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 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 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 and the memory doesn't run out from the hosts which runs all these innumerable containers hosting your applications.

Check Out Docker Tutorials

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.

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.