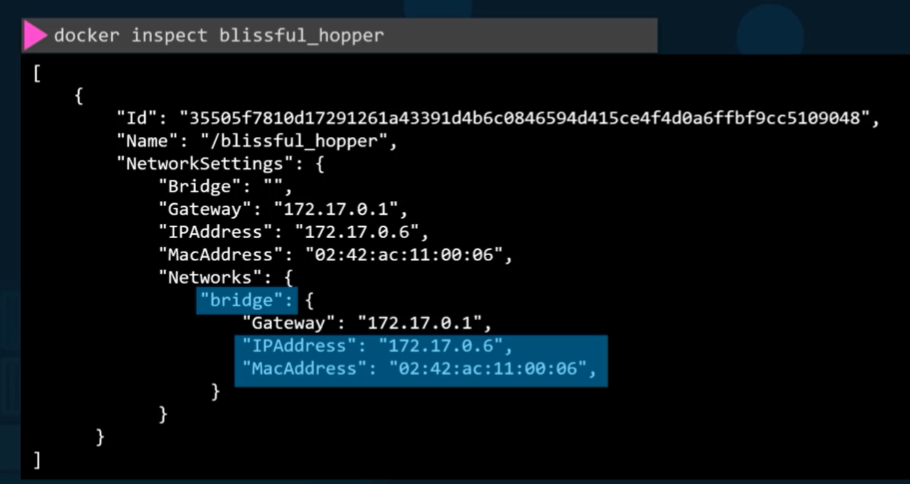




**How to create user-defined networks?**

Docker network create --driver bridge --subnet 182.18.0.0/16 custom-isolated-network

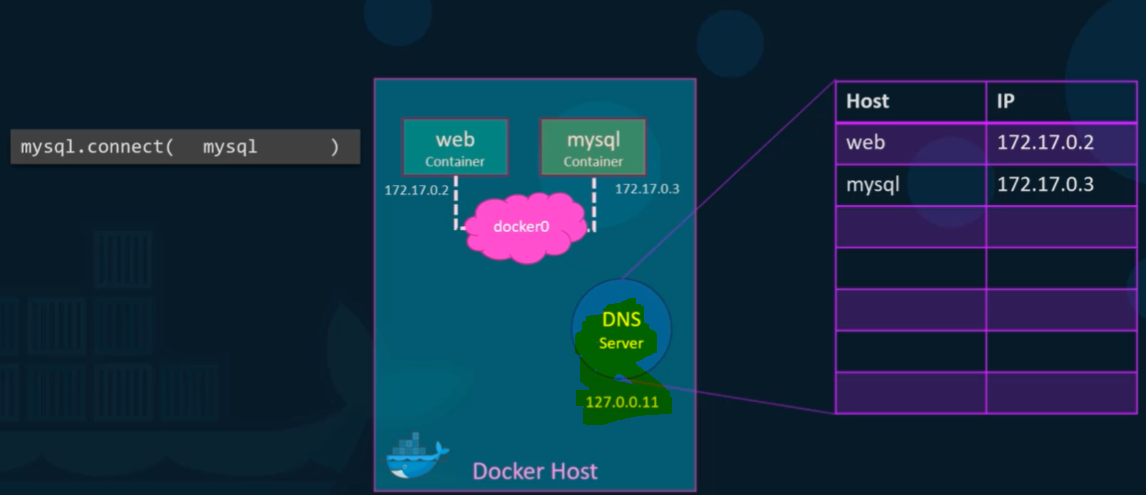




# Can you please explain about Embedded DNS server?

All the containers in a docker host can resolve each other using their name.

And by default docker DNS server IP will be 127.0.0.11 and it maintains an IP table for each container.



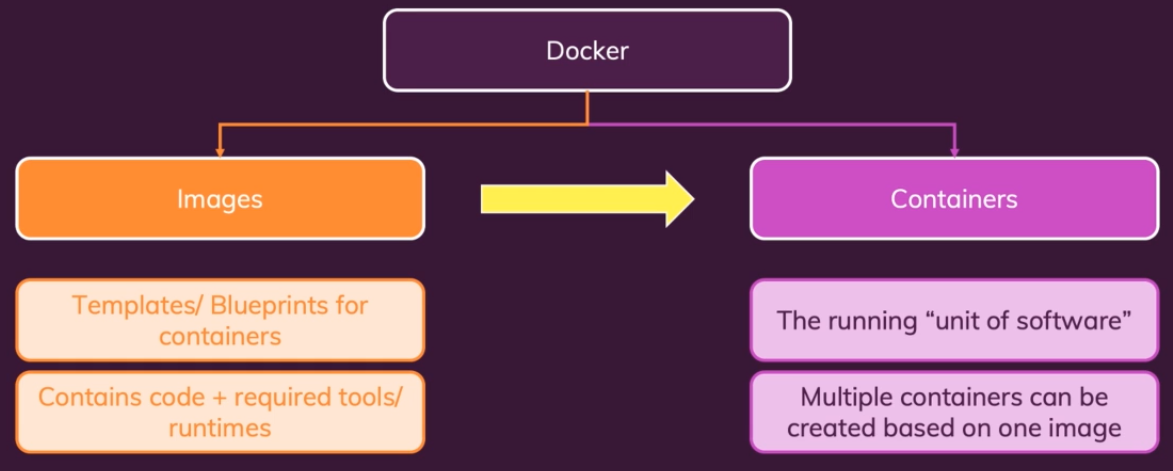
**How is Docker different from other container technologies?**

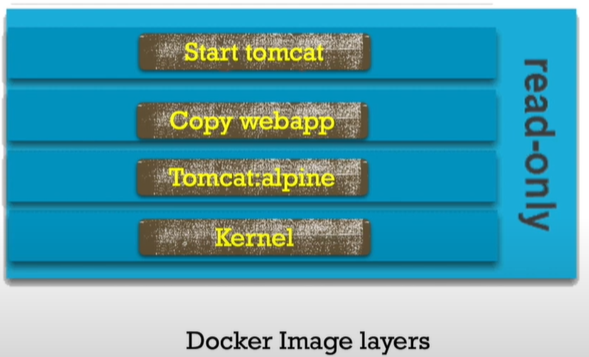
* Docker is the latest and most popular container technology
* Docker containers are very easy to deploy in any cloud platform.
* There is no limitation on running Docker containers.
* Docker is one of the best-documented technologies available in the Containerization space.

**Q. What is Docker image?**

Docker image is a read-only template/plan that contains source code ,libraries ,dependancies and set of instructions for creating a container.

Docker image is immutable(un-changeble)





**Q. What is Docker container?**

* Docker containers are basically runtime instances of Docker images.
* Everytime when we create/start a container,an extra writable layer is created on top of the image layer.



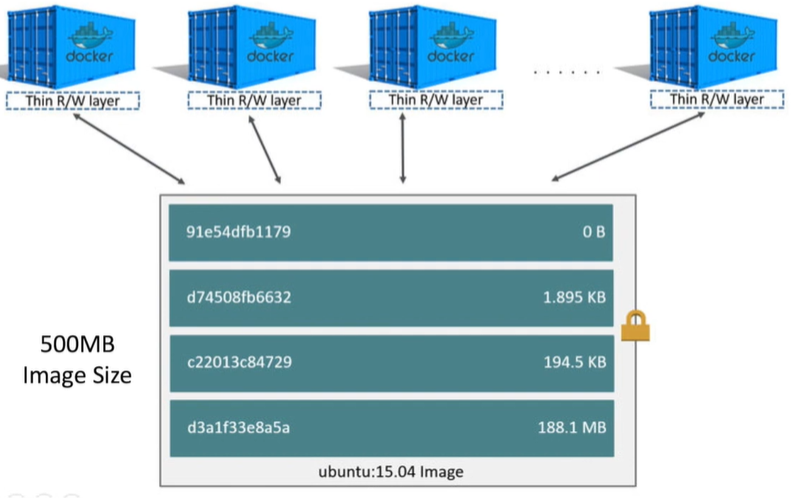
**Is there any dependency between docker image and docker container?**

Docker image is not dependent on container ,bcz we can pull an image without creating/running container out of it.

But container is dependent on image ,when we create/run a container a writeable layer is created on top of the image later(read-only),also the writeable layer contains a copy of image layer ,so that it allows us to do some modification on configuration if required.

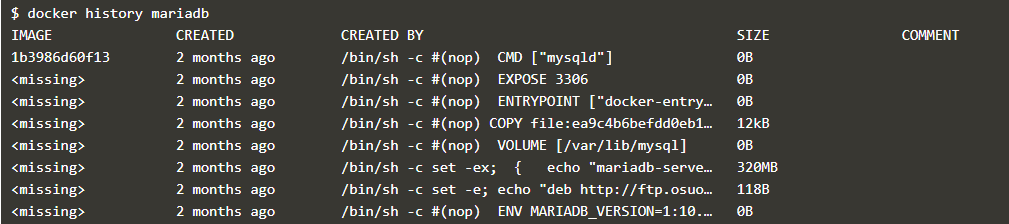
**Suppose host server has 2GB space and your image size is 500MB then is it possible to create more than 4 containers of this image on the host?**

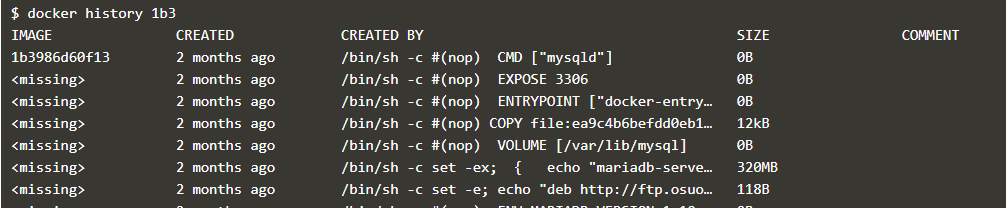
Yes, we can create it when we pull an image it downloaded only ones ,and when we create a containers out of it docker uses cache and it helps us to create more containers.



**How to check details about each layer in docker image like size,task done and time stamp?**

docker history <imagename/id>



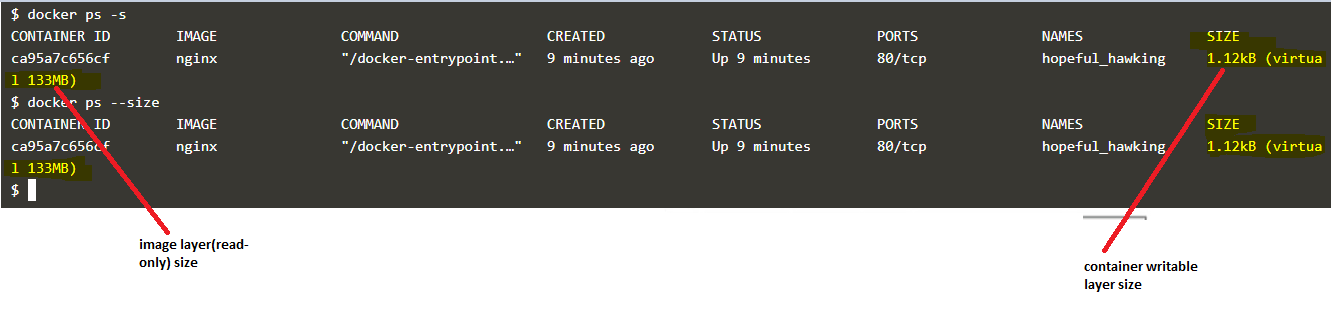


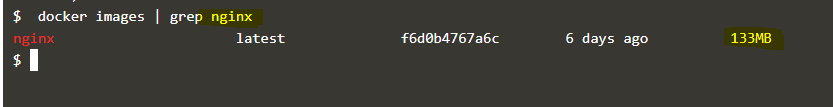
**Is it possible to create containers writable layer ,without running a docker container?**

Yes ,using “docker create” command

**How to find size of any docker containers writable layer?**

docker ps –s **or** docker ps --size





**. What is Docker hub?**

* Docker Hub is a registry service on the cloud that allows you to download Docker images that are built by other communities.
* You can also upload your own Docker built images to Docker hub.

**Q. What is Docker Swarm?**

Docker swarm is a container orchestration tool, meaning that it allows the user to manage multiple containers deployed across multiple host machines

**. Can I use JSON instead of YAML for my compose file in Docker?**

YES, you can very comfortably use JSON instead of the default YAML for your Docker compose file. In order to use JSON file with compose, you need to specify the filename to use as the following:

docker-compose -f docker-compose.json up

**Q. How to create Docker container?**

**docker run –it imagename**

**docker run -d image name**  
The command above will create the container and also starts it for you. In order to check whether the Docker container is created and whether it is running or not, you could make use of the following command. This command will list out all the Docker containers along with its statuses on the host that the Docker container runs.  
**docker ps -a**

**Q. How to stop and restart the Docker container?**

**docker stop CONTAINER\_ID**

**docker restart CONTAINER\_ID**

**How far do Docker containers scale? Are there any requirements for the same?**

* Large web deployments like Google and Twitter and platform providers such as Heroku and dotCloud, all run on container technology.
* Containers can be scaled to hundreds of thousands or even millions of them running in parallel.
* Talking about requirements, containers require the memory and the OS at all the times and a way to use this memory efficiently when scaled.

**Q. What platforms does Docker run on?**

Docker is currently available on the following platforms and also on the following Vendors or Linux:

• Ubuntu 12.04, 13.04  
• Fedora 19/20+  
• RHEL 6.5+  
• CentOS 6+  
• Gentoo  
• ArchLinux  
• openSUSE 12.3+  
• CRUX 3.0+  
Docker is currently available and also is able to run on the following Cloud environment setups given as below:  
• Amazon EC2  
• Google Compute Engine  
• Microsoft Azure  
• Rackspace

Docker is extending its support to Windows and Mac OSX environments and support on Windows has been on the growth in a very drastic manner.

**Q. Do I lose my data when the Docker container exits?**

No, you won't lose any data when Docker container exits. Any data that your application writes to the container gets preserved on the anonymous volume under (var/lib/docker) directory. until you explicitly delete the container.

**Q. What’s the benefit of “Dockerizing?”**

Listed below are the few advantages of dockerizing your environment.

* **Continuous Integration**: Any changes in the code will be automatically deployed immediately and would be available for testing anytime. Thus, [Docker helps in Continuous Integration](http://www.tothenew.com/devops-chef-puppet-docker) by significantly reducing the time.
* **Continuous Delivery**: The transition time from development to production can be greatly reduced as one container can be used across multiple environments. This way applications can be delivered much faster and in a more reliable way than ever before.
* **Portability**: Docker can be moved from one server to another with ease. Docker images come very handy while moving the container from one server to another without much efforts thereby saving a lot of time. The images can be either private or public.
* **Scalability**: A Docker container is very lightweight in size like tens of megabytes when compared to gigabytes in the case of virtual machines. Thus, multiple docker containers can be launched on a single machine and are highly scalable as per the demand.
* **Micro Services Integration**: Integrating microservices with the applications running on containers is easy. Each tier of a multi-tier application running on Docker behaves as an independent container and can be used to integrate microservices with the application.
* **Reduced Cost**: Due to its various advantages such as continuous integration and continuous delivery, Docker significantly reduces the cost of running an application .

**Q. How many containers can run per host?**

* Depending on the environment where Docker is going to host the containers, there can be as many containers as the environment supports.
* The application size, available resources (like CPU, memory) will decide on the number of containers that can run on an environment.
* Though containers create newer CPU on their own but they can definitely provide efficient ways of utilizing the resources.
* The containers themselves are super lightweight and only last as long as the process they are running.

.

**Q. Is there a way to identify the status of a Docker container?**

We can identify the status of a Docker container by running the command ‘docker ps –a’, which will in turn list down all the available docker containers with its corresponding statuses on the host. From there we can easily identify the container of interest to check its status correspondingly.

**Q. What are the differences between the ‘docker run’ and the ‘docker create’?**

The most important difference that can be noted is that, by using the ‘docker create’ command we can create a Docker container in the Stopped state. We can also provide it with an ID that can be stored for later usages as well.  
This can be achieved by using the command ‘docker run’ with the option –cidfile FILE\_NAME as like this:  
‘docker run –cidfile FILE\_NAME’

**Q. Can you remove a paused container from Docker?**

* To answer this question blatantly, No, it is not possible to remove a container from Docker that is just paused.
* It is a must that a container should be in the stopped state, before it can be removed from the Docker container.

**Q. Is there a possibility that a container can restart all by itself in Docker?**

* To answer this question blatantly, No, it is not possible.
* The default –restart flag is set to never restart on its own. **(on-failure & always)**

**Q. What is the preferred way of removing containers - ‘docker rm -f’ or ‘docker stop’ then followed by a ‘docker rm’?**

* The best and the preferred way of removing containers from Docker is to use the ‘docker stop’, as it will allow sending a SIG\_HUP signal to its recipients giving them the time that is required to perform all the finalization and cleanup tasks.
* Once this activity is completed, we can then comfortably remove the container using the ‘docker rm’ command from Docker and thereby updating the docker registry as well.

**Q. Difference between Docker Image and container?**

* Docker container is the runtime instance of docker image.
* Docker Image does not have a state and its state never changes as it is just set of files whereas docker container has its execution state.

**Write a Dockerfile to install nginx?**

Now, let us look at another Dockerfile shown below:

|  |
| --- |
| **FROM ubuntu MAINTAINER vijay (**[**vijay15.biradar@gmail.com**](mailto:vijay15.biradar@gmail.com)**)  RUN apt-get update && apt-get install -y nginx ENTRYPOINT [“/usr/sbin/nginx”,”-g”,”daemon off;”] EXPOSE 80** |

**Explain some docker commands?**

**Docker – version**

You can check the currently used Docker version on your system through this command -

$ docker –version

**Docker pull**

This command can pull the images from docker’s hub or repository that is hub.docker.com

$ docker pull ubuntu

All the images of the hub will be cached and stored from docker’s hub.

**Docker run**

You can create a container from the image through this command.

$ docker run –d ubuntu

**Docker ps**

To check the running containers or to know that how many containers are running right now, you can use this command:

$ docker ps

**Docker ps –a**

To view all the running and exited containers, you can use this command:

$ docker ps –a

**Docker exec**

To access the running container, you can use this command:

$ docker exec it <container id> bash

**Docker stop**

To stop the running container, we can use this command:

$ docker stop <container id>

**Docker kill**

The containers get killed after getting stopped by this command. In Docker, stop command container gets the full time to shut down, but when you need to shut down any container immediately then you can kill the Docker container through kill command.

$ docker kill <container id>

**Docker commit**

To create a new image of the edited container on the local system, you can use this command:

$ docker commit <container id> <username/imagename>

**Docker login**

To login the docker hub repository, you can use this command:

$ docker login

**Docker push**

You can push a new image into Docker hub through this command:

$ docker push <username/image name>

**Docker images**

All locally stored images in docker hub will be listed through this command:

$ docker images

**Docker rm**

If you want to delete any stopped container then this command can help you:

$ docker rm <container id>

**Docker build**

If you want to build an image from a Docker file then you can use this command:

$ docker build <path to docker file>

Apart from the above-listed Docker commands cheat sheet, one can also use other commands for Docker like ‘docker export’ command that can export a container’s filesystem as an archive file or ‘docker attach’ that can attach any running container, etc.

## Docker network

To view Docker networks, run:

docker network ls

To get further details on networks, run:

docker network inspect

docker network inspect bridge

docker network create --driver bridge <bridge\_network\_name>

Create the overlay network in a similar manner to the bridge network (network name my\_multi\_host\_network):

1. **docker network create --driver overlay <bridge\_network\_name>**
2. Launch containers on each host; make sure you specify the network name:

docker run -itd -net=<bridge\_network\_name> my\_python\_app

we can use host network directly.

docker run -d --name web1 --net=host <image\_name>

docker run -d --name web1 --net=host nginx

docker run -d --name web2 --net=none <image\_name>

**55. Can we run multiple apps on one server with Docker?**

Yes, theoretically we can run multiples apps on one Docker server. But in practice, it is better to run different components on separate containers.

With this we get cleaner environment and it can be used for multiple uses.

**57. What is the most popular use of Docker?**

* The most popular use of Docker is in build pipeline.
* With the use of Docker it is much easier to automate the development to deployment process in build pipeline.
* We use Docker for the complete build flow from development work, test run and deployment to production environment.



**64. What is Docker Machine?**

* We can use Docker Machine to install Docker Engine on virtual hosts.
* It also provides commands to manage virtual hosts.
* Some of the popular Docker machine commands enable us to start, stop, inspect and restart a managed host.
* Docker Machine provides a Command Line Interface (CLI), which is very useful in managing multiple hosts.

**66. How will you create a Container in Docker?**

* To create a Container in Docker we have to create a Docker Image. We can also use an existing Image from Docker Hub Registry.
* We can run an Image to create the container.

**67. Do you think Docker is Application-centric or Machine-centric?**

* Docker is an Application-centric solution.
* It is optimized for deployment of an application.
* It does not replace a machine by creating a virtual machine. Rather, it focuses on providing ease of use features to run an application.

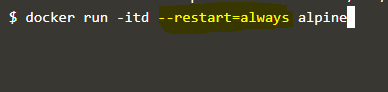
**Lets say 10+ containers are running on my host .All these containers stops on server reboot or docker engine restart ,which requires a manual container start.Is there any way to auto restart the containers?**

Enable auto restart policy

\*on-failure

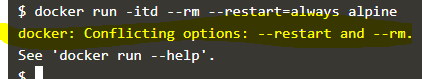
\*always

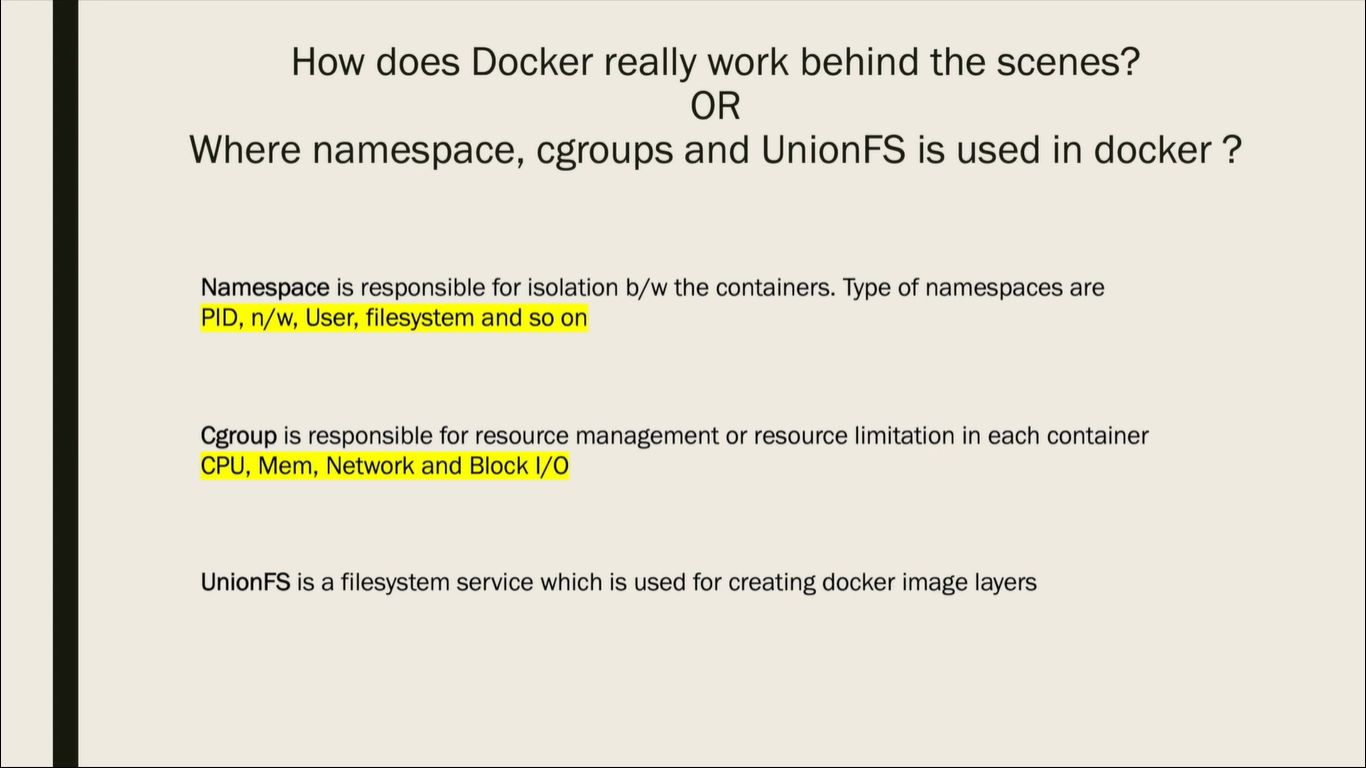
\*unless-stopped



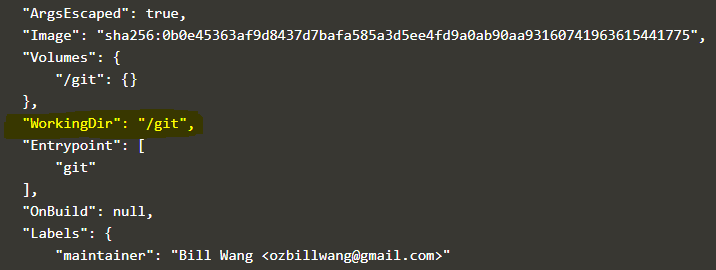
**Can we use restart policy and –rm flag together when starting a container?**

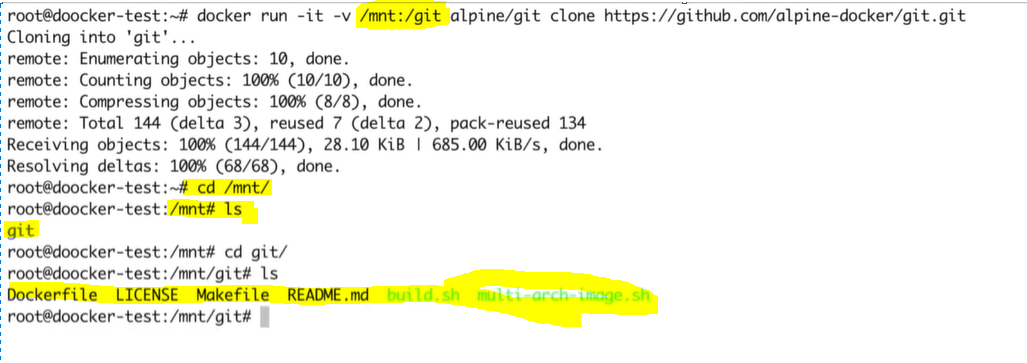
No we can’t





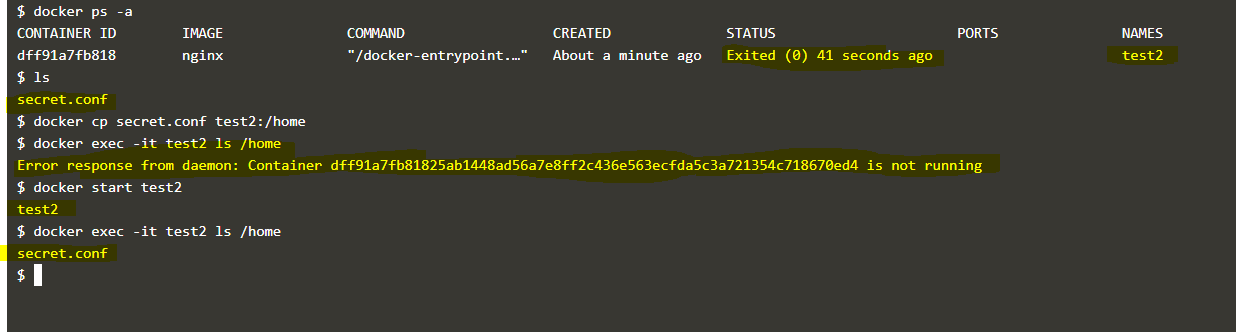
**I want to clone a github repo but I don’t have a git utility installed in my system,although docker is running in my system and git image is also available .So considering the scenario ,how can I clone the repo now?**



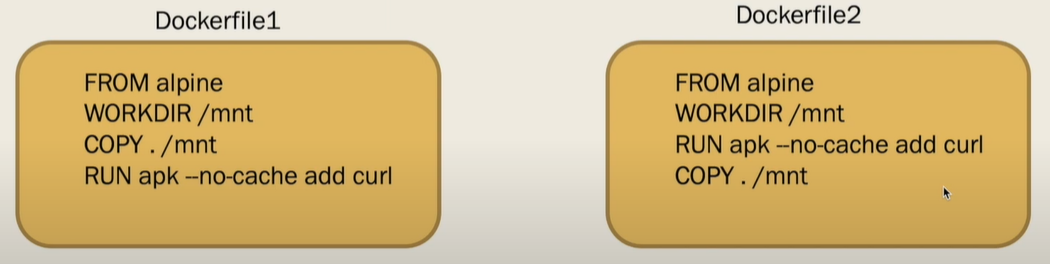


**I have a few running and stopped containers on my host.i have to copy a secret.conf file in all the containers.Is it possible to copy a file inside a stopped container without starting it?**

It is possible to copy files to stopped containers but we can’t interact with that container.



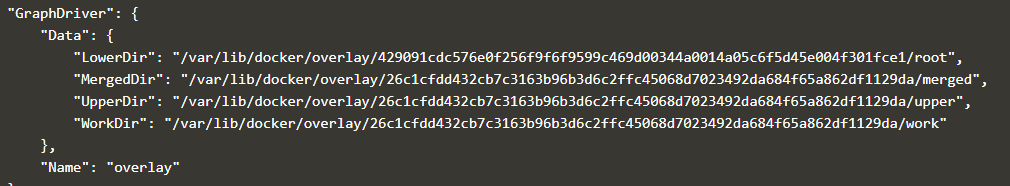
**Choose the preferred Dockerfile which uses docker build cache feature correctly?**

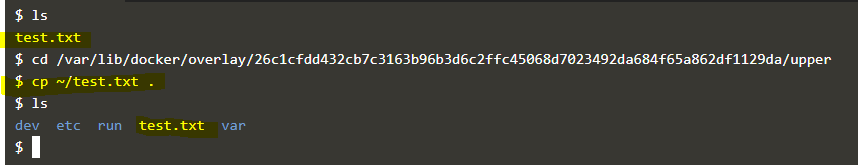


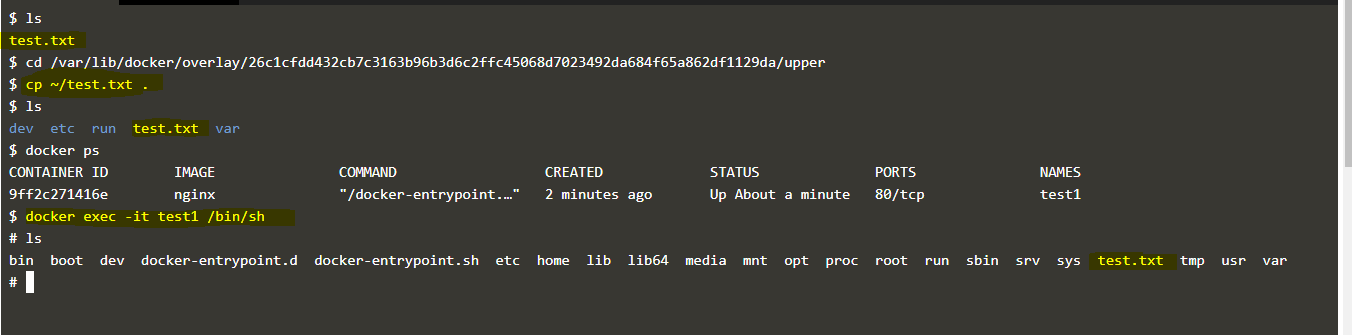
We need to availd COPY/ADD instruction in early stages of the Dockerfile.



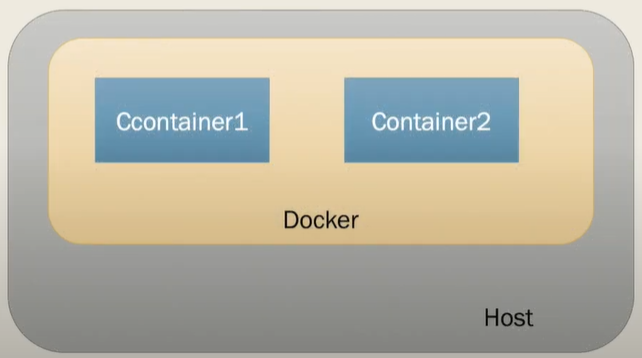
**I have a requirement to copy token/config file inside a container but can’t use docker command for that? Is there any alternate way to achieve this?**







This can also be used for container to container

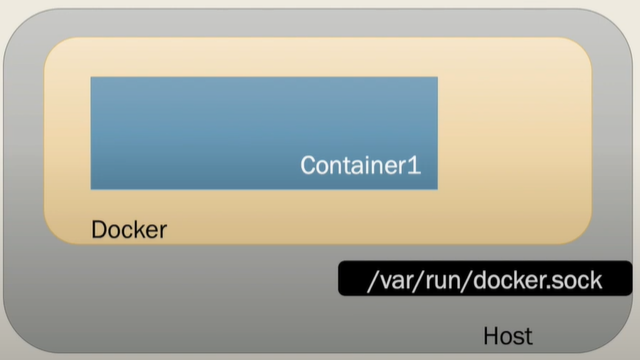
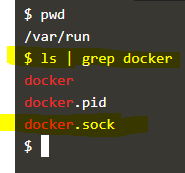


**Is it possible to pull/push image ,run/stop other docker containers from inside a container? Or talk to host daemon from inside a container?** (comm b/w docker daemon and container)

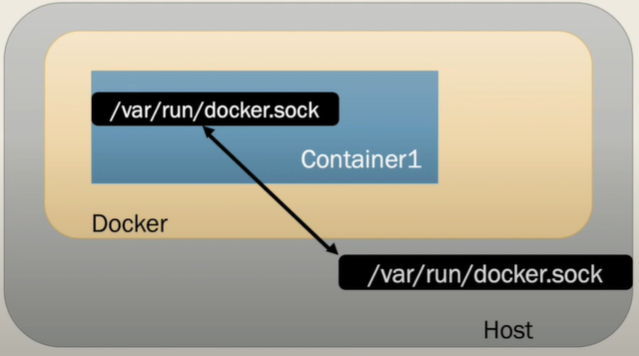
When we create a docker daemon on host it opens a unix socket with the name **docker.socket** under **/var/run**

and starts listening for the user inputs on the socket.

Docker API and docker client use this socket to interact with daemon.



So for our use-case we use this socket ,where we mount this socket inside container as a volume

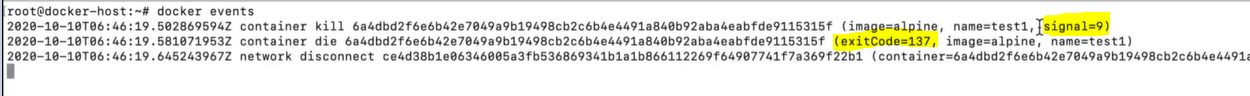




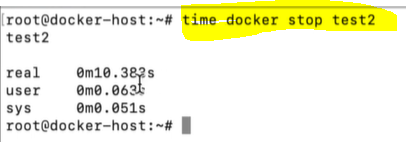
**Stopping a running container using docker stop is slow as compared to docker kill but still Is advised to use docker stop ,why?**

**SIG KILL—**directly send signal terminate(signal=9)





**SIG STOP:** signal terminate (signal=15) + signal kill (signal=9)





**I have an app and db container running on host .How can these two containers talk to each other?**

Use custom bridge network and run the containers on that

**What are the main security concerns with Docker based containers?**

Docker based containers have following security concerns:

1. **Kernel Sharing** : In a container-based system, multiple containers share same Kernel. If one container causes Kernel to go down, it will take down all the containers. In a virtual machine environment we do not have this issue.
2. **Container Leakage** : If a malicious user gains access to one container, it can try to access the other containers on the same host. If a container has security vulnerabilities it can allow the user to access other containers on same host machine.
3. **Denial of Service** : If one container occupies the resources of a Kernel then other containers will starve for resources. It can create a Denial of Service attack like situation.
4. **Tampered Images** : Sometimes a container image can be tampered. This can lead to further security concerns. An attacker can try to run a tampered image to exploit the vulnerabilities in host machines and other containers.
5. **Secret Sharing** : Generally one container can access other services. To access a service it requires a Key or Secret. A malicious user can gain access to this secret. Since multiple containers share the secret, it may lead to further security concerns.

**75. What are the main benefits of using Docker?**

Docker is a very powerful tool. Some of the main benefits of using Docker are as follows:

1. **Utilize Developer Skills** : With Docker we maximize the use of Developer skills. With Docker there is less need of build or release engineers. Same Developer can create software and wrap it in one single file.
2. **Standard Application Image** : Docker based system allows us to bundle the application software and Operating system files in a single Application Image that can be deployed independently.
3. **Uniform deployment** : With Docker we can create one package of our software and deploy it on different platforms seamlessly.

**How does Docker simplify Software Development process?**

Prior to Docker, Developers would develop software and pass it to QA for testing and then it is sent to Build & Release team for deployment.

In Docker workflow, Developer builds an Image after developing and testing the software. This Image is shipped to Registry. From Registry it is available for deployment to any system. The development process is simpler since steps for QA and Deployment etc take place before the Image is built. So Developer gets the feedback early.

**What type of applications- Stateless or Stateful are more suitable for Docker Container?**

* It is preferable to create Stateless application for Docker Container.
* We can create a container out of our application and take out the configurable state parameters from application.
* Now we can run same container in Production as well as QA environments with different parameters.
* Also a stateless application is much easier to scale with Docker Containers than a stateful application.

**How can Docker run on different Linux distributions?**

* Docker directly works with Linux kernel level libraries.
* In every Linux distribution, the Kernel is same.
* Docker containers share same kernel as the host kernel.
* Since all the distributions share the same Kernel, the container can run on any of these distributions.

**Why do we use Docker on top of a virtual machine?**

* Generally we use Docker on top of a virtual machine to ensure isolation of the application.
* On a virtual machine we can get the advantage of security provided by hypervisor.
* We can implement different security levels on a virtual machine.
* And Docker can make use of this to run the application at different security levels.

**What is the difference between Add and Copy command in a Dockerfile?**

Both Add and Copy commands of Dockerfile can copy new files from a source location(host) to a destination in Container’s file path.

They behave almost same.

The main difference between these two is that Add command can also read the files from a URL.  Secondly, you can extract a tar file from the source directly into the destination.

As per Docker documentation, **Copy command is preferable**. Since Copy only supports copying local files to a Container, it is preferred over Add command.

**What is Build cache in Docker?**

* When we build an Image, Docker will process each line in Dockerfile.
* It will execute the instructions on each line in the order that is mentioned in the file.
* But at each line, before running any command, Docker will check if there is already an existing image in its cache that can be reused rather than creating a new image.
* This method of using cache in Docker is called Build cache in Docker.
* We can also specify the option **–no-cache=true** to let Docker know that we do not want to use cache for Images. With this option, Docker will create all new images.

**What is the purpose of EXPOSE command in Dockerfile?**

We use EXPOSE command to inform Docker that Container will listen on a specific network port during runtime.  
But these ports on Container may not be accessible to the host. We can use –p to publish a range of ports from Container.

**What are the Cloud platforms that support Docker?**

Some of the popular cloud platforms that support Docker are:

1. Amazon AWS
2. Google Cloud Platform
3. Microsoft Azure
4. IBM Bluemix

**92. How can we control the start-up order of services in Docker compose?**

In Docker compose we can **use the depends on option** to control the start-up order of services.

With compose, the services will start in the dependency order. Dependencies can be defined in the options like- depends\_on, links, volumes\_from, network\_mode etc.

But Docker does not wait for until a container is ready.

**Docker hub**:: Docker Hub is a service provided by Docker for finding and sharing container images.

**Docker store**:: Docker store will offer enterprises “with compliant, commercially supported software from trusted and verified publishers, that is packaged as Docker images,

**Container**—Run time environment of docker image.(An isolated area for an application to run)

\*\*Note :: **We can create multiple containers of the same image if container names are diff.**

++For the application running on container ,container looks like a fully fledged OS.

++For the Host OS container is one process.

++Docker on wind7 or 8 ,will create a linux vm(Virtual box) and in it it’ll create a container.

**Micro service**: smaller self-executable service.

Ex: Uber,uber eats Gmail,youtube,Google Amazon,kindle,amazon prime

++docker rm –f $(docker ps –q)

FROM nginx

MAINTAINER Lokesh

RUN touch test.txt 🡨 docker build –t myfirstimage .

CMD ["echo","myfirstimage"] 🡨 docker run myfirstimage

**Docker important commands**

**docker ps** | grep -i <container\_name> —Lists the specified Docker containers

**docker ps -a** | grep <container\_name> —Lists the specified Docker containers including exited containers

**docker stats** <container\_id>-Running docker stats on multiple containers by name and id against a Linux daemon

**docker ps** | grep <container\_id>—

**docker logs** --tail 100 <container\_id>—Number of lines to show from the end of the logs

**docker stats** --no-stream $(docker ps | grep selenium | awk '{print $1}')

**docker restart** <container\_id> -Restart Docker container

docker ps -a | grep rabbitmq | awk '{print $1}’ —rabbitmq container ID which is the first column value

docker rm $(docker ps -a | grep rabbitmq | awk '{print $1}’)—Remove all containers using the rabbitmq image:

docker rm $(docker ps -a | grep "46 hours ago”)—Or by time created:

**docker ps -a | grep -i exited | head -n 10**

**docker ps -a | head**

**docker ps -a** **| grep -i "exited"**

**Q. Why do my services take 10 seconds to recreate or stop?**

* Docker compose stop will attempt to stop a specific Docker container by sending a SIGTERM message.(Termination signal)
* Once this message is delivered, it waits for the default timeout period of 10 seconds and once the timeout period is crossed, it then sends out a SIGKILL message to the container – in order to kill it forcefully.
* If you are actually waiting for the timeout period, then it means that the containers are not shutting down on receiving SIGTERM signals / messages.

In an attempt to solve this issue, the following is what you can do:

• You can ensure that you are using the JSON form of the CMD and also the ENTRYPOINT in your dockerfile.  
• Use [“program”, “argument1”, “argument2”] instead of sending it as a plain string as like this – “program argument1 argument2”.  
• Using the string form, makes Docker run the process using bash that can’t handle signals properly. Compose always uses the JSON form.  
• If it is possible then modify the application which you intend to run by adding an explicit signal handler for the SIGTERM signal  
• Also set the stop\_signal to a proper signal that the application can understand and also know how to handle it

**Q. How do I run multiple copies of a Compose file on the same host?**

* Docker’s compose makes use of the Project name to create unique identifiers for all of the project’s containers and resources.
* In order to run multiple copies of the same project, you will need to set a custom project name using the –p command line option or you could use the COMPOSE\_PROJECT\_NAME environment variable for this purpose.

**Q. What’s the difference between up, run, and start?**

* On any given scenario, you would always want your docker-compose up. Using the command UP, you can start or restart all the services that are defined in a docker-compose.yml file.
* In the “attached” mode, which is also the default mode – we will be able to see all the log files from all the containers.
* In the “detached” mode, it exits after starting all the containers, which continue to run in the background showing nothing over in the foreground.
* Using docker-compose run command, we will be able to run the one-off or the ad-hoc tasks that are required to be run as per the Business needs and requirements.
* This requires the service name to be provided which you would want to run and based on that, it will only start those containers for the services that the running service depends on.
* Using the run command, you can run your tests or perform any of the administrative tasks as like removing / adding data to the data volume container.
* It is also very similar to the docker run –ti command, which opens up an interactive terminal to the containers an exit status that matches with the exit status of the process in the container.
* Using the docker-compose start command, you can only restart the containers that were previously created and were stopped. This command never creates any new Docker containers on its own.

**What is Docker compose**

tool for defining & running multi-container docker applications

use yaml files to configure application services (docker-compose.yml)

can start all services with a single command : **docker compose up**

can stop all services with a single command : **docker compose down**

**can scale up selected services when required**

* docker compose is a tool for defining and running multi containers docker application.
* docker compose allows us to define all the services in a configuration file and with one command it will spin up all the containers that we need.
* it uses yaml files to configure application services (docker-compose.yml)
* it uses single command to start and stop all the services (docker-compose up & docker-compose down)
* it can scale up services when ever required.

version: '3'

services:

web:

image: nginx

db:

image: mysql

ports:

- "3306:3306"

environment:

- MYSQL\_ROOT\_PASSWORD=password

- MYSQL\_USER=user

- MYSQL\_PASSWORD=password

- MYSQL\_DATABASE=demodb

docker-compose up

docker-compose up -d

docker-compose ps

## What is a service?

* A service is a group of containers of the same **image:tag**.
* Services make it simple to scale your application.
* Services are really just “containers in production.”
* A service only runs one image, but it codifies the way that image runs—what ports it should use, how many replicas of the container should run so the service has the capacity it needs, and so on.
* Scaling a service changes the number of container instances running that piece of software, assigning more computing resources to the service in the process.

When you create a service, you specify which container image to use and which commands to execute inside running containers. You also define options for the service including:

* the port where the swarm makes the service available outside the swarm
* an overlay network for the service to connect to other services in the swarm
* CPU and memory limits and reservations
* a rolling update policy
* the number of replicas of the image to run in the swarm

docker service create --replicas 5 -p 80:80 --name web nginx

docker service scale web=8

**56. What are the main features of Docker-compose?**

Some of the main features of Docker-compose are as follows:

1. **Multiple environments on same Host** : We can use it to create multiple environments on the same host server.
2. **Preserve Volume Data on Container Creation** : Docker compose also preserves the volume data when we create a container.
3. **Recreate the changed Containers** : We can also use compose to recreate the changed containers.
4. **Variables in Compose file** : Docker compose also supports variables in compose file. In this way we can

create variations of our containers.

**58. What is the role of open source development in the popularity of Docker?**

* Since Linux was an open source operating system, it opened new opportunities for developers who want to contribute to open source systems.
* One of the very good outcomes of open source software is Docker.
* It has very powerful features.
* Docker has wide acceptance due to its usability as well as its open source approach of integrating with different systems.

**59. What is the difference between Docker commands: up, run and start?**

We have up and start commands in docker-compose. The run command is in docker.

1. **Up** : We use this command to build, create, start or restart all the services in a docker-compose.yml file. It also attaches to containers for a service. This command can also start linked services.
2. **Run** : We use this command for adhoc requests. It just starts the service that we specifically want to start. We generally use it run specific tests or any administrative tasks.
3. **Start** : This command is used to start the container that were previously created but are not currently running. This command does not create new containers.

**60. What is Docker Swarm?**

* Docker Swarm is used to create a cluster environment.
* It can turn a group of Docker engines into a Single virtual Docker Engine.
* This creates a system with pooled resources.
* We can use Docker Swarm to scale our application.

**61. What are the features of Docker Swarm?**

Some of the key features of Docker Swarm are as follows:

1. **Compatible** : Docker Swarm is compatible with standard Docker API.
2. **High Scalability** : Swarm can scale up to as much as 1000 nodes and 50000 containers. There is almost no performance degradation at this scale in Docker Swarm.
3. **Networking** : Swarm comes with support for Docker Networking.
4. **High Availability** : We can create a highly available system with Docker Swarm. It allows use to create multiple master nodes so that in case of a failure, another node can take over.

**Node Discovery** : In Docker Swarm, we can add more nodes and the new nodes can be found with any discovery service like etcd or zookeeper etc.

**65. Why do we use Docker Machine?**

There are two main uses of Docker Machine:

1. **Old Desktop** : If we have an old desktop and we want to run Docker then we use Docker Machine to run Docker. It is like installing a virtual machine on an old hardware system to run Docker engine.
2. **Remote Hosts** : Docker Machine is also used to provision Docker hosts on remote systems. By using Docker Machine you can install Docker Engine on remote hosts and configure clients on them.

**68. Can we run more than one process in a Docker container?**

* Yes, a Docker Container can provide process management that can be used to run multiple processes.
* There are process supervisors like runit, s6, daemontools etc that can be used to fork additional processes in a Docker container.

**70. How will you take backup of Docker container volumes in AWS S3?**

We can use a utility named Dockup provided by Docker Cloud to take backup of Docker container volumes in S3.

**71. What are the three main steps of Docker Compose?**

Three main steps of Docker Compose are as follows:

1. **Environment** : We first define the environment of our application with a Dockerfile. It can be used to recreate the environment at a later point of time.
2. **Services** : Then we define the services that make our app in docker-compose.yml. By using this file we can define how these services can be run together in an environment.
3. **Run** : The last step is to run the Docker Container. We use docker-compose up to start and run the application.

**72. What is Pluggable Storage Driver architecture in Docker based containers?**

* Docker storage driver is by default based on a Linux file system. But Docker storage driver also has provision to plug in any other storage driver that can be used for our environment.
* In Pluggable Storage Driver architecture, we can use multiple kinds of file systems in our Docker Container.
* In Docker info command we can see the Storage Driver that is set on a Docker daemon.

We can even plug in shared s

**78. What are the popular tasks that you can do with Docker Command line tool?**

Docker Command Line (DCL) tool is implemented in Go language. It can compile and run on most of the common operating systems. Some of the tasks that we can do with Docker Command Line tool are as follows:

1. We can download images from Registry with DCL.
2. We can start, stop or terminate a container on a Docker server by DCL.
3. We can retrieve Docker Logs via DCL.
4. We can build a Container Image with DCL.

**93. Why Docker compose does not wait for a container to be ready before moving on to start next service in dependency order?**

The problem with waiting for a container to be ready is that in a Distributed system, some services or hosts may become unavailable sometimes. Similarly during startup also some services may also be down.

Therefore, we have to build resiliency in our application. So that even if some services are down we can continue our work or wait for the service to become available again.

We can use wait-for-it or dockerize tools for building this kind of resiliency.

**94. How will you customize Docker compose file for different environments?**

In Docker compose there are two files docker-compose.yml and docker-compose.override.yml. We specify our base configuration in docker- compose.yml file. For any environment specific customization we use docker-compose.override.yml file.

We can specify a service in both the files. Docker compose will merge these files based on following rules:

For single value options, new value replaces the old value.

For multi-value options, compose will concatenate the both set of values.

We can also use extends field to extend a service configuration to multiple environments. With extends, child services can use the common configuration defined by parent service.

Development command:

docker-compose -f docker-compose.yml -f docker-compose.dev.yml up

Production command:

docker-compose -f docker-compose.yml -f docker-compose.prod.yml up

Note: If you name your second dockerfile `docker-compose.override.yml`, a simple docker-compose up would read the overrides automatically.  
But in your case, a name based on the environment is clearer.