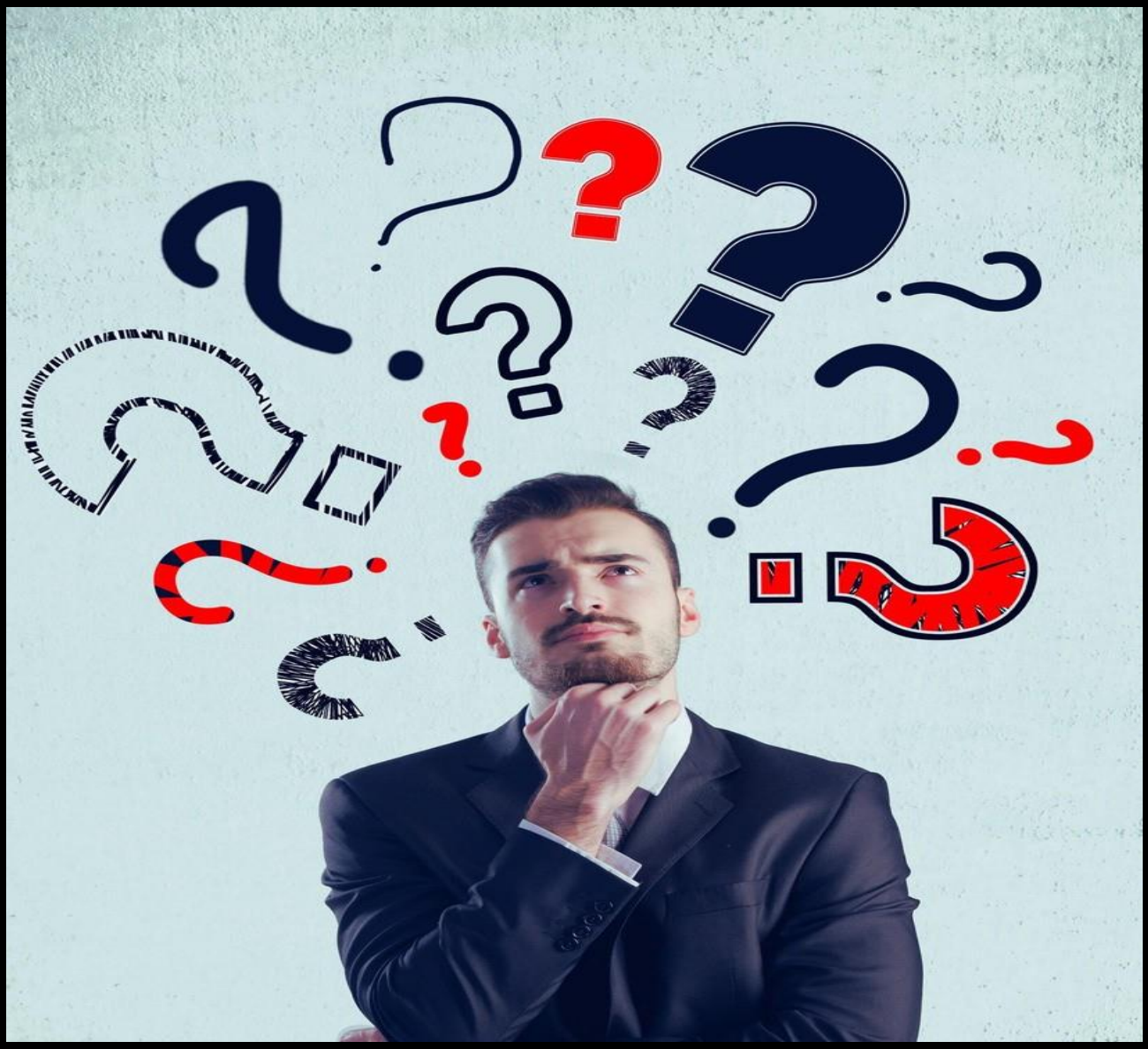
DOCKER

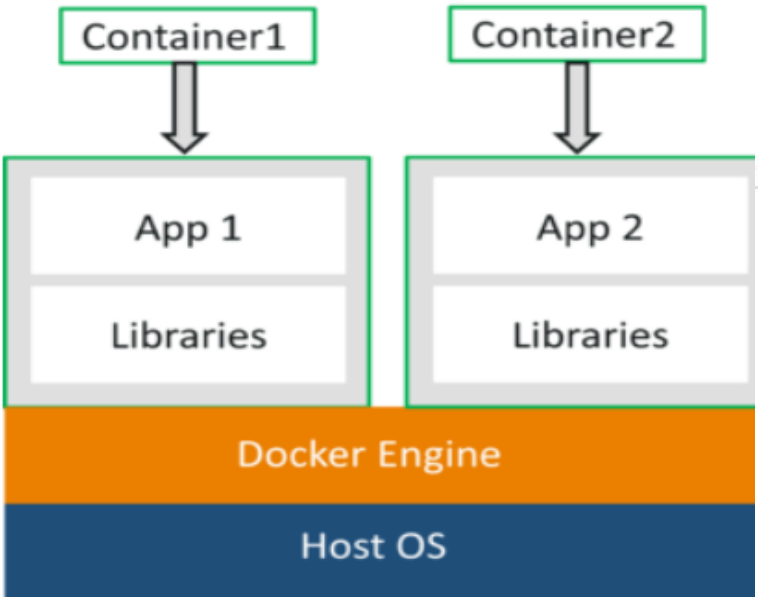
INTERVIEW QUESTIONS AND ANSWERS



Phone/WhatsApp: +1 (515) 309-7846 (USA)
Email: info@zarantech.com
Website: www.zarantech.com

1. What is Docker?

Docker is a platform which packages an application and all its dependencies together in the form of containers. This containerization aspect ensures that the application works in any environment. Containerization - Docker Explained - Eureka's you can see in the diagram, each and every application runs on separate containers and has its own set of dependencies & libraries. This makes sure that each application is independent of other applications, giving developers surety that they can build applications that will not interfere with one another. So a developer can build a container having different applications installed on it and give it to the QA team. Then the QA team would only need to run the container to replicate the developer's environment. If you wish to learn more about Docker, then you can click here. Now, let me tell you some more basic concepts of Docker, such as Docker file, images & containers. You can get a better understanding with this Online Docker Certification Training Course.



2. What is Docker & Docker Container?

Docker is a containerization platform that packages your application and all its dependencies together in the form of a Docker container to ensure that your application works seamlessly in any environment. Docker Container is a standardized unit which can be created on the fly to deploy a particular application or environment. It could be an Ubuntu container, CentOS container, etc. to full-fill the requirement from an operating system point of view. Also, it could be an application oriented container like CakePHP container or a Tomcat-Ubuntu container etc.

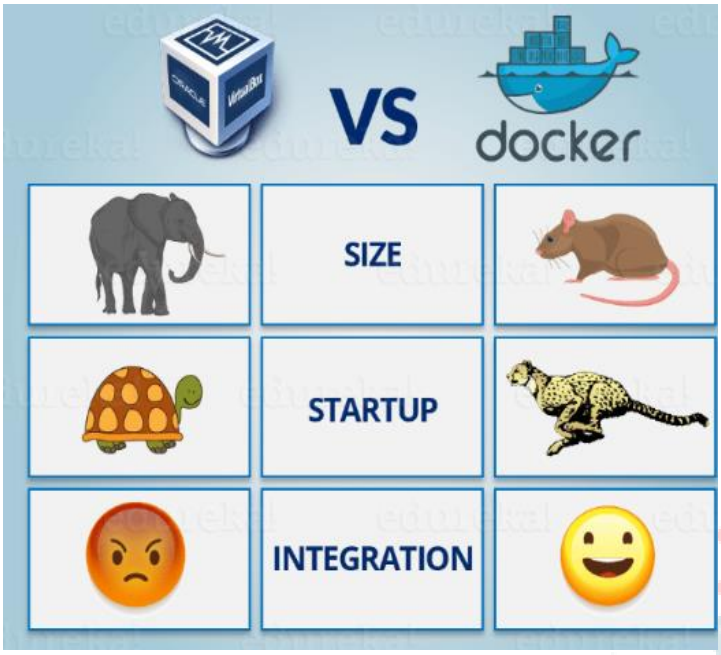
Let's understand it with an example:

A company needs to develop a Java Application. In order to do so the developer will setup an environment with tomcat server installed in it. Once the application is developed, it needs to be tested by the tester. Now the tester will again set up tomcat environment from the scratch to test the application. Once the application testing is done, it will be deployed on the production server. Again the production needs an environment with tomcat installed on it, so that it can host the Java application. If you see the same tomcat environment setup is done thrice. There are some issues that I have listed below with this approach:

- There is a loss of time and effort.
- There could be a version mismatch in different setups i.e. the developer & tester may have installed tomcat 7, however the system admin installed tomcat 9 on the production server.

Now, I will show you how Docker container can be used to prevent this loss. In this case, the developer will create a tomcat Docker image (An Image is nothing but a blueprint to deploy multiple containers of the same configurations) using a base image like Ubuntu, which is already existing in Docker Hub (the Hub has some base images available for free). Now this image can be used by the developer, the tester and the system admin to deploy the tomcat environment. This is how this container solves the problem. I hope you are with me so far into the article. In case you have any further doubts, please feel to leave a comment, I will be glad to help you. However, now you would think that this can be done using Virtual Machines as well. However, there is

catch if you choose to use virtual machine. Let's see a comparison between the two to understand this better.



Let me take you through the above diagram. Virtual Machine and Docker Container are compared on the following three parameters:

- Size – This parameter will compare Virtual Machine & Docker Container on their resource they utilize.
- Start-up – This parameter will compare on the basis of their boot time.
- Integration – This parameter will compare on their ability to integrate with other tools with ease.

I will follow the above order in which parameters are listed. So first parameter would be "Size".

3. What is Virtualization?

Virtualization is the technique of importing a Guest operating system on top of a Host operating system. This technique was a revelation at the beginning because it allowed developers to run multiple operating systems in different virtual machines all running on the same host. This eliminated the need for extra hardware resource. The advantages of Virtual Machines or Virtualization are:

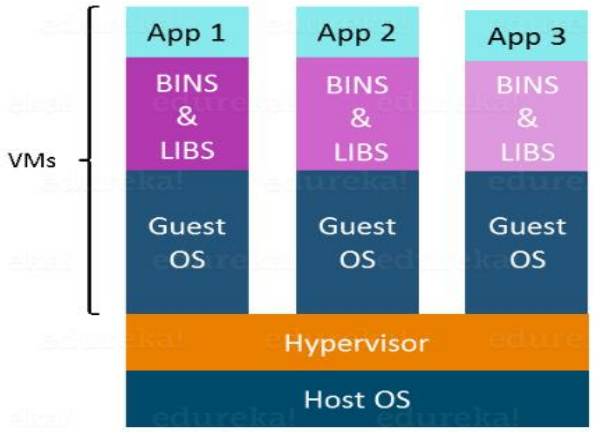
- Multiple operating systems can run on the same machine
- Maintenance and Recovery were easy in case of failure conditions
- Total cost of ownership was also less due to the reduced need for infrastructure

Virtual Machine Architecture - Docker Tutorial On Introduction to Docker – Eureka in the diagram on the right, you can see there is a host operating system on which there are 3 guest operating systems running which is nothing but the virtual machines. As you know nothing is perfect, Virtualization also has some shortcomings. Running multiple Virtual Machines in the same host operating system leads to performance degradation. This is because of the guest OS running on top of the host OS, which will have its own kernel and set of libraries and dependencies. This takes up a large chunk of system resources, i.e. hard disk, processor and especially RAM. Another problem with Virtual Machines which uses virtualization is that it takes almost a minute to boot-up. This is very critical in case of real-time applications.

Following are the disadvantages of Virtualization:

- Running multiple Virtual Machines leads to unstable performance
- Hypervisors are not as efficient as the host operating system
- Boot up process is long and takes time

These drawbacks led to the emergence of a new technique called Containerization. Now let me tell you about Containerization.



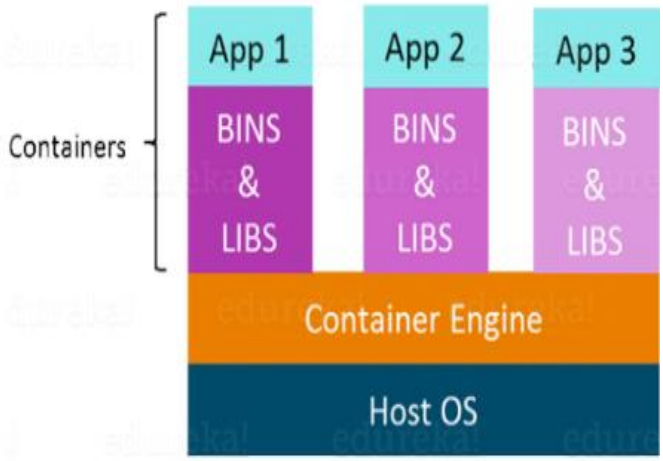
4. What is Containerization?

Containerization is the technique of bringing virtualization to the operating system level. While Virtualization brings abstraction to the hardware, Containerization brings abstraction to the operating system. Do note that Containerization is also a type of Virtualization. Containerization is however more efficient because there is no guest OS here and utilizes a host's operating system, share relevant libraries & resources as and when needed unlike virtual machines. Application specific binaries and libraries of containers run on the host kernel, which makes processing and execution very fast. Even booting-up a container takes only a fraction of a second. Because all the containers share, host operating system and holds only the application related binaries & libraries. They are lightweight and faster than Virtual Machines.

Advantages of Containerization over Virtualization:

- Containers on the same OS kernel are lighter and smaller
- Better resource utilization compared to VMs
- Boot-up process is short and takes few seconds
- Container Architecture - Docker Tutorial On Introduction to Docker - Eureka

In the diagram on the right, you can see that there is a host operating system which is shared by all the containers. Containers only contain application specific libraries which are separate for each container and they are faster and do not waste any resources.

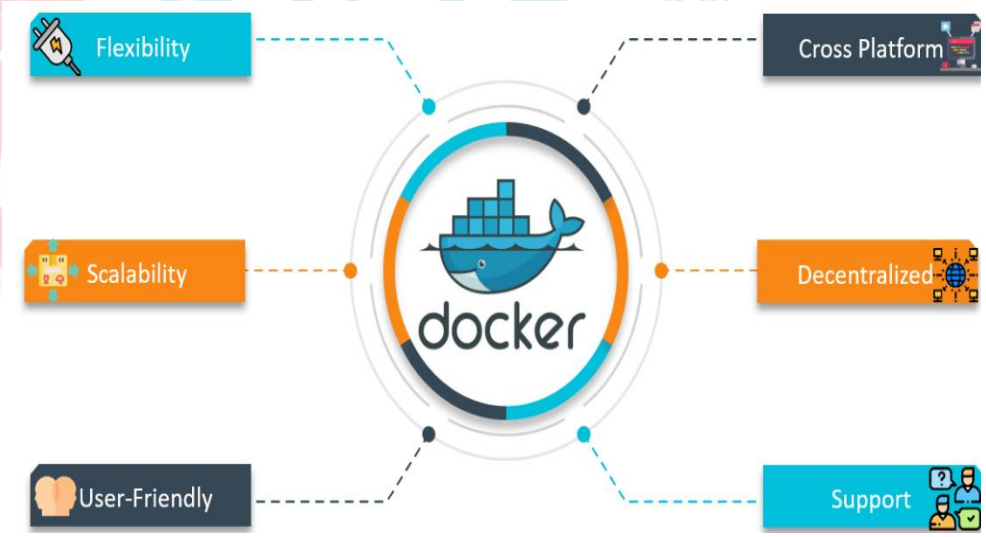


5. What is Docker?

Docker is an open-source project that offers a software development solution known as containers. To understand Docker, you need to know what containers are. According to Docker, a container is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it. Containers are platform-independent and hence Docker can run across both Windows and Linux-based platforms. In fact, Docker can also be run within a virtual machine if there arises a need to do so. The main purpose of Docker is that it lets you run micro service applications in a distributed architecture. When compared to Virtual machines, the Docker platform moves up the abstraction of resources from the hardware level to the Operating System level. This allows for the realization of the various benefits of Containers e.g. application portability, infrastructure separation, and self-contained micro services. In other words, while Virtual Machines abstract the entire hardware server, Containers abstract the Operating System kernel. This is a whole different approach to virtualization and results in a much faster and more lightweight instances.

6. What do you think are the goals of Docker Networking?

- Flexibility – Docker provides flexibility by enabling any number of applications on various platforms to communicate with each other.
- Cross-Platform – Docker can be easily used in cross-platform which works across various servers with the help of Docker Swarm Clusters.
- Scalability – Docker is a fully distributed network, which enables applications to grow and scale individually while ensuring performance.
- Decentralized – Docker uses a decentralized network, which enables the capability to have the applications spread and highly available. In the event that a container or a host is suddenly missing from your pool of resource, you can either bring up an additional resource or pass over to services that are still available.
- User – Friendly – Docker makes it easy to automate the deployment of services, making them easy to use in day-to-day life.
- Support – Docker offers out-of-the-box supports. So, the ability to use Docker Enterprise Edition and get all of the functionality very easy and straightforward, makes Docker platform to be very easy to be used.



7. What is a VM (Virtual Machine)?

A VM is a virtual server that emulates a hardware server. A virtual machine relies on the system's physical hardware to emulate the exact same environment in which you install your applications. Depending on your use case, you can use a system virtual machine (that runs an entire OS as a process, allowing you to substitute a real machine for a virtual machine), or process virtual machines that let you execute computer applications alone in the virtual environment. Earlier, we used to create virtual machines, and each VM had an OS which took a lot of space and made it heavy.

8. What Is Docker Swarm?

Docker Swarm is a technique to create and maintain a cluster of Docker Engines. The Docker engines can be hosted on different nodes, and these nodes which are in remote locations form a Cluster when connected in Swarm mode.

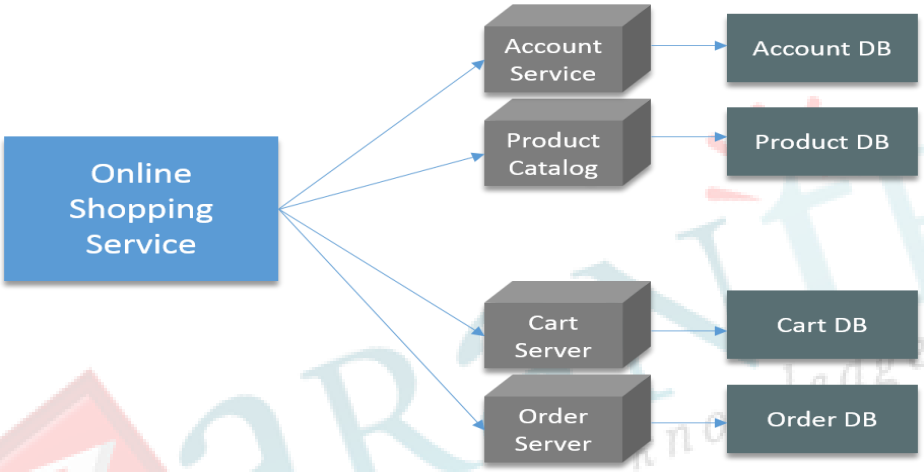
9. Why Use Docker Swarm?

For reasons mentioned already! Achieving high availability without any downtime is a priority for every service provider out there. Will high availability impress your clients? Well, they won't be impressed if they face downtime. That is a no-brainer.

10. Why We Need Docker Containers?

I still remember it correctly; I was working on a project. In that project we were following the micro service architecture. For those of you who don't know what is micro service, don't worry I will give you an introduction to it. The idea behind micro services is that certain types of applications become easier to build and maintain when they are broken down into smaller, compostable pieces which work together. Each component is developed separately, and the application is then simply the sum of its constituent components.

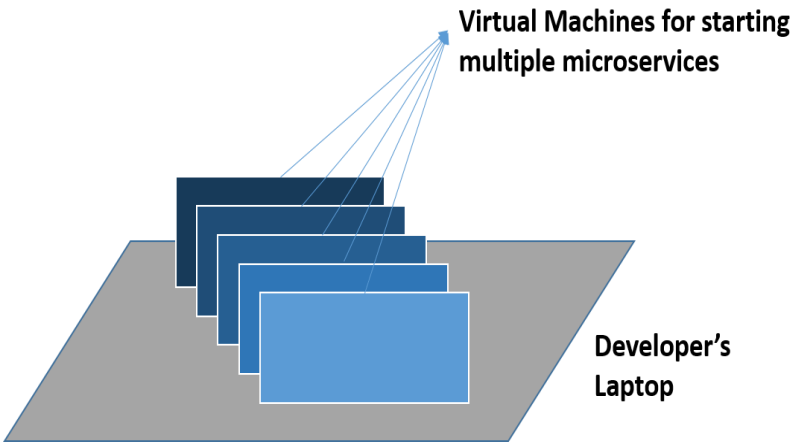
Consider the example below:



In the above diagram there is an online shop with separate micro services for user-account, product catalo, order processing and shopping carts. Well, this architecture has a lot of benefits:

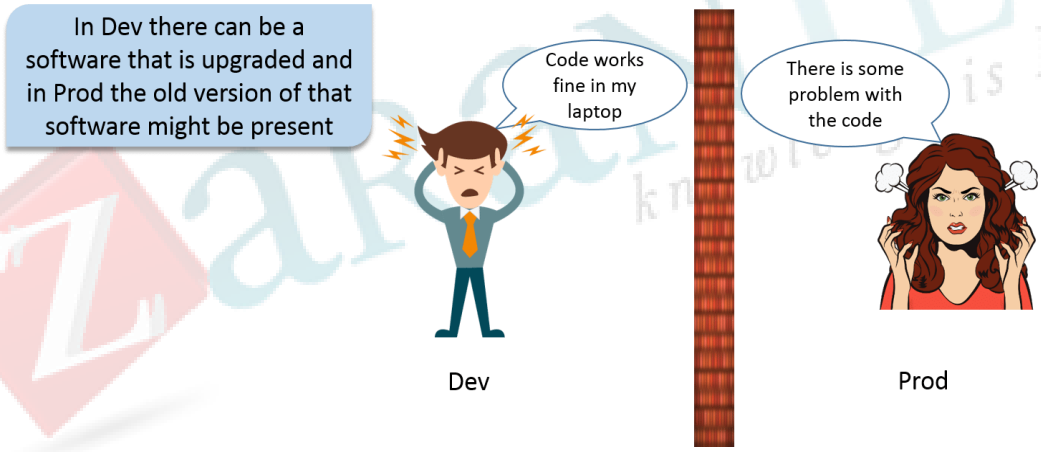
- Even if one of your micro service fails, your entire application is largely unaffected.
- It is easier to manage

There are many other benefits as well, I won't go into much detail about micro services in this post. But, soon I will be coming up with a couple of blogs on micro services as well. In this architecture, we were using CentOS Virtual Machines. Those Virtual Machines were configured by writing long scripts. Well, configuring those VMs was not the only problem. Developing such applications requires starting of several of micro services in one machine. So if you are starting five of those services you require five VMs on that machine. Consider the diagram below:



The other problem is pretty common; I know a lot of you can relate to it. The application works in a developer's laptop but not in testing or production. This can be because of not keeping a consistent.

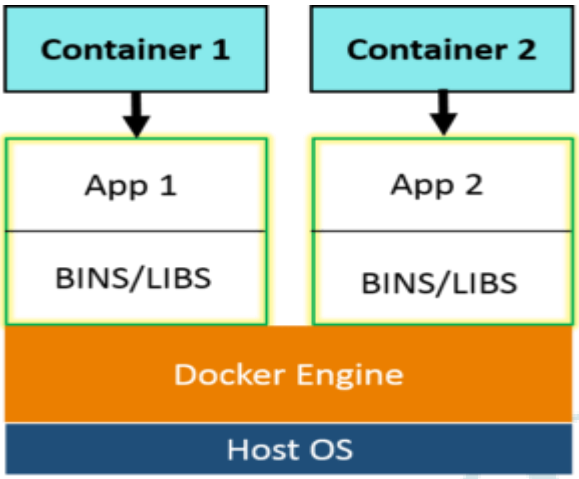
computing environment. Consider the diagram below:



There were many other problems apart from this as well, but I feel, these problems are enough for me to explain you the need of Docker Containers.

11. What is a Docker Container?

Docker is a tool designed to make it easier to create, deploy and run applications by using containers. You can create Docker Containers, these containers will contain all the binaries and libraries required for your application or micro service in my case. So your application is present in a container, or you have containerized your application. Now, that same container can be used in the Test and Prod environment.



Docker Containers are a lightweight solution to Virtual Machines, and it uses the host OS. The best part, you don't have to pre-allocate any RAM to the Docker Container, it will take it as and when required. So, with Docker Container I don't have to worry about wastage of resources.

12. Why use Docker for Windows?

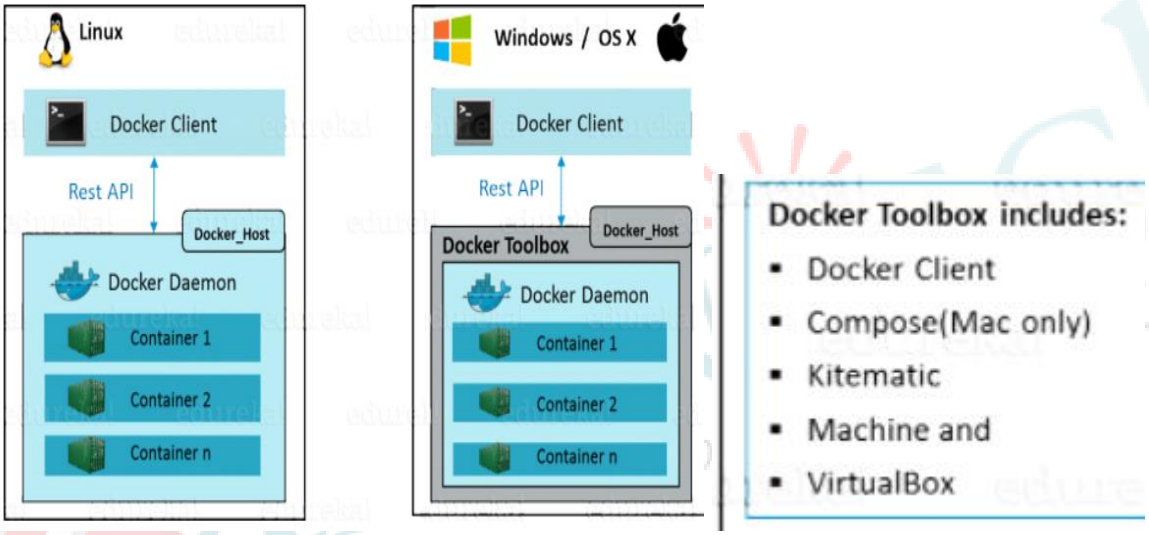
- Avoids the work on my machine but doesn't work on production problem: This problem occurs due to the inconsistent environment throughout the software development workflow. With Docker you can run an application within a container which contains all the dependencies of the application and the container can be run throughout the software development cycle. This practice provides a consistent environment throughout the software development life cycle
- Improves productivity: By installing Docker on windows we're running Docker natively. If you've been following Docker for a while, you know that Docker containers originally supported only Linux operating systems. But thanks to the recent release, Docker can now natively run on windows, which means that Linux support is not needed, instead the Docker container will run on the windows kernel itself
- Supports native networking: Not only the Docker container, the entire Docker tool set is now compatible with windows. This includes the Docker CLI (client), Docker compose, data volumes and all the other building blocks for Dockerizied infrastructure are now compatible with windows. But how is this advantageous? Since all the Docker components are locally compatible with windows, they can now run with minimal computational overhead. You can get a better understanding with this Online Docker Training Course.

13. What is Docker Engine?

Now I will take you through Docker Engine which is the heart of the system.

Docker Engine is simply the application that is installed on your host machine. It works like a client-server application which uses:

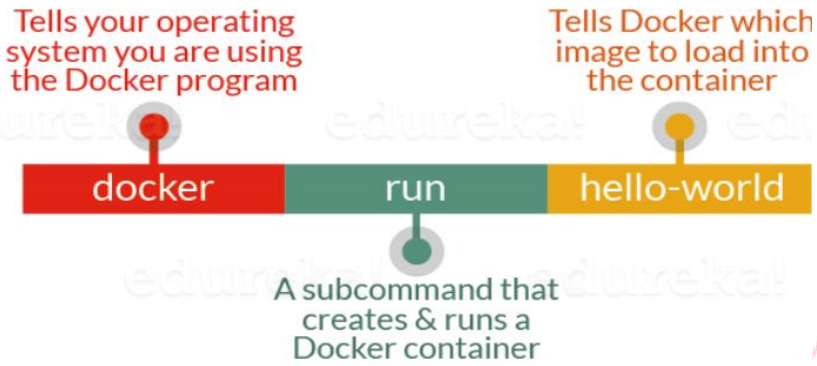
- A server which is a type of long-running program called a daemon process
- A command line interface (CLI) client
- REST API is used for communication between the CLI client and Docker Daemon



As per the above image, in a Linux Operating system, there is a client which can be accessed from the terminal and a Host which runs the Daemon. We build our images and run containers by passing commands from the CLI client to the Daemon. However, in case of Windows/Mac there is an additional Toolbox component inside the Docker host. This Docker Toolbox is an installer to quickly and easily install and setup a Docker environment on your Windows/iOS. This Toolbox installs Docker Client, Machine, Compose (Mac only), Kinematic and Virtual Box. Let's now understand three important terms, i.e. Docker Images, Docker Containers and Docker Registry.

14. What is Docker Image?

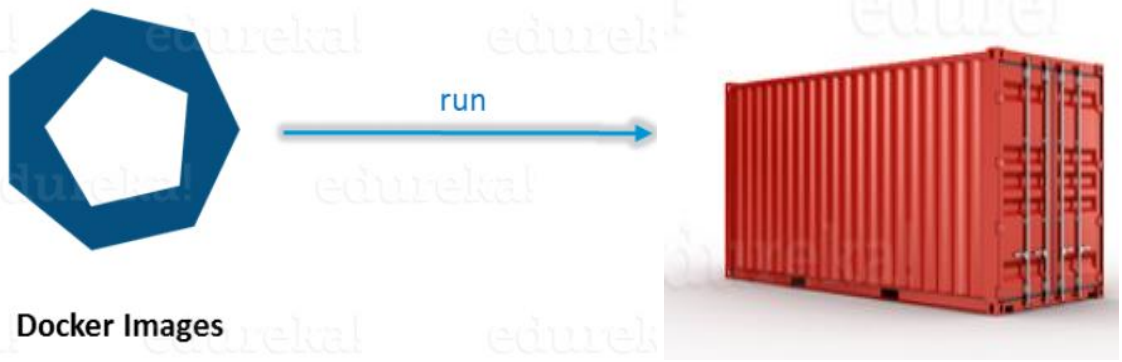
Docker Image can be compared to a template which is used to create Docker Containers. They are the building blocks of a Docker Container. These Docker Images are created using the build command. These Read only templates are used for creating containers by using the run command. We will explore Docker commands in depth in the "Docker Commands blog".



Docker lets people (or companies) create and share software through Docker images. Also, you don't have to worry about whether your computer can run the software in a Docker image — a Docker container can always run it. I can either use a ready-made Docker image from Docker-hub or create a new image as per my requirement. In the Docker Commands blog we will see how to create your own image.

15. What is Docker Container?

Docker Containers are the ready applications created from Docker Images. Or you can say they are running instances of the Images and they hold the entire package needed to run the application. This happens to be the ultimate utility of the technology.



16. What is Docker Registry?

Finally, Docker Registry is where the Docker Images are stored. The Registry can be either a user's local repository or a public repository like a Docker Hub allowing multiple users to collaborate in building an application. Even with multiple teams within the same organization can exchange or share containers by uploading them to the Docker Hub, which is a cloud repository similar to GitHub.

17. What is Docker and what are the benefits of using it?

Docker is a containerization platform that packages applications into standardized units called containers. Benefits include portability across machines, lightweight with less resource usage than VMs, faster start times, and easier scaling & maintenance of apps.

18. What are containers in Docker?

Containers are lightweight, standalone, executable packages of software that include application code, runtimes, dependencies, and configurations that can run quickly and reliably from one computing environment to another.

19. How is Docker different from virtual machines?

Docker containers share the host OS kernel and run as isolated processes making them more lightweight and efficient unlike VMs which need full guest OSes. This allows faster start times and lower resource usage for Docker.

20. What are Docker images?

Docker images are read-only templates used for creating container instances. Images are made up of file system changes and configuration instructions used to deploy containers. Images become containers when they run on the Docker engine.

21. What is a Docker container?

Containers are runtime instances of Docker images launched by the Docker engine. They are isolated application platforms that bundle code and dependencies to run the application on host OS sharing the kernel.

22. What is Docker Hub?

Docker Hub is a cloud-based registry service for hosting Docker images. It provides capabilities to upload/download, store and share Docker images with teams and allows integration with Docker CLI to simplify pulling images for container launches.

23. What tools can be used to run Docker locally?

Docker engine, Docker Desktop (Mac/Windows) and Docker Compose can be used to run Docker containers on local dev and test machines for building applications.

24. What platforms and architectures are supported by Docker?

Docker supports all Linux distributions and versions later than 3.10 kernel. It also runs on macros and Windows 10 using its native hypervisor technologies. ARM and IBM systems architectures are also supported.

25. What is a Docker file?

A Docker file is a text file that contains instructions for building Docker images automatically using simple commands. It simplifies maintenance allowing image sources to be tracked and customized for specific environments.

26. How do you create a persistent data volume in a Docker container?

By attaching a host directory or an anonymous volume container to the container's image using the Docker run -v option during launch of container. Data gets persisted in volumes even if container is removed.

27. What is Docker network and how does it work?

The docker network command allows configuring networks for communication between containers launched within an isolated Docker network. Containers can be attached to multiple networks like bridge, overlay, macula etc. based on app needs.

28. How is security implemented for Docker based applications?

Through namespace isolation of containers from host and between containers, restricting privileges of containers using capabilities, network policies, file system permissions and deploying images scanned for vulnerabilities into trusted registries after sign & verification.

29. What is a Docker registry?

A Docker registry is a repository for storing, managing and distributing Docker images. Docker Hub is a public example while enterprises often host private registries to maintain development pipelines across environments and control image access.

30. How can Docker deployment environments be automated?

Using tools like Docker Compose for local development, Docker Swarm for cluster management and popular CI/CD frameworks like Jenkins, Circles etc. to set up deployment pipelines - promoting images to higher environments.

31. What is Docker Compose?

Docker Compose is a tool for defining and running multi-container Docker applications by using a YAML configuration file for orchestrating the application services dependency and networking. This allows automated one-click environment setups.

32. What platforms offer managed Docker hosting solutions?

AWS ECS, Azure Container Instances, Google Kubernetes Engine and IBM Cloud Kubernetes Service allow organizations to run Docker zed applications without having to self-manage the infrastructure. They offer scalable, highly-available container hosting platforms.

33. How can Docker containers be monitored?

Using Docker APIs and CLI commands for basic monitoring. For advanced monitoring, metrics and logging solutions like Prometheus, Granma, cad visor, Elastic Stack etc. allow gathering operational insights on container health, app performance, resource usage and logs.

34. What is a Docker Image Registry?

A Docker image registry is a storage and content delivery system holding named Docker images, available in different tagged versions, that are pulled for container launches based on requirement. Popular registries are public like Docker Hub or private like AWS ECR.

35. How does networking work between Docker containers?

Containers can be attached to virtual bridge networks that route traffic between containers internally based on IP addresses and expose ports externally to allow ingress traffic from outside based on mappings defined at runtime.

36. What are the popular storage drivers for the Docker engine?

Overlays, AUFS, ZFS, Bars, Device Mapper are some of the widely used Docker storage drivers. Overlay2 and AUFS see maximum adoption currently for union file system capabilities layering image fs over host fs.

37. How can persistent data volumes be shared between containers?

By using shared data volumes that allow detachable/reattach able data directories used by containers identified by volume names allowing portability of tasteful data across environments between app upgrades.

38. What is a Docker Swarm?

Docker Swarm provides native clustering capabilities to turn a group of Docker engines participating as nodes into a single virtual Docker engine for easier scaling and maintenance of a distributed application. Enables natively deploying containers across a pool of nodes.

39. What platforms can be used for orchestrating Docker at scale?

Docker Swarm, Kubernetes and Apache Mesas are popular container orchestration technologies that help manage clusters of Docker hosts as a single deployment target supporting replication, scaling, networking, security etc.

40. What is a Docker repository?

A Docker repository refers to a hosted registry codebase containing Docker images organized by families i.e. suites of images identified by names which denote applications/middleware software releases tagged by versions.

41. How does one control CPU and memory resources for containers?

By limiting container access to a defined percentage of total host resources using the -m and -cpuset-cpus flags with Docker run to restrict memory usage and assign specific CPUs.

42. How can Docker containers be terminated automatically if inactive?

Using Docker run --rm instructs Docker engine to delete containers immediately after they stop executing allowing removal of stale temporary containers that are no longer active and not required.

43. What interface helps applications use Docker containers without knowing specifics?

Docker Engine SDK allows applications to make calls to control Docker via simple APIs rather than CLI/tool specifics. Libraries are available for popular languages allowing app portability across environments.

44. What is a Union File System as mentioned in Docker storage drivers?

Unions allows overlaying multiple directories transparently into cohesive file system providing single merged view across layers forming image via efficient copy-on-write model rather than duplicating files.

45. What is container orchestration in context of Docker?

Container orchestration refers to automated deployment, scaling and management of containerized applications across clusters of hosts. Tools like Docker Swarm and Kubernetes help coordinate containerized services across infers.

46. How does Docker simplify dependency management for apps?

Building containers with app runtimes & libraries needed bundled inside isolated containers makes running apps with dependencies easier without worrying about external environment differences across deployments.

47. What is a Docker Entry point?

Entry point allows specifying a default executable for running containers as command overrides when launched. If not overridden, entry point gets invoked allowing standardization of runtime behaviours without altering base images.

48. What is meant by Docker zing applications & why is it advantageous?

Docker zing apps refers to packaging, distributing apps and dependencies into standardized images that can run as isolated, portable containers consistently across environments - simplifying maintenance and coordination.

49. What are the popular monitoring solutions for Docker deployments?

cad visor, Prometheus, Granma provide insights into resource usage and performance metrics of Docker engines and containers. Additional logging tools are required for gathering application logs from containers.

50. How can security vulnerabilities be minimized for Docker containers in devils pipelines?

By following guidelines on creating optimized base images free of additional packages, fixing packages with only latest security patches, and scanning container images in pipelines before deploying to higher environments.

Corporate Training Course Catalog
https://bit.ly/devops-course-catalog

DevOps Learner Community
https://www.linkedin.com/showcase/devops-learner-community/

Get any DevOps Video Training
https://zarantech.teachable.com/courses/category/devops

Phone/WhatsApp: +1 (515) 309-7846 (USA)
Email: info@zarantech.com
Website: www.zarantech.com