**1. What is Docker?**

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.

Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined

**2. What is Docker image?**

A **Docker image** is a file, comprised of multiple layers, that is used to execute code in a **Docker container**. ...

When the **Docker** user runs an **image**, it can become one or multiple instances of that **container**. **Docker** is an open source OS-level virtualization software platform primarily designed for Linux.

**3. What is Docker container?**

Container management is a thin virtualization (only application libbers can have remaining are taken from host OS) here containers means vm. Container has only one application, it will take the required files

* Host os, it will use the kernel of host os.
* There is no OS in container. Only liberies and binary files remaining taken from host os
* It will run only application related files.

**4. What is Docker hub?**

Docker hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker cloud so you can deploy images to your hosts. It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

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

Docker containers are easy to deploy in a cloud. It can get more applications running on the same hardware than other technologies, it makes it easy for developers to quickly create, ready-to-run containerized applications and it makes managing and deploying applications much easier. You can even share containers with your applications.

**6. What is Docker Swarm?**

Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts.

**7. What is Dockerfile used for?**

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.

**8.** **Can I use json instead of yaml for my compose file in Docker?**

You can use json instead of yaml for your compose file, to use json file with compose, specify the filename to use for Eg:  
**docker-compose -f docker-compose.json up**

**9. Tell us how you have used Docker in your past position?**

Explain how you have used Docker to help rapid deployment. Explain how you have scripted Docker and used Docker with other tools like Puppet, Chef or Jenkins.

If you have no past practical experience in Docker and have past experience with other tools in a similar space, be honest and explain the same. In this case, it makes sense if you can compare other tools to Docker in terms of functionality.

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

This command will create and start a container.

docker run -t -i command name

To check the list of all running container with the status on a host:

docker ps -a

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

In order to stop the Docker container:

Docker stop container ID

To restart the Docker container:

Docker restart container ID

**12. How far do Docker containers scale?**

Large web deployments like Google and Twitter, and platform providers such as Hurok and dot Cloud all run on container technology, at a scale of hundreds of thousands or even millions of containers running in parallel.

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

Docker runs on only Linux and Cloud platforms and then I will mention the below

**Vendors of Linux:**

* Ubuntu 12.04, 13.04
* Fedora 19/20+
* RHEL 6.5+
* CentOS 6+
* Gentoo
* Arch Linux
* Opensuse 12.3+
* CRUX 3.0+

**Cloud**:

* Amazon EC2
* Google Compute Engine
* Microsoft Azure
* Rackspace

**Note that Docker does not run on Windows or Mac.**

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

No I won’t lose my data when Docker container exits, any data that your application writes to disk gets preserved in its container until you explicitly delete the container. The file system for the container persists even after the container halts.

**15. Is Container technology new?**

No, it is not. Different variations of containers technology were out there in \*NIX world for a long time. Examples are:-Solaris container (aka Solaris Zones)-FreeBSD Jails-AIX Workload Partitions (aka WPARs)-Linux Openvz

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

Docker can run on any infrastructure, you can run Docker on your laptop or you can run it in the cloud.

Docker has a Container HUB, it is basically a repository of containers which you can download and use. You can even share containers with your applications.

Docker is quite well documented.

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

Docker container is the runtime instance of docker image.

A **Docker image** is an immutable (unchangeable) file that contains the source code, libraries, dependencies, tools, and other files needed for an application to run

**Docker container** is a virtualized run-time environment where users can isolate applications from the underlying system. These containers are compact, portable units in which you can start up an application quickly and easily.

**18. What is the use case for Docker?**

Well, I think, docker is extremely useful in development environments. Especially for testing purposes. You can deploy and re-deploy apps in a blink of eye.

Also, I believe there are use cases where you can use Docker in production. Imagine you have some Node.js application providing some services on web. Do you really need to run full OS for this?

**19. How exactly containers (Docker in our case) are different from hypervisor virtualization (vSphere)? What are the benefits?**

To run an application in virtualized environment (e.g. vSphere), we first need to create a VM, install an OS inside and only then deploy the application. To run same application in docker all you need is to deploy that application in Docker. There is no need of additional OS layer. You just deploy the application with its dependent libraries, the rest (kernel, etc.) is provided by Docker engine. This table from a Docker official website shows it in a quite clear way.

**20. How to know the container status?**

Just fire**docker ps –a** to list out all running container with status (running or stopped) on a host

**21. How to stop and restart the container?**

To stop container, we can use docker stop <container id>

To start a stopped container, **docker start** <container id>is the command

To restart a running container, **docker restart** <container id>

**22. Docker is the new craze in virtualization and cloud computing. Why are people so excited about it?**

I think it’s the lightweight nature of Docker combined with the workflow. It’s fast, easy to use and a developer-centric DevOps tool. Its mission is basically make it easy to package and ship code. Developers want tools that abstract away a lot of the details of that process. They just want to see their code working. That leads to all sorts of conflicts with Sys Admins when code is shipped around and turns out not to work somewhere other than the developer’s environment. Docker turns to work around that by making your code as portable as possible and making that portability user friendly and simple.

**24. What, in your opinion, is the most exciting potential use for Docker?**

It’s definitely the build pipeline. I mean I see a lot of folks doing hyper-scaling with containers, indeed you can get a lot of containers on a host and they are blindingly fast. But that doesn’t excite me as much as people using it to automate their dev-test-build pipeline.

**25. How is Docker different from standard virtualization?**

Docker is operating system level virtualization. Unlike hypervisor virtualization, where virtual machines run on physical hardware via an intermediation layer (“the hypervisor”), containers instead run user space on top of an operating system’s kernel. That makes them very lightweight and very fast.

**26. Do you think cloud technology development has been heavily influenced by open source development?**

I think open source software is closely tied to cloud computing. Both in terms of the software running in the cloud and the development models that have enabled the cloud. Open source software is cheap, it’s usually low friction both from an efficiency and a licensing perspective.

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

Compose uses the project name to create unique identifiers for all of a project’s containers and other resources. To run multiple copies of a project, set a custom project name using the -p command line option or the COMPOSE\_PROJECT\_NAME environment variable.

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

Typically, you want docker-compose up. Use up to start or restart all the services defined in a docker-compose.yml. In the default “attached” mode, you’ll see all the logs from all the containers. In “detached” mode (-d), Compose exits after starting the containers, but the containers continue to run in the background.

The docker-compose run command is for running “one-off” or “adhoc” tasks. It requires the service name you want to run and only starts containers for services that the running service depends on. Use run to run tests or perform an administrative task such as removing or adding data to a data volume container.

The run command acts like docker run -ti in that it opens an interactive terminal to the container and returns an exit status matching the exit status of the process in the container.

The docker-compose start command is useful only to restart containers that were previously created, but were stopped. It never creates new containers.

**29. Can I use json instead of yaml for my Compose file?**

Yes. Yaml is a superset of json so any JSON file should be valid Yaml. To use a JSON file with Compose, specify the filename to use, for example:

docker-compose -f docker-compose.json up

**30. Should I include my code with COPY/ADD or a volume?**

You can add your code to the image using COPY or ADD directive in a Dockerfile. This is useful if you need to relocate your code along with the Docker image, for example when you’re sending code to another environment (production, CI, etc).

You should use a volume if you want to make changes to your code and see them reflected immediately, for example when you’re developing code and your server supports hot code reloading or live-reload.

There may be cases where you’ll want to use both. You can have the image include the code using a COPY, and use a volume in your Compose file to include the code from the host during development. The volume overrides the directory contents of the image.

**32. How does Docker help manage my infrastructure? Do I containerize all my infrastructure or something?**

Docker isn’t focused on managing your infrastructure. The platform, which is infrastructure agnostic, manages your applications and helps ensure that they can run smoothly, regardless of infrastructure type via solutions like Docker Datacentre. This gives your company the agility, portability and control you require. Your team is responsible for managing the actual infrastructure.

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

As far as the number of containers that can be run, this really depends on your environment. The size of your applications as well as the amount of available resources (i.e. like CPU) will all affect the number of containers that can be run in your environment. Containers unfortunately are not magical. They can’t create new CPU from scratch.

The typical organization that uses a container orchestrator runs **11.5 containers** per host, as compared to about **6.5 containers** per host in unorchestrated environments