**Docker Interview Questions And Answers**

**Interview Questions On Docker**

**What is Docker?**

Docker is a platform to run each application isolated and securely. Internally it achieves it by using kernel containerization feature.

**What is the advantage of Docker over hypervisors?**

Docker is lightweight and more efficient in terms of resource uses because it uses the underlying host kernel rather than creating its hypervisor.

**What is Docker Container?**

[Docker Container](http://tekslate.com/tutorials/docker) is the instantiation of docker image. In other words, it is the run time instance of images. Images are set of files whereas containers are the one who run the image inside isolated.

**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](http://en.wikipedia.org/wiki/Solaris_Containers))-FreeBSD Jails-AIX Workload Partitions (aka WPARs)-Linux [OpenVZ](http://openvz.org/" \o "OpenVZ" \t "_blank)

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

Well, Docker is a quite fresh project. It was created in the Era of Cloud, so a lot of things are done much nicer than in other container technologies. Team behind Docker looks to be full of enthusiasm, which is of course very good. I am not going to list all the features of Docker here, but I will mention those which are important to me.

Docker can run on any infrastructure, you can run [docker on your laptop](https://tekslate.com/docker-training/), or you can run it in the cloud.

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

Docker is quite well documented.

**What are the networks that are available by default?**

|  |  |
| --- | --- |
| **bridge** | It is the default network all containers connect to if you don’t specify the network yourself |
| **none** | connects to a container-specific network stack that lacks a network interface |
| **host** | connects to the host’s network stack - there will be no isolation between the host machine and the container, as far as network is concerned |

**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.

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

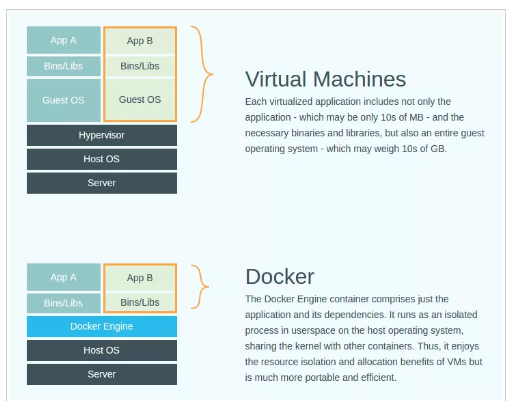
Well, I think, docker is extremely useful in development environments. Especially for testing purposes. You can deploy and re-deploy apps in a blink of an eye.

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 the web. Do you need to run full OS for this?

Eventually, if docker is good or not should be decided on an application basis. For some apps, it can be sufficient, for others not.

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

To run an application in a virtualized environment (e.g., vSphere), we first need to create a VM, install an OS inside and only then deploy the application. To run the same application in docker, all you need is to 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.This table from a Docker official website shows it in a quite clear way.



Another benefit of Docker, from my perspective, is speed of deployment. Let's imagine a scenario:

ACME inc. needs to virtualize application GOOD APP for testing purposes.

Conditions are:

Application should run in an isolated environment.

Application should be available to be redeployed at any moment in a very fast manner.

***Solution 1.***

In vSphere 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.

***Solution 2.***

* Deploy Docker.
* Deploy the app GOODAPP in container.
* Redeploy the container with an app when needed.

***Benefits:*** No need of deploying full OS for each instance of the application. Deploying a container takes seconds.

**How did you become involved with the Docker project?**

I came across Docker not long after Solomon open sourced it. I knew a bit about LXC and containers (a past life includes working on Solaris Zones and LPAR on IBM hardware too), and so I decided to try it out. I was blown away by how easy it was to use. My prior interactions with containers had left me with the feeling they were complex creatures that needed a lot of tuning and nurturing. Docker just worked out of the box. Once I saw that and then saw the CI/CD-centric workflow that Docker was building on top I was sold.

[Docker Interview Questions](https://tekslate.com/tutorials/docker)

**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-ish 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.

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

It's 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.

**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 inter-mediation layer ("the hypervisor"), containers instead run userspace on top of an operating system's kernel. That makes them very lightweight and very fast.

**Docker Container Interview Questions And Answers For Experienced**

**Do you think open source development has heavily influenced cloud technology 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.

**How do you think Docker will change virtualization and cloud environments? Do you think cloud technology has a set trajectory, or is there still room for significant change?**

I think there are a lot of workloads that Docker is ideal for, as I mentioned earlier both in the hyper-scale world of many containers and in the dev-test-build use case. I fully expect a lot of companies and vendors to embrace Docker as an alternative form of virtualization on both bare metal and in the cloud.

As for cloud technology's trajectory. I think we've seen a significant change in the last couple of years. I think they'll be a bunch more before we're done. The question of OpenStack and whether it will succeed as an IAAS alternative or DIY cloud solution. I think we've only touched on the potential for PAAS and there's a lot of room for growth and development in that space. It'll also be interesting to see how the capabilities of PAAS products develop and whether they grow to embrace or connect with consumer cloud-based products.

**Can you give us a quick rundown of what we should expect from your Docker presentation at OSCON this year?**

It's very much a crash course introduction to Docker. It's aimed at Developers and SysAdmins who want to get started with Docker in a very hands on way. We'll teach the basics of how to use Docker and how to integrate it into your daily workflow.

**Your bio says "for a real job" you're the VP of Services for Docker. Do you consider your other open source work a hobby?**

That's mostly a joke related to my partner. Like a lot of geeks, I'm often on my computer, tapping away at a problem or writing something. My partner jokes that I have two jobs: my "real" job and my open source job. Thankfully over the last few years, at places like Puppet Labs and Docker, I've been able to combine my passion with my paycheck.

[Docker Interview Questions](https://tekslate.com/docker-training/)

**Why is Docker the new craze in virtualization and cloud computing?**

It's OSCON time again, and this year the tech sector is abuzz with talk of cloud infrastructure. One of the more interesting startups is Docker, an ultra-lightweight containerization app that's brimming with potential

I caught up with the VP of Services for Docker, James Turnbull, who'll be running a Docker crash course at the con. Besides finding out what Docker is anyway, we discussed the cloud, open source contributing and getting a real job.

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

Compose stop attempts to stop a container by sending a SIGTERM. It then waits for a default timeout of 10 seconds. After the timeout, a SIGKILL is sent to the container to kill it forcefully. If you are waiting for this timeout, it means that your containers aren’t shutting down when they receive the SIGTERM signal.

There has already been a lot written about this problem of processes handling signals in containers.

To fix this issue, try the following:

Make sure you’re using the JSON form of CMD and ENTRYPOINT in your [Dockerfile](http://tekslate.com/save-docker/" \t "_blank).

For example use ["program", "arg1", "arg2"] not"program arg1 arg2". Using the string form causes Docker to run your process using bash which doesn’t handle signals properly. Compose always uses the JSON form, so don’t worry if you override the command or entrypoint in your Compose file.

-If you are able, modify the application that you’re running to add an explicit signal handler for SIGTERM.

-Set the stop\_signal to a signal which the application knows how to handle:

-web: build: . stop\_signal: SIGINT

-If you can’t modify the application, wrap the application in a lightweight init system (like s6) or a signal proxy (like dumb-init or tini). Either of these wrappers take care of handling SIGTERM properly.

**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 theCOMPOSE\_PROJECT\_NAME environment variable.

**Docker Container Interview Questions**

**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 “ad-hoc” 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](http://tekslate.com/docker-commands/) 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.

**What is the base image required to build a Docker container?**

The base image depends on the tool or script. You can pursue available images by searching Docker Hub for the domain (e.g., “biology”, “science”) and read the documentation for specific images.

*Most frequently used CyVerse base images:*

ubuntu:12.04

ubuntu:14.04

**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

**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 the 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.

**Where can I find example compose files?**

There are many examples of Compose files on github.

**Compose documentation**

* Installing Compose
* Get started with Django
* Get started with Rails
* Get started with WordPress
* Command line reference
* Compose file reference

**Are you operationally prepared to manage multiple languages/libraries/repositories?**

Last year, we encountered an organization that developed a modular application while allowing developers to “use what they want” to build individual components. It was a nice concept but a total organizational nightmare — chasing the ideal of modular design without considering the impact of this complexity on their operations.

The organization was then interested in Docker to help facilitate deployments, but we strongly recommended that this organization not use Docker before addressing the root issues. Making it easier to deploy these disparate applications wouldn’t be an antidote to the difficulties of maintaining several different development stacks for long-term maintenance of these apps.

**Do you already have a logging, monitoring, or mature deployment solution?**

Chances are that your application already has a framework for shipping logs and backing up data to the right places at the right times. To implement Docker, you not only need to replicate the logging behavior you expect in your virtual machine environment, but you also need to prepare your compliance or governance team for these changes. New tools are entering the Docker space all the time, but many do not match the stability and maturity of existing solutions. Partial updates, rollbacks, and other common deployment tasks may need to be re-engineered to accommodate a containerized deployment.

If it’s not broken, don’t fix it. If you’ve already invested the engineering time required to build a continuous integration/continuous delivery (CI/CD) pipeline, containerizing legacy apps may not be worth the time investment.

**Docker Questions And Answers**

**Will cloud automation overtake containerization?**

At AWS Re:Invent last month, Amazon chief technology officer Werner Vogels spent a significant portion of his keynote on AWS Lambda, an automation tool that deploys infrastructure based on your code. While Vogels did mention AWS’ container service, his focus on Lambda implies that he believes dealing with zero infrastructure is preferable to configuring and deploying containers for most developers.

Containers are rapidly gaining popularity in the enterprise, and are sure to be an essential part of many professional CI/CD pipelines. But as technology experts and CTOs, it is our responsibility to challenge new methodologies and services and properly weigh the risks of early adoption. I believe Docker can be extremely effective for organizations that understand the consequences of containerization — but only if you ask the right questions.

**You say that Ansible can take up to 20x longer to provision, but why?**

Docker uses cache to speed up builds significantly. Every command in Dockerfile is building in another docker container, and its results are stored in a separate layer. Layers are built on top of each other.

Docker scans Dockerfile and try to execute each steps one after another, before executing it probes if this layer is already in cache. When a cache is hit, building step is skipped, and from the user perspective is almost instant.

When you build your Dockerfile in a way that the most changing things such as application source code are on the bottom, you will experience instant builds.

Another way of amazingly fast building docker images is using a good base image - which you specify inFROM command, you can then only make necessary changes, not rebuild everything from scratch. This way, the build will be quicker. It's especially beneficial if you have a host without the cache like Continuous Integration server.

Summing up, building Docker images with Dockerfile is faster than provisioning with Ansible, because of using docker cache and good base images. Moreover, you eliminate provisioning, by using ready to use configured images such stgresus.

$ docker run --name some-postgres -d postgres No installing postgres at all - it's ready to run.

**Also, you mention that docker allows multiple apps to run on one server.**

It depends on your use case. You probably should split different components into separate containers. It will give you more flexibility.

Docker is very lightweight and running containers is cheap, especially if you store them in RAM - it's possible to spawn new container for every http callback, however, it's not very practical.

At work, I develop using a set of five different types of containers linked together.

In production some of them are replaced by real machines or even clusters of machine - however, settings on application level don't change.

It's possible because everything is communicating over the network. When you specify links in dockerrun command - docker bridges containers and injects environment variables with information about IPs and ports of linked children into the parent container.

This way, in my app settings file, I can read those values from the environment. In python it would be:

import os VARIABLE = os.environ.get('VARIABLE')

There is a tool which greatly simplifies working with docker containers, linking included. It's called fig, and you can read more about it here.

Finally, what does the deploy process look like for dockerized apps stored in a git repo?

It depends on how your production environment looks like.

Example deploy process may look like this:

* Build an app using docker build . in the code directory.
* Test an image.
* Push the new image out to [registry](http://tekslate.com/docker-registry/) docker push myorg/myimage.
* Notify remote app server to pull image from registry and run it (you can also do it directly using some configuration management tool).
* Swap ports in a http proxy.
* Stop the old container.

You can consider using amazon elastic beanstalk with docker or dokku.

Elastic beanstalk is a powerful beast and will do most of the deployment for you and provide features such as auto-scaling, rolling updates, zero deployment deployments and more.

Dokku is a very simple platform as a service similar to heroku.

How is Docker different from other container technologies?

Docker containers are easy to deploy in the cloud. It is capable of getting more applications running on the same hardware when compared with other technologies like Kubernete, Amazon Elastic Contain, etc. Thus making learners who take [Kubernetes Training Hyderabad](https://tekslate.com/kubernetes-training/) and developers create, ready-to-run containerized applications and make them manage, deploy and share easily.

**Mention some commonly used Docker Commands?**

Some among the most commonly used Docker Commands are as follows:

|  |  |
| --- | --- |
| Command | Description |
| Dockerd | Launch the Docker Daemon |
| Info | Displays information System-Wide |
| Version | Displays the Docker Version information |
| Build | Builds images for Docker files |
| Inspect | Returns low-level information on an image or container |
| History | Shows Image History |
| Commit | Creates new images from Container changes |
| Attach | Attaches to a running container |
| Load | Load an image from STDIN or tar archive |
| Create | Create a new container |
| Diff | Inspect changes on a container’s file system |
| Kill | Kill a running container |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Virtual Machines Vs Docker Containers** | | | |
| **Virtual Machines** | **Docker Containers** |  |  |
| Need more resources | Less resources are used |  |  |
| Process isolation is done at hardware level | Process Isolation is done at Operating System level |  |  |
| Separate Operating System for each VM | Operating System resources can be shared within Docker |  |  |
| VMs can be customized. | Custom container setup is easy |  |  |
| Takes time to create Virtual Machine | Creation of docker is very quick |  |  |
| Booting takes minutes | Booting is done within seconds |  |  |
| Related Article : [***Kubernetes Vs Docker Swarn***](https://mindmajix.com/kubernetes-vs-docker-swarm) | |  |  |

**Q. What is Docker?**

Docker can be defined as Containerization platform that packs all your applications, all the necessary dependencies combined to form containers. This will not only ensure the applications work seamlessly given any environment but also provides better efficiency to your Production ready applications. Docker wraps up bits and pieces of software with all the needed filesystems containing everything that needs to run the code, provide the runtime, system tools / libraries. This will ensure that the software is always run and executed the same, regardless of the environment.

Containers run on the same machine sharing the same Operating system Kernel, this makes it faster – as starting the applications is the only time that is required to start your Docker container (remember that the OS Kernel is already UP and running and uses the least of the RAM possible).

**Q. What is the advantage of Docker over hypervisors?**

Docker is light weight and more efficient in terms of resource uses because it uses the host underlying kernel rather than creating its own hypervisor.

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

To start with, Docker is one of the upcoming and is a fresh project. Since its inception has been done in the Cloud era, it is way better many of the other competing Container technologies which have ruled their way until Docker came into existence. There is an active community that is running towards the better upbringing of Docker and it has also started extending its support to Windows and Mac OSX environments in the recent days. Other than these, below are the best possible reasons to highlight Docker as one of the better options to choose from than the existing Container technologies.

* There is no limitation on running Docker as the underlying infrastructure can be your laptop or else your Organization’s Public / Private cloud space
* Docker with its Container HUB forms the repository of all the containers that you are ever going to work, use and download. Sharing of applications is as well possible with the Containers that you create.
* Docker is one of the best-documented technologies available in the Containerization space.

**Q. What is Docker image?**

Docker image can be understood as a template from which Docker containers can be created as many as we want out of that single Docker image. Having said that, to put it in layman terms, Docker containers are created out of Docker images. Docker images are created with the build command, and this produces a container that starts when it is run. Docker images are stored in the Docker registry such as the public Docker registry (registry.hub.docker.com) as these are designed to be constituted with layers of other images, enabling just the minimal amount of data over the network.

**Q. What is Docker container?**

This is a very important question so just make sure you don’t deviate from the topic and I will advise you to follow the below mentioned format:

1. Docker containers include the application and all of its dependencies, but share the kernel with other containers, running as isolated processes in user space on the host operating system. Docker containers are not tied to any specific infrastructure: they run on any computer, on any infrastructure, and in any cloud.  
2. Now explain how to create a Docker container, Docker containers can be created by either creating a Docker image and then running it or you can use Docker images that are present on the Dockerhub.  
3. Docker containers are basically runtime instances of Docker images.

**Q. 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.

**Q. What is Docker Swarm?**

Docker Swarm can be best understood as the native way of Clustering implementation for Docker itself. Docker Swarm turns a pool of Docker hosts into a single and virtual Docker host. It serves the standard Docker[API](https://mindmajix.com/api-testing-training) or any other tool that can already communicate with a Docker daemon can make use of Docker Swarm to scale in a transparent way to multiple hosts. Following are the list of some of the supported tools that will be helpful in achieving what we have discussed just now.

• Dokku  
• Docker Compose  
• Docker Machine  
• Jenkins

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

Dockerfile is nothing but a set of instructions that have to be passed on to Docker itself, so that it can build images automatically reading these instructions from that specified 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.

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

This is a question that you could bring upon your whole experience with Docker and if you have used any other Container technologies before Docker. You could also explain the ease that this technology has brought in the automation of the development to production lifecycle management. You can also discuss about any other integrations that you might have worked along with Docker such as Puppet, Chef or even the most popular of all technologies – Jenkins. If you do not have any experience with Docker itself but similar tools from this space, you could convey the same and also show in your interest towards learning this leading containerization technology.

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

You can create a Docker container out of any specific Docker image of your choice and the same can be achieved using the command given below:  
docker run -t -i command 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?**

The following command can be used to stop a certain Docker container with the container id as

**CONTAINER\_ID:**  
docker stop CONTAINER\_ID  
The following command can be used to restart a certain Docker container with the container id as

**CONTAINER\_ID:**  
docker restart CONTAINER\_ID

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

Best examples in the Web deployments like Google, Twitter and best examples in the Platform Providers like Heroku, dotCloud run on Docker which can scale from the ranges of hundreds of thousands to millions of containers running in parallel, given the condition that the [OS](https://mindmajix.com/os-programming-and-administration-courses) and the memory doesn’t run out from the hosts which runs all these innumerable containers hosting your applications.

[Check Out Docker Tutorials](https://mindmajix.com/docker)

**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?**

There is no loss of data when any of your Docker containers exits as any of the data that your application writes to the disk in order to preserve it. This will be done until the container is explicitly deleted. The file system for the Docker container persists even after the Docker container is halted.

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

The most exciting potential use of Docker that I can think of is its build pipeline. Most of the Docker professionals are seen using hyper-scaling with containers, and indeed get a lot of containers on the host that it actually runs on. These are also known to be blatantly fast. Most of the development – test build pipeline is completely automated using the Docker framework.

**Q. Why is Docker the new craze in virtualization and cloud computing?**

Docker is the newest and the latest craze in the world of Virtualization and also Cloud computing because it is an ultra-lightweight containerization app that is brimming with potential to prove its mettle.

**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. 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.

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

Dockerizing enterprise environments helps teams to leverage over the Docker containers to form a service platform as like a CaaS (Container as a Service). It gives teams that necessary agility, portability and also lets them control staying within their own network / environment.

Most of the developers opt to use Docker and Docker alone because of the flexibility and also the ability that it provides to quickly build and ship applications to the rest of the world. Docker containers are portable and these can run on any environment without making any additional changes when the application developers have to move between Developer, Staging and Production environments. This whole process is seamlessly implemented without the need of performing any recoding activities for any of the environments. These not only helps reduce the time between these lifecycle states, but also ensures that the whole process is performed with utmost efficiency. There is every possibility for the Developers to debug any certain issue, fix it and also update the application with it and propagate this fix to the higher environments with utmost ease.

The operations teams can handle the security of the environments while also allowing the developers build and ship the applications in an independent manner. The CaaS platform that is provided by Docker framework can deploy on-premise and is also loaded with full of enterprise level security features such as role-based access control, integration with LDAP or any Active Directory, image signing and etc. Operations teams have heavily rely on the scalability provided by Docker and can also leverage over the Dockerized applications across any environments.

Docker containers are so portable that it allows teams to migrate workloads that run on an Amazon’s AWS environment to Microsoft Azure without even having to change its code and also with no downtime at all. Docker allows teams to migrate these workloads from their cloud environments to their physical datacenters and vice versa. This also enables the Organizations to focus on the infrastructure from the gained advantages both monetarily and also the self-reliability over Docker. The lightweight nature of Docker containers compared to traditional tools like virtualization, combined with the ability for Docker containers to run within VMs, allowing teams to optimize their infrastructure by 20X, and save money in the process.

**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 possibility to include specific code with COPY/ADD or a volume?**

This can be easily achieved by adding either the COPY or the ADD directives in your dockerfile. This will count to be useful if you want to move your code along with any of your Docker images, example, sending your code an environment up the ladder – Development environment to the Staging environment or from the Staging environment to the Production environment.

Having said that, you might come across situations where you’ll need to use both the approaches. 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.

**Q. Will cloud automation overtake containerization any sooner?**

Docker containers are gaining the popularity each passing day and definitely will be a quintessential part of any professional Continuous Integration / Continuous Development pipelines. Having said that there is equal responsibility on all the key stakeholders at each Organization to take up the challenge of weighing the risks and gains on adopting technologies that are budding up on a daily basis. In my humble opinion, Docker will be extremely effective in Organizations that appreciate the consequences of Containerization.

**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. What are the various states that a Docker container can be in at any given point in time?**

There are four states that a Docker container can be in, at any given point in time. Those states are as given as follows:

• Running  
• Paused  
• Restarting  
• Exited

**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.

Explore Docker Sample Resumes! Download & Edit, Get Noticed by Top Employers![**Download Now!**](https://mindmajix.com/docker-sample-resumes)

**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. If you want to tweak this, then you may give it a try.

**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 doesnot have a state and its state never changes as it is just set of files whereas docker container has its execution state.

## Top Answers to Docker Interview Questions

**1. What is Docker?**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Docker** | **Virtual Machines** |
| Use of OS | All containers share host OS | Each VM runs in its own OS |
| Startup time | Very fast | Slow |
| Isolation | Process level isolation | Full isolation |
| security | Low | High |

You can define Docker as a containerization platform that combines all your applications in a package so that you have all the dependencies to run your applications in any environment. This means your application will run seamlessly on any environment and this makes it easy for having a product ready application. What Docker does is wrap the software that is needed in a file system that has everything for running the code, providing the runtime and all the necessary libraries and system tools. Containerization technology like Docker will share the same operating system kernel with the machine and due to this it is extremely fast. This means that you have to run the Docker only at the beginning and after that since your OS is already running, you will have a smooth and seamless process.

**2. What is the benefit of using a Docker over a hypervisor?**

Though Docker and Hypervisor might do the same job overall there are many differences between them in terms of how they work. Docker can be thought of as light weight since it uses very less resources and also the host kernel rather than creating it like a Hypervisor.

Become DevOps Certified in 16 hrs.

**GET CERTIFIED**

**3. What are the unique features of Docker over other containerization technology?**

Here we list some of the most important and unique features of Docker that makes it a top containerization technology unlike any other in the market today

* You can run your Docker container either on your PC or your enterprise IT system
* Along with the Docker Hub which is a repository of all containers you can deploy and download all your applications from a central location
* You can even share your applications with the containers that you create.

**4. What is Docker image?**

Here we will be explaining what is the Docker image. The Docker image help to create the Docker containers. You can create the Docker image with the build command, due to this it creates a container that starts when it begins to run. All the docker images are stored in the Docker registry like the public docker registry. These have minimal amounts of layers within the image so that there is minimum amount of data on the network.

**5. What is Docker container?**

Here we will be discussing what is a Docker container. It is a comprehensive set of applications including all its dependencies which share the same OS kernel along with the other containers running in separate processes within the operating system in a user space. The Docker is not tied to any IT infrastructure and thus it can run on any computer system or the cloud. You can create a Docker container using the Docker images and then running it or you can use the images that are already created in the Docker Hub. To simplify things, let us say that the Docker containers are just runtime instances of the Docker image.

**6. What is Docker hub?**

You can think of Docker Hub as a cloud registry that lets you link the code repositories, create the images and test them. You can also store your pushed images, or you can link to the Docker Cloud, so that the images can be deployed to the host. You have a centralized container image discovery resource which can be used for collaboration of your teams, automating the workflow, distribution and change management by creating the development pipeline.

**7. What is Docker Swarm?**

You can think of Docker Swarm as the way of orchestrating the Docker containers. You will be able to implement the Dockers in a cluster. You can convert your Docker pools into a single Docker Swarm for easy management and monitoring.

**8. What is the use of Dockerfile?**

The Dockerfile can be thought of as a set of instructions that you need to pass on to the Docker so that the images can be built from the specified instructions in the Dockerfile. You can think of the Dockerfile as a text document which has all the commands that are needed for creating a Docker image. You can create an automated build that lets you execute multiple command-lines one after the other.

**9. Is it possible to use JSON instead of YAML for Docker compose?**

You can use JSON instead of YAML for Docker compose file. So when you are using the JSON file for composing then you have to specify the filename with the following command:

docker-compose -f docker-compose.json up

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

This is a question that you could bring upon your whole experience with Docker and if you have used any other Container technologies before Docker. You could also explain the ease that this technology has brought in the automation of the development to production lifecycle management. You can also discuss about any other integrations that you might have worked along with Docker such as Puppet, Chef or even the most popular of all technologies – Jenkins. If you do not have any experience with Docker itself but similar tools from this space, you could convey the same and also show in your interest towards learning this leading containerization technology.

[**Learn DevOps from Experts! Enrol Today**](https://intellipaat.com/devops-certification-training/#course-content)

**11. What is the process for creating a Docker container?**

You can use any of the specific Docker image for creating a Docker container using the below command.

docker run -t -i command name

This command not only creates the container but also will start it for you. If you want to check if the Docker container has been created or not then you need to have the following command which will list all the Docker containers along with the host on which the Docker container runs.

docker ps -a

**12. What is the process for stopping and restarting a Docker container?**

If you want to stop a Docker container then you need to use the following command:

**CONTAINER\_ID:**

docker stop CONTAINER\_ID

If you want to restart a Docker container then you need to use the following command

**CONTAINER\_ID:**

docker restart CONTAINER\_ID

**13. How do you scale your Docker containers?**

The Docker containers can be scaled to any level starting from a few hundreds to even thousands or millions of containers. The only condition is that the containers need the memory and the OS at all times and there should not be a constraint on these when the Docker is getting scaled.

**1. What are the main differences between Git and SVN?**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Git** | **SVN** |
| Type of version control | Distributed | Centralized |
| Access to networks | Not mandatory | Mandatory |
| Global revision number | Not available | Available |
| Content | Cryptographic SHA-1 Hash | No hashed content |

Here we list some of the most important differences between Git and SVN:

* When it comes to handling large files, Git is not preferred but SVN can handle multiple projects in the same repository
* Git does not have ‘commits’ across multiple branches but SVN lets you create the folders on any location in the repository layout
* You cannot commit changes in Git but SVN lets you create a tag as a branch and you can create multiple revisions under a root tag

**2. What are the advantages of using GIT?**

Here are some of the most important advantages of Git:

* Data redundancy and data replication is possible
* It is a highly available service
* For one repository you can have only one directory of Git
* The network performance and disk utilization are excellent
* It is very easy to collaborate on any project
* You can work on any sort of project within the Git

**3. What language is used in GIT?**

C is the programming language that is used for creating Git which ensures that the overheads are reduced.

Become DevOps Certified in 16 hrs.

**GET CERTIFIED**

**4. What are the advantages of Git over SVN?**

Since Git is an open source version control system it lets you run multiple versions of your project so that it shows the changes that are made to the code over time and if needed you can keep a track of the changes that you have made. This means that large number of developers can make their own changes and upload those changes so that the changes can be attributed to the particular developers.

**5. What is the meaning of “Index” or “Staging Area” in GIT?**

When you are making the commits you can make changes to it, format it and review it in the intermediate area known as ‘Staging Area’ or ‘Index’.

**6. What is GIT stash?**

The Git stash will take the working directory in the current state and index it to put on the stack at a later stage so that what you get is a clean working directory. This means that if you are in the middle of some task and need to get a clean working directory and simultaneously you want to keep all your current edits then you can use the Git stash.

**7. What is GIT stash drop?**

When you are finished with working on the stashed item or want to remove the list then you can use  the Git stash drop. This will ensure that the item that is last added by default or any particular item can be removed from the argument.

**8. How to identify if a certain branch has been merged into master?**

**Git branch –merged master**– shows all branches that are merged into master

**Git branch – merged**– shows all branches that are merged into the head

**Git branch – no-merged**–shows all the branches that are not merged

**9. What is the use of a Git clone?**

The Git clone command lets you copy the existing Git repository. If you want to get a copy of the central repository then the best way to do it is using ‘cloning’.

[**Learn DevOps from Experts! Enrol Today**](https://intellipaat.com/devops-certification-training/#course-content)

**10. What is the function of ‘git config’?**

The ‘Git config’ is a great way to configure your options for the Git installation. Using this command you can define the repository behavior, preferences and user information.

**11. What are the constituents of the commit object contain?**

* the state of a project at a given point of time is contained in a set of files
* Parent object commit references
* A 40-character string that uniquely identifies the commit object called a SHAI name

**12. What is the process for creating a repository in Git?**

If you want to create a repository in Git then you need to run the command “git init”. With this command .git repository you can create a directory in the project directory.

**13. What is ‘head’ in git and how many heads can be created in a repository?**

The reference to a commit object is called as the ‘head’. Every repository has a ‘Master’ which is the default head. There can be multiple heads in a repository.

**14. Why do we need branching in GIT?**

With the help of branching you can have your own branch and you can also jump between the various branches. You can go to your previous work while at the same time keeping your recent work intact.

**15. What is the regular way for branching in GIT?**

The best way to create a branch in GIT is to have one ‘main’ branch and then create another branch for implementing the changes that you want to make. This is extremely useful when there are a large number of developers working on a single project.

**16. State a way to create a new branch in Git?**

If you want to create a new feature into the main branch then you can use the command ‘git merge’ or ‘git pull’.

**17. How do you define a ‘conflict’ in git?**

If you want to merge a commit there is a change in one place and same change already exists then while merging the Git will not be able to predict which is the change that needs to be taken precedence.

**18. How to resolve a conflict in Git?**

If you want to resolve a conflict in Git then you need to edit the files for fixing the conflicting changes and then you can run “git add” to add the resolved files and after that you can run the ‘git commit’ for committing the repaired merge.

**1. What are the software prerequisites that must be met before Jenkins is installed?**

First let us compare Jenkins with TeamCity:

|  |  |  |
| --- | --- | --- |
| **Comparison** | **Jenkins** | **TeamCity** |
| Open Source | Yes | No |
| Default security | No | Yes |
| Individual validation | No | Yes |
| Popularity | Widely used | Not so widely used |

The software prerequisites for installing Jenkins is that first you need to install Java Development Kit. It also needs you to install the Jakarta Enterprise Edition. Jenkins also comes with an embedded Jetty runtime that can be used if WebSphere or Tomcat is not available.

**2. How to configure and use third-party tools in Jenkins?**

These are some of the steps used for working with a third-party tool in Jenkins.

1. You have to first install the third-party software
2. You need to have the plug-in that supports the third-party tool.
3. You have to configure the third-party tool in the admin console.
4. You can then use the plug-in from the Jenkins build job.

Become DevOps Certified in 16 hrs.

**GET CERTIFIED**

**3. Name two ways a Jenkins node agent can be configured to communicate back with the Jenkins master.**

These are the mechanisms for starting a Jenkins node agent:

1. From the browser window launch a Jenkins node agent.
2. From the command line launch a Jenkins node agent.

When you launch a Jenkins node agent it will download a JNLP file. A new process is launched on the client machine by the JNLP when it runs.

**4. How to take a backup of your Jenkins build jobs?**

Within the XML configuration each Jenkins build is stored. When this folder is copied, the configuration of all the build jobs that are managed by the Jenkins master are backed up. If you can perform a Jenkins Git integration, then it is good. When you copy the contents of the folder, you will see that the build jobs described in the folder will be restored when the Jenkins server is started the next time.

**5. what are the steps included in a Jenkins pipeline.**

A complete Jenkins pipeline will include building a project from the source code, putting it through a variety of unit, integrating, testing for user acceptance and performance and then finally deploying the packaged application on an application server.

So the steps in a Jenkins pipeline will include:

* Build
* Test
* Deploy

**6. How to turn off Jenkins security if the administrative users have locked out of the admin console?**

There is a folder that contains a file named config.xml. you need to change the settings to false for the security to be disabled when Jenkins is started the next time.

**7. Explain what is the Jenkins tool?**

Jenkins can be thought of as an open source automation tool that is used for continuous integration. You will be able to continuously test your software projects so that the developers will be able to integrate the changes to the project. You can also integrate with a large number of testing and deployment technologies.

**8. State some of the advantages of using Jenkins?**

Here are some of the most important advantages of Jenkins:

* You will get an automated build report every time a change is made to the source code
* You will be able to achieve continuous integration with agile methodology principles
* You can automate the maven release project with a few simple steps
* The bugs can be easily tracked at the early development stage.

**9. What are the requirements for using Jenkins?**

Here we list some of the requirements for using Jenkins:

* A source code repository like a Git repository
* A build script like a Maven script that is checked into the repository.

[**Learn DevOps from Experts! Enrol Today**](https://intellipaat.com/devops-certification-training/#course-content)

**10. What are some of the useful plugins in Jenkins?**

Here we are listing some of the important plugins that are used with Jenkins:

* Git repository
* Amazon EC2
* HTML Publisher
* JDK parameter plugin
* Configuration slicing plugin.

**11. How to setup a Jenkins job?**

First you need to create the Jenkins job by going to the Jenkins top page, selecting a “New Job” and building a freestyle software project.

Some of the elements of a free style project include:

* You need a CVS or a subversion where your source code will reside
* When the Jenkins will perform the builds, you will need the optional triggers
* You need a build script like a Maven or Ant where actually the script is built

**12. What is the process for creating a backup and copy files in Jenkins?**

If you want to create a backup for your file then you need to regularly backup your Jenkins\_Home directory. This will include all the build jobs configuration, all the slave node configuration and the build history. If you want to create a Jenkins backup, you can copy a job directory to the clone or can rename the directory.

**13. What is the process for securing Jenkins?**

Here we will explain the process for securing Jenkins. First you need to ensure global security. Then you have to make sure that the Jenkins is integrated with the user directory through appropriate plugin. The project matrix is enabled for fine tuning the access. Using the custom version-controlled script for automating the process of rights and privileges in Jenkins. The access to Jenkins data or folder is limited. You will run security audits on it.

**14. How do you compare the two tools of Hudson and Jenkins?**

Initially Jenkins was called as Hudson and there were some issues and due to this the name was changed to Jenkins from Hudson.

**15. If there is a broken build in your Jenkins project, then what do you do?**

First you need to open the console output where the broken build is created and then see if there are any file changes that were missed. If you do not find any issues in this manner, then you can update your local workspace and replicate the problem and then try to solve it.

**16. From one server to another how do you copy or move your Jenkins jobs?**

First you need to copy your jobs directory from the old to the new server. There are multiple ways to do it. You can either move the job from installation to simply copying the corresponding job directory, or you can make a clone of the job directory by making an existing job copy. First you need to have a different name which you can rename later.

**17. How to schedule builds in Jenkins?**

Here there are some steps for scheduling of builds in Jenkins

* First you should have a source code management commit
* You have to complete the other builds
* You have to schedule it to run at a specified time
* Give a manual build request.

**18. Name some of the SCM tools that are supported by Jenkins?**

Some of the important SCM tools that are supported by Jenkins include:

* Git
* Subversion
* CVS
* Mercurial

## Top Answers to Kubernetes Interview Questions

**1. What is Kubernetes?**

First let us compare Kubernetes with Docker Swarm

|  |  |  |
| --- | --- | --- |
| **Comparison** | **Kubernetes** | **Docker Swarm** |
| Controller | Master | Manager |
| Slave | Nodes | Worker |
| Deployment Unit | Pod | Task |
| Load balancing | Service | Ingress |

Kubernetes is a container orchestration tool that is used for automating the managing, monitoring, scaling and deploying of containerized applications. It creates groups of containers that can be logical discovered and managed for easy operations on containers.

**2. What are the benefits of Kubernetes?**

With container orchestration tool Kubernetes, it becomes extremely easy to handle the containers. You can respond to customer demands by deploying the applications faster and in a more predictable manner.

Here we will list some of the benefits of Kubernetes:

* Automatic scheduling
* Automated rollback
* Horizontal scaling
* Auto healing capabilities.

Become DevOps Certified in 16 hrs.

**GET CERTIFIED**

**3. What is the difference between Kubernetes and Docker Swarm?**

Docker Swarm is the default container orchestration tool that comes with Docker. Docker Swarm can only orchestrate simple Docker containers. Kubernetes on the other hand helps to manage much more complex software application containers. Kubernetes offers support for larger demand production environment.

**4. What is orchestration in software?**

Application orchestration in software process means you can integrate two or more applications. You will be able to automate arrangement, coordination and management of computer software. The goal of any orchestration process is to streamline and optimize frequent repeatable processes.

**5. What is a Kubernetes namespace?**

The Kubernetes namespace is used in the environment wherein you have multiple users spread in geographically vast areas and working on multiple projects. What the namespace does is divide the cluster resources between multiple users.

**6. What is a Pod in Kubernetes?**

You can think of Kubernetes Pod as a group of containers that are run on the same host. So if you regularly deploy single containers then your container and Pod will be one and the same.

**7. What is a node in Kubernetes?**

A node in Kubernetes is a worker machine which is also known as a minion. This node could be a physical machine or a virtual machine. For each of the node there is a service to run pods and it is managed by master components. The node services could include kubelet, kube-proxy and so on.

**8. What is a Heapster?**

The Heapster lets you do the container cluster monitoring.It lets you do cluster-wide monitoring and event data aggregation. It has native support for Kubernetes.

**9. What is a container cluster?**

A container cluster lets you place and manage the containers in a dynamic setup. It can be considered as a set of nodes or Compute Engine instances. The API server of the Kubernetes does not run on cluster nodes but instead the Container Engine hosts the API server.

**10. What is the Kubelet?**

You can think of Kubelet as the lowest level component in a Kubernetes. The Kubelet is responsible for making the individual machines to run. The sole purpose of a Kubelet is given a set of containers, it has to ensure that they are all running.

[**Interested in Learning DevOps? Click Here**](https://intellipaat.com/devops-certification-training/#course-content)

**11. What is Minikube?**

The Minikube makes it easy for the local running of Kubernetes. Within a Virtual Machine, the Minikube runs a single-node Kubernetes cluster.

**12. What is Kubectl?**

The Kubectl is a Kubernetes command line tool that is used for deploying and managing applications on Kubernetes. The Kubectl is especially useful for inspecting the cluster resources, for creating, updating and deleting the components.

**13. What is the Gke?**

The Gke is a Google Kubernetes Engine which is used for managing and orchestrating systems for Docker containers. The Gke also lets you orchestrate container clusters within the Google Public Cloud as well.

**14. What is Kube proxy?**

The Kube proxy runs on each of the node. It can do simple tasks like TCP, UDP Forwarding and so on. It shows the services in the Kubernetes API on each node.

**15. What are the Kubernetes Master components?**

The components of the Kubernetes Master include the API server, the controller manager, Scheduler, and the etc components. The Kubernetes Master components are responsible for running and managing the Kubernetes cluster.

**16. What is the use of kube-controller-manager?**

It is the Kubernetes controller manager. The kube-controller-manager is a daemon that embeds the core control loops which regulates system state and it is a non-terminating loop.

**17. What is load-balancing on Kubernetes?**

The process of load-balancing will let you expose the services. There are two types of load-balancing when it comes to Kubernetes.

Internal load balancing: This is used for balancing the loads automatically and allocating the pods with the required configuration.

External load balancing: This directs the traffic from external loads to the backend pods.

**18. what does a Kube-scheduler do?**

The Kube-scheduler has to job of assigning the nodes to the newly created pods.

**19. Where is the Kubernetes cluster data stored?**

The primary data store of Kubernetes is etcd which is responsible for all Kubernetes cluster data store.

**20. How to set a static IP for Kubernetes load balancer?**

Kubernetes Master assigns a new IP address.

You can set a static IP for Kubernetes load balancer by changing the DNS records every time the