**Docker and Container Interview Questions and Answers**

1. **What is Docker?**

Docker is an open-source lightweight containerization technology. Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.

1. **What is Container?**

A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another.

Put simply, a container consists of an entire runtime environment: an application, plus all its dependencies, libraries and other binaries, and configuration files needed to run it, bundled into one package. By containerizing the application platform and its dependencies, differences in OS distributions and underlying infrastructure are abstracted away.

1. **What's the difference between containers and virtualization?**

With virtualization technology, the package that can be passed around is a virtual machine, and it includes an entire operating system as well as the application. A physical server running three virtual machines would have a hypervisor and three separate operating systems running on top of it.

By contrast a server running three containerized applications with Docker runs a single operating system, and each container shares the operating system kernel with the other containers. Shared parts of the operating system are read only, while each container has its own mount (i.e., a way to access the container) for writing. That means the containers are much more lightweight and use far fewer resources than virtual machines.

1. **What other benefits do containers offer?**
2. A container may be only tens of megabytes in size, whereas a virtual machine with its own entire operating system may be several gigabytes in size. Because of this, a single server can host far more containers than virtual machines.
3. Another major benefit is that virtual machines may take several minutes to boot up their operating systems and begin running the applications they host, while containerized applications can be started almost instantly. That means containers can be instantiated in a "just in time" fashion when they are needed and can disappear when they are no longer required, freeing up resources on their hosts.
4. A third benefit is that containerization allows for greater modularity. Rather than run an entire complex application inside a single container, the application can be split in to modules (such as the database, the application front end, and so on). This is the so-called microservices approach. Applications built in this way are easier to manage because each module is relatively simple, and changes can be made to modules without having to rebuild the entire application. Because containers are so lightweight, individual modules (or microservices) can be instantiated only when they are needed and are available almost immediately.
5. **Security:** Docker makes sure that applications that are running on containers are completely segregated and isolated from each other, from a security point of view, by granting us complete control over traffic flow and management.
6. **What are the main drawbacks of Docker?**
7. **Missing features:** There are a ton of feature requests are under progress, like container self-registration, and self-inspects, copying files from the host to the container, and many more.
8. **Data in the container:** There are times when a container goes down, so after that, it needs a backup and recovery strategy, although we have several solutions for that they are not automated or not very scalable yet.
9. **Doesn’t provide cross-platform compatibility:** The one major issue is if an application designed to run in a Docker container on Windows, then it can’t run on Linux or vice versa. However, Virtual machines are not subject to this limitation. So, this limitation makes Docker less attractive in some highly heterogeneous environments which are composed of both Windows and Linux servers.
10. **Docker basics, good document.**

<https://docker-curriculum.com/>

1. **What is Docker Image?**

A Docker image is a file, comprised of multiple layers, that is used to execute code in a Docker container. An image is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel. When the Docker user runs an image, it can become one or multiple instances of that container.

1. **What is Docker Engine?**

Docker Engine is the underlying client-server technology that builds and runs containers using Docker's components and services. When people refer to Docker, they mean either Docker Engine -- which comprises the Docker daemon, a REST API and the CLI that talks to the Docker daemon through the API -- or the company Docker Inc., which offers various editions of containerization technology around Docker Engine. Docker Engine supports the tasks and workflows involved to build, ship and run container-based applications.

1. **What command should you run to see all running container in Docker?**

docker ps

1. **Write the command to stop the docker container**

sudo docker stop <container name/id>

1. **What is the command to run the image as a container?**

sudo docker run -i -t alpine /bin/bash

or

sudo docker run -it alpine sh

1. **What is memory-swap flag?**

By default, a container has no resource constraints and can use as much of a given resource as the host’s kernel scheduler allows. Docker provides ways to control how much memory, or CPU a container can use.

It is important not to allow a running container to consume too much of the host machine’s memory. On Linux hosts, if the kernel detects that there is not enough memory to perform important system functions, it throws an OOME, or Out of Memory Exception, and starts killing processes to free up memory. Docker can enforce hard memory limits, which allow the container to use no more than a given amount of user or system memory, or soft limits.

Most of options take a positive integer, followed by a suffix of b, k, m, g, to indicate bytes, kilobytes, megabytes, or gigabytes.

**-m or --memory=** The maximum amount of memory the container can use. If you set this option, the minimum allowed value is 4m (4 megabyte).

**--memory-swap\*** The amount of memory this container is allowed to swap to disk. Total memory limit (memory + swap

**--memory-swap** is a modifier flag that only has meaning if **--memory** is also set. Using swap allows the container to write excess memory requirements to disk when the container has exhausted all the RAM that is available to it.

1. **What is virtualization?**

Virtualization is the process of creating a software-based virtual version of something(compute storage, servers, application, etc.). These virtual versions or environments are call virtual machines and created from a single physical hardware system.

1. **What is containerization?**

So basically, an application that is being developed and deployed is bundled and wrapped together with all its configuration files and dependencies. This bundle is called a container.