1. **What is Hypervisor?**

* A**hypervisor**, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs).
* A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

**Types of Hypervisor:**

There are two main hypervisor types, such as “Type 1” and “Type 2” Hypervisor.

* **Type 1 Hypervisor (or bare metal):** A type 1 hypervisor acts like a lightweight operating system and runs directly on the host’s hardware.
* **Type 2 Hyper visor**: A type 2 hypervisor runs as a software layer on an operating system, like other computer programs.

1. **What is virtualization?**

* [Virtualization](https://www.redhat.com/en/topics/virtualization) is technology that lets you create useful IT services using resources that are traditionally bound to hardware.
* It allows you to use a physical machine’s full capacity by distributing its capabilities among many users or environments.
* Virtualization is the process of creating a software-based, virtual version of something (compute storage, servers, application, etc). These virtual versions or environments are created from single physical hardware system.

1. **What is containerization?**

* Containerization is the packaging together of software code with all it’s necessary components like libraries, frameworks, and other dependencies so that they are isolated in their own "[container](https://www.redhat.com/en/topics/containers/whats-a-linux-container)."
* This is so that the software or [application](https://www.redhat.com/en/topics/cloud-native-apps/what-are-cloud-applications) within the container can be moved and run consistently in any environment and on any infrastructure, independent of that environment or infrastructure’s operating system.
* The container acts as a kind of bubble or a computing environment surrounding the application and keeping it independent of its surroundings.
* So basically an application that is being developed and deployed is bundled and wrapped together with all its configuration files and dependencies. The bundle is called container.

1. **What is the difference between virtualization and containerization?**
2. **What is Docker?**

* Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
* With Docker, you can manage your infrastructure in the same ways you manage your applications.
* By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

1. **What is Docker container?**

* Docker containers include the application and all of its dependencies.
* It shares the kernel with other containers, running as isolated processes in user space on the host operating system.
* A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.

1. **What are Docker Images?**

* A Docker image contains application code, libraries, tools, dependencies and other files needed to make an application run.
* When a user runs an image, it can become one of many instances of a container.

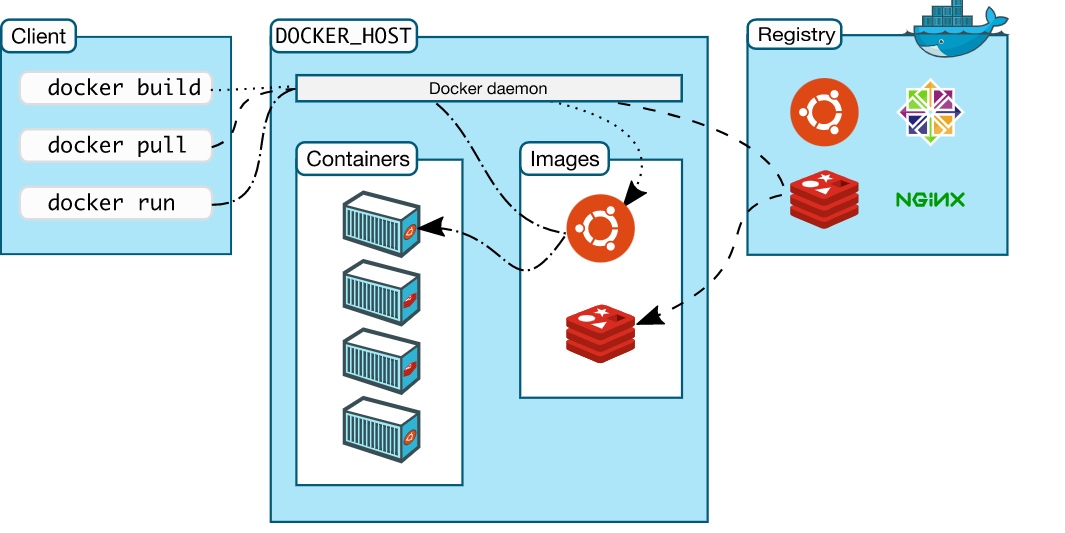
You can create a Docker image by using one or two methods.

* **Interactive:** By running container from an existing Docker image, manually changing that container environment through a series of live steps, and saving the resulting state as a new image.
* **Dockerfile**: By constructing a plain-text file, known as a **Dockerfile**, which provides the specifications for creating a Docker image.

1. **What is Docker Hub?**

* Docker images create Docker containers. There has to be a registry where these Docker images live. The Registry is Docker Hub.
* [Docker Hub](https://hub.docker.com/) is a service provided by Docker for finding and sharing container images with your team.

1. **Explain Docker Architecture**



* Docker uses a **client-server** architecture.
* The Docker ***client***  talks to the **Docker *daemon***, which does the heavy lifting of building, running, and distributing your Docker containers.
* The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon.
* In the above architecture, there 3 major components. 1. Docker daemon, 2. The Docker client, 3. Docker registries.

**Docker Daemon:**

* The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
* A daemon can also communicate with other daemons to manage Docker services.

**The Docker Client:**

* The Docker client (docker) is the primary way that many Docker users interact with Docker.
* When you use commands such as docker run, the client sends these commands to dockerd, which carries them out.
* The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

**Docker Registries:**

* A Docker *registry* stores Docker images.
* Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.
* When you use the docker pull or docker run commands, the required images are pulled from your configured registry.
* When you use the docker push command, your image is pushed to your configured registry.

1. **What is 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.
* Docker can build images automatically by reading the instructions from a Dockerfile.

1. **Tell us something about Docker Compose**

* Docker compose is a YAML file which contains details about the services, networks, and volumes for setting up the Docker application

1. **What is Docker Swarm?**

* Docker Swarm is an orchestration management tool that runs on Docker applications.
* It helps end-users in creating and deploying a cluster of Docker nodes.
* Each node of a Docker Swarm is a Docker daemon, and all Docker daemons interact using the Docker API.
* Each container within the Swarm can be deployed and accessed by nodes of the same cluster.
* Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host,

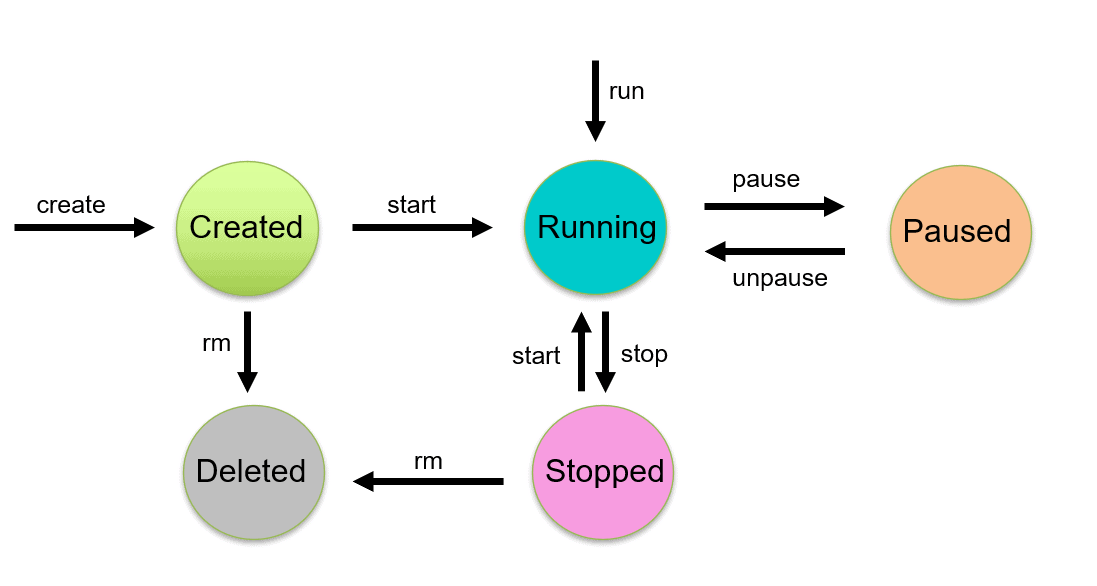
1. **What is a Docker Namespace?**

* A namespace is one of the Linux features and an important concept of containers.
* Namespace add a layer of isolation in containers.
* Docker uses a technology called namespaces to provide the isolated workspace called the *container*. When you run a container, Docker creates a set of *namespaces* for that container.

1. **What is the lifecycle of a Docker container?**

* The complete lifecycle of a docker container revolves around five phases.

1. Create phase
2. Running phase
3. Paused phase/unpause phase
4. Stopped phase
5. Killed Phase



* **Create containers:** $ docker create - - name <container name> <image name>
* **Start container:** $ docker start <container id/ container name>
* **Run container:** $ docker run –it - - name <container name> <image name>
* **Pause container:** $ docker pause <container name>

**To unpause :** $ docker unpause <container name>

* **Stop container: $ docker stop <container name>**

**To stop running container:** $ docker stop $(docker container ls –aq)

* **Delete container:** $ docker rm <container name>

**We can delete all stop container:** $ docker rm $(docker ps –aq)

* **Kill container**: $ docker kill <container name>

1. **What is docker Machine?**

* Docker machine is a tool that lets you install Docker Engine on virtual hosts.
* **Docker Machine** is a tool for provisioning and managing your Dockerized hosts (hosts with Docker Engine on them).
* Docker Machine has its own command line client **docker-machine** and the**Docker Engine client, docker.**

1. **How to check for Docker Client and Docker Server version?**

* The following command gives you information about Docker Client Server versions.

$ docker version

* Using this you can check all the versions present in docker engine

1. **How do you get the number of containers running, paused and stopped?**

* You can use the following command to get detailed information about the docker installed on your system.

$ docker info

1. **If you vaguely remember the command and you’d like to confirm it, how will you get on that particular command?**

* The following command is very useful as its gives you help on how to use a command, the syntax, etc.

$ docker – help

e.g. $ docker - - run help

1. **How to login into docker repository?**

* You can use the following command to login into hub.docker.com:

$docker login

* After type above command you will fill the username and password of docker hub for login into it.

1. **If you wish to use base image and make modifications or personalize it, how do you do that?**

* It’s one simple command to pull an image from docker hub:
* $ docker pull <image name>
* To get the list of image go to hub.docker.com website and pull the image by taking the image name
* E.g. $ docker pull Ubuntu:latest – In this command we pull the Ubuntu image form docker hub

1. **How to create a docker container from an image?**

* Use the following command:
* $ docker run –it – d <image name>
* E.g. $ docker run – it – d Ubuntu:latest

1. **How do you list all the running containers?**

* The following command lists down all the running containers:

$ docker ps

$ docker container ls

1. **How do you access a running container?**

* The following command let us access a running container:
* $ docker exec –it <container id > bash
* E.g. $ docker exec – it c6f bash

1. **How to start, stop and kill a container?**

* Here are below commands

$ docker start <container id>

$ docker stop <container id>

$ docker kill <container id>

1. **Can you use a container, edit it, and update it?**

* Here the command syntax:
* $ docker commit <container id> <username/imagename>
* $ docker commit c6f80324d ubuntu

1. **Once you have worked with an image, how do you push it to docker hub?**

* The following command as below
* $ docker push <username/image name>

1. **How to delete a stopped container?**

* Here the below command:
* $ docker rm <container id>

1. **How to delete an image from the local storage system?**

* Here the below command:
* $ docker rmi <image id>

1. **How to build a Dockerfile?**

* Once you have written a Dockerfile, you need to build it to create an image with those specifications. Use the following command to build a Dockerfile.
* $ docker build <path to docker file>

1. **Do you know why docker system prune is used? What does it do?**

* The basic usage of the command docker system prune is Remove unused data.
* Removes all unused containers, networks, images (both dangling and unreferenced), and optionally, volumes.
* $ docker system prune
* **Refer link:** https://docs.docker.com/engine/reference/commandline/system\_prune/

1. **Will you lose your data, when a docker container exists?**

* NO, you won’t lose any data when Docker container exists. Any data that your application writes to the container gets preserved.
* You have to explicitly delete the data

1. **Where all do you think Docker is being used?**

Docker is being used in the following area.

* **Simplifying configuration:** Docker lets you put your environment and configuration into code and deploy it.
* **Code Pipeline Management:** There are different systems used for development and production. As the code travels from development to testing to production, it goes through a difference in the environment. Docker helps in maintaining the code pipeline consistency.
* **Developer Productivity**: Using Docker for development gives us two things – We’re closer to production and development environment is built faster.
* **Application Isolation:** As containers are applications wrapped together with all dependencies, your apps are isolated. They can work by themselves on any hardware that supports Docker.
* **Debugging Capabilities:** Docker supports various debugging tools that are not specific to containers but work well with containers.
* **Multi-tenancy:** Docker lets you have multi-tenant applications avoiding redundancy in your codes and deployments.
* **Rapid Deployment:** Docker eliminates the need to boost an entire OS from scratch, reducing the deployment time.

1. **How is Docker different from other containerization methods?**

There are some differences between Docker and other container technologies or Docker alternatives. Here I'm sharing some of them.

* **Different configuration:** In the simple context, Docker is easy and its system configuration isn’t enriched like other container technologies like Lxc, runc, podman, etc. So from this aspect, you can identify the differences.
* Docker containers very easy to deploy.
* **Working Capacity:** Docker performs differently than other container-based management tools. Because some container management technologies work combinedly. But, docker works for isolated tasks.
* **Productivity:** Basically, Docker alternatives could be a wise decision for container-based applications. Because Docker’s productivity is not efficient and effective like others.
* Overall, Docker is a simple and different tool from others.

1. **How have you used Docker in your previous position?**

* This answer depend on your experience on Docker

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

* 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.
* Refer link for know about docker service scale : https://docs.docker.com/engine/reference/commandline/service\_scale/

1. **What platforms does docker run on?**

Docker runs on various Linux flavors:

* + - Ubuntu
    - RHEL
    - CENTOS 6 +
    - ArchLinux

**Refer links:** https://docs.docker.com/engine/faq/

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

* Using docker commands identify the status of Docker container.
* Such commands are : $ docker ps and $docker ps -a

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

* No, Best idea to remove container is first stop it and then remove the container

1. **Can a container restart by itself?**

No, it’s not possible for a container to restart by itself. By default the flag -restart is set to false.

1. **Is it better to directly remove the container using the rm command or stop the container followed by remove container?**

* It is better to stop the container and then remove it using remove command.

*$ docker stop <container id>*

*$ docker rm <container id>*

1. **Will cloud overtake the use of containerization?**
2. **How many containers can run per host?**

* Depend on your hardware of your system and containers need like storage space, CPU and memory, we can run the containers.
* It can be multiple numbers of containers or depends on your application requirement.

1. **Is it good practice to run stateful applications on Docker?**
2. **How will you monitor Docker in Production?**
3. **Have you used Kubernetes? If you have which one would you prefer amongst Docker and Kubernetes?**
4. **Are you aware of load balancing across containers and hosts? How does it work?**

# ****47 How Will you reduce the size of the Docker image?****

Using the official node alpine image as a base image, is a simple solution to reduce the overall size of the image, because even the base alpine image is a lot smaller compared to the base ubuntu image.

48.How to create a docker image if no internet connectivity is there

1. pull the image on a machine with internet access. $docker pull hello-world.
2. save that image to a . tar file. $ docker save --output hello-world.tar {your image name or ID}
3. copy that file to any machine.
4. load the . tar file to docker. $docker load --input hello-world.tar.

How to create zip file in docker container?

**The steps are:**

1. Prepare file Dockerfile. with the following lines: ...
2. Prepare a zip file called image. zip with some files.
3. Run command: docker build -t test3 .
4. At this point the image is built. image. ...
5. Run the container: docker run --rm -it test3 powershell.
6. From the container powershell run: dir.

**What is docker volume :**

Docker volumes are a widely used and useful tool for ensuring data persistence while working in containers. Docker volumes are file systems mounted on Docker containers to preserve data generated by the running container.

# 45. Suppose you have an application that has many dependant services. Will docker-compose wait for the current container to be ready to move to the running of the next service?

The answer is yes. Docker-compose always runs in the dependency order. These dependencies are specifications like depends\_on, links, volumes\_from, etc.

# 46. How will you monitor Docker in production?

Docker provides functionalities like docker stats and docker events to monitor docker in production. Docker stats provides CPU and memory usage of the container. Docker events provide information about the activities taking place in the docker daemon.

Q.Have you worked on multi-stage dockerfile and why we need that?

ANS:FROM maven as maven

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

Q. can you copy a file form local to run container?

ANS:docker cp filename containerid:path(where files needs to copy)

Q.Command to list conatiners which state is exited?

->docker ps -a -f status=running

docker ps -a -f status= exited

Q.command to clean-up docker host ( deleting stopped conatiners, dangling images and unused networks)?

->docker system prune

Q.Can we have multiple CMD in dockerfile ?

->There can only be one CMD instruction in a Dockerfile. If you list more than one CMD then only the last CMD will take effect

Q. Useful docker commands:

docker build -t rajakumargupta/websitevisitcount:latest . # Create image using this directory's Dockerfile

docker run -p 4000:80 rajakumargupta/websitevisitcount:latest # Run "friendlyhello" mapping port 4000 to 80

docker run -d --name mywebapp -p 4000:80 rajakumargupta/websitevisitcount:latest # Same thing, but in detached mode and container name of your choice

docker container ls # List all running containers

docker container ls -a # List all containers, even those not running

docker container stop <Container-hash> # Gracefully stop the specified container

docker container kill <Container-hash> # Force shutdown of the specified container

docker container rm <Container-hash> # Remove specified container from this machine

docker container rm $(docker container ls -a -q) # Remove all containers

docker logs <Container-hash> # See logs of container

docker exec -it <Container-hash> /bn/bash # Connect to a container via running /bin/bash inside that

docker image ls -a # List all images on this machine

docker images # List all images on this machine

docker image rm <image id> # Remove specified image from this machine

docker image rm $(docker image ls -a -q) # Remove all images from this machine

docker login # Log in this CLI session using your Docker credentials

docker tag <image> username/repository:tag # Tag <image> for upload to registry

docker push username/repository:tag # Upload tagged image to registry

docker run username/repository:tag # Run image from a registry

docker image ls # List all images (locally)

docker search <image-name> # Search an image in REPO

docker container ls # List all container

docker volume ls # List all volumes

docker network ls # List all network

docker image inspect <image-ID> # Get all info of images

docker container inspect <container-ID> # Get all info of container

docker volume inspect <volume-ID> # Get all info of volume

docker network inspect <network-ID> # Get all info of network

docker images --format "table {{.ID}}\t{{.Repository}}\t{{.Tag}}" # List images in tabular format having 3 columns ImageID, ImageRegistry and ImageTag

docker images --filter "dangling=true" # Filter using dangling

docker images --filter "label=com.example.version" # Filter using labels

docker images --filter "before=image1" # Filter using before

docker images --filter "since=image1" # Filter using since

Q.Explain docker project you have worked :

-><https://github.com/DeekshithSN/Micro_services/blob/main/micro-service-starter/docker_code/ui-web-app-reactjs/Dockerfile>

Q.what is the use of .dockerignore file ?

ANS:The . dockerignore file is helpful to avoid inadvertently sending files or directories that are large or contain sensitive files to the daemon or avoid adding them to the image using the ADD or COPY commands.

How to delete all stopped containers and unused images command for that?

->docker container prune

docker imgae prune to delete all images

and to delete unsued images- >docker rmi $ (docker images -aq )