1. what is docker?

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

1. what is difference between vm's and docker ?

Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application.

1. what is dockerfile?

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

1. what is difference between the kill and stop?

[docker stop](https://docs.docker.com/engine/reference/commandline/stop/): *Stop a running container (****send SIGTERM, and then SIGKILL after grace period****) [...] The main process inside the container will receive SIGTERM, and after a grace period, SIGKILL. [emphasis mine]*

stop attempts to trigger a graceful shutdown by sending the standard [POSIX signal](https://en.wikipedia.org/wiki/Unix_signal#POSIX_signals) SIGTERM

docker stop will try to stop it gracefully (will ask politely )

* [docker kill](https://docs.docker.com/engine/reference/commandline/kill/): *Kill a running container (****send SIGKILL, or specified signal****) [...] The main process inside the container will be sent SIGKILL, or any signal specified with option --signal. [emphasis mine]*

killjust kills the process by default (but also allows to send any other signal):

docker kill will stop the main entrypoint process/program abruptly

1. what is difference between and Add and copy commands in a docokerfile?

COPY takes in a src and destination. It only lets you copy in a local file or directory from your host (the machine building the Docker image) into the Docker image itself.

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.

ADD is when you want to extract a local tar file into a specific directory in your Docker image. This is exactly what the [Alpine image](https://github.com/gliderlabs/docker-alpine/blob/c7368b846ee805b286d9034a39e0bbf40bc079b3/versions/library-3.5/Dockerfile) does with ADD rootfs.tar.gz /.

1. what is difference between cmd and entry point ?

CMD instruction allows you to set a default command, which will be executed only when you run container without specifying a command. If Docker container runs with a command, the default command will be ignored. If Dockerfile has more than one CMD instruction, all but last CMD instructions are ignored.

ENTRYPOINT instruction allows you to configure a container that will run as an executable. It looks simil Tags are annotations that can be applied to Docker images after they're built. If you look at an image in a repository like Docker Hub or Quay.io you'll see different versions of that image with different tagsar to CMD, because it also allows you to specify a command with parameters. The difference is ENTRYPOINT command and parameters are not ignored when Docker container runs with command line parameters. (There is a way to ignore ENTTRYPOINT, but it is unlikely that you will do it.)

* ENTRYPOINT ["executable", "param1", "param2"] (exec form, preferred)
* ENTRYPOINT command param1 param2 (shell form)

1. which is the base image you will prefer your docker file ? alpine

8. how login a running container ?

docker exec -it contianerid

1. how to check live logs in docker containers?

docker logs –follow

1. how to check no.of containers running your machine ?

docker ps

1. how to check stop and running containers ?

docker ps -a

12. what is port forwarding?

-p 8181:9090

13. How to run a container in background?

-d

14. what is first line in a dockerfile?

FROM

14. what are the commands have you used in your docker file ? FROM RUN CMD ENTRYPOINT label workdir COPY ADD EXPOSE VOLUME

15. what is label in docker? 16. what is docker tag?

**Tags** are annotations that can be applied to **Docker** images after they're built. If you look at an image in a repository like **Docker** Hub or Quay.io you'll see different versions of that image with different **tags.**

17. How you store data persistently? volumes

18. have you worked on docker volumes?

-v hostmachie/path:container/data/path

19. what is docker network ?

Bridge network is default networks that comes with all docker installation. It is also known as docker0 network. If not mentioned otherwise, all docker containers are created within docker0 network.

None network is generally known as container specific network. A container can be attached to none network. This is utilised for internal communication between containers being isolated to outside network.

The host network adds a container on the host’s network stack.. As far as the network is concerned, there is no isolation between the host machine and the container. For instance, if you run a container that runs a web server on port 80 using host networking, the web server is available on port 80 of the host machine.

20. I have a one web application is running on one container and database is running in one container how you communicate these two containers?

21. where you are store docker images? registry2 |jforg |dockerhub

22. how to run multiple containers at a time? docker-compose

23. how to share data between two containers?

container 1 docker run --volumes-from container1 --name container2

24.What is difference between RUN and CMD ?

RUN lets you execute commands inside of your Docker image. These commands get executed once at build time and get written into your Docker image as a new layer.

For example if you wanted to install a package or create a directory inside of your Docker image then RUN will be what you’ll want to use. For example, RUN mkdir -p /path/to/folder.

CMD lets you define a default command to run when your container starts.

You could say that CMD is a Docker run-time operation, meaning it’s not something that gets executed at build time. It happens when you run an image. A running image is called a container.

For example, if you were creating a Dockerfile for your own web application, a reasonable CMD to have would be to start your web application’s app server.