What is the need for DevOps?

Companies are trying to release updates as soon as they are developed. Releasing small features like this has many advantages over full one time Big-Release like quick feedback from clients, better quality etc. To achieve this, companies need to:

Increases the deployment frequency

Very fast Deployments of new features

Less complexity in SDLC process

Lower failure rate of new releases

Shortened lead time between fixes

Improved communication and collaboration

Will get more time to develop new features

Rolling back a new update will not be as complex as in Big-Release.

DevOps fulfills all these and helps in achieving a seamless software delivery cycle.

If you vaguely remember a command, without googling how will you recollect it?

This command lists all Docker commands. If you need help with one specific command, you can use the following syntax:

$ docker < command > --help (or) simply $ docker --help

What is Hyper-V or Hypervisor?

Hypervisor is a software that makes virtualization possible. It divides the host system resources and allocates the resources to each divided virtual environment. The hypervisor allows you to create a virtual environment on your system so that guest virtual machines can be created and operated. It has control over resource allocation to multiple virtual machines.

In short, hypervisor allows you to create multiple virtual servers on a single physical server. Some hypervisor software are Hyper-V, Xen, VMware Player, Virtual-PC, Oracle-Box, KVM, etc.

What is Docker?

Docker is an open-source lightweight OS-level containerization technology. It bundles the whole application and its dependencies together and forms a container so that your application works hassle free in any environment (Development, Test or Production). Containers are lightweight and portable and they can communicate with each other. Containerization technology like Docker will share the same base machine’s operating system kernel, and due to this it is extremely fast. With the help of containers any app can Build, Ship and Run.

What is virtualization?

Virtualization is the logical division of your physical machine into multiple machines(compute storage, servers, application, etc.). However, this allows multiple OS to run simultaneously on a single system. A software called Hypervisor makes this kind of splitting possible. The virtual environment created by the hypervisor is called Virtual Machine.

What is containerization?

Usually, code developed on one machine might not work perfectly fine on any other machine because of the dependencies. So an application is bundled with all its configuration files and dependencies so that it could work on any machine. To make you understand this bundle is called a container. Most popular containerization tool after rocker is Docker.

What is a Docker Container?

Container is a process, or Docker containers are runtime instances of the Docker image. one container is an instance of one image. Docker containers have the application and its dependencies in it. It shares the kernel of the host operating system. Docker containers are not limited to any specific IT infrastructure, they can run on any computer, infrastructure, and in any cloud. Docker containers are created using Docker images and then run it, or we can use the images that are already created in the Docker Hub.

What are Registries and types?

Docker's registry is nothing but a Docker Hub, which allows you to store images publicly/privately. Millions of images/registries can be stored to docker hub.

There are two types of registry is

Public Registry

Private Registry

Public registry can be accessed by any one, and private has restrictions to be defined.

What is a Docker Hub?

There has to be a centralized registry to store docker images. Docker Hub is one among those cloud registry which store repositories(images). We can also store or push images to Docker Hub, So that the images can be deployed to the host. Users can pull images from Docker Hub and use them to create customized images and containers.

What are Docker Images?

To make you understand, just assume Docker image is a kind of mini OS of Docker container. But keep in mind that containers do not have their own OS or a separate kernel, containers share the OS of the base machine. Docker images are used to create containers. Every docker image can be stored as the Docker registry in Docker Hub. You can create your own images by using an existing container or by writing Dockerfile.

How to create a docker container from an image?

Get required images from docker repository and run it to create a container. Use the following command:

$ docker run -it <image\_name> /bin/bash

-t (tty) flag for a terminal which is bin bash here.

-i flag for interactive mode, means we will get terminal in interactive mode.

This is a basic command to create a container. This command is enough to create a container.

But we can create a container by adding extra flags as per need, as follows

$ docker run -itd <image\_name> /bin/bash

-d flag to create a container in the detached mode.

$ docker run -itd <image\_name> --name <container\_name> /bin/bash

--name option to give a name to your container.

$ docker run -itd <image\_name> --name <container\_name> -h <host-name> /bin/bash

-h flag to provide a host name to your container.

$ docker run -itd <image\_name> --name <container\_name> -h <host-name> -v <source-path>:<destination-path> /bin/bash

-v flag to allocate volume to your container.

$ docker run -itd <image\_name> --name <container\_name> -h <host-name> -v <source-path>:<destination-path> -p8080:8080 /bin/bash

-p flag to expose the base container’s 8080 port on the 8080 port of the base machine.

What are docker volumes?

Docker volumes are like virtual disks to store data and can be shared between containers, in simple docker volumes are just like normal directories which contain sub directories and files stored in host machines, but these directories are mounted to containers so that the data can be accessed in containers. We can create volumes by the following command:

$ docker volume create <volume-name>

Then mount this volume to the container using -v flag, or you can mount the existing directory to containers as volume.

Where the docker volumes are stored?

In a linux machine Docker volumes are stored at this location:

/var/lib/docker/volumes

What are the states of Docker containers?

Important states of Docker container are as listed below:

Running

Paused

Restarting

Exit.

What is a Docker Swarm?

Docker Swarm is an orchestration tool for Docker containers. Docker Swarm is used to scale up/down nodes in a cluster. Even clusters can be created by Docker Swarm. It uses a network for internal communication among containers. Docker Swarm is native clustering for Docker. It turns a container pool of Docker hosts into a single Docker host. Docker Swarm manages the Docker API as well.

Do Docker support for IPV6?

Yes, but only if you modify /etc/docker/daemon.json and set the ipv6 key to "true". IPv6 networking is supported only on Docker daemons run on Linux hosts.

How does communication happen between Docker clients and the Docker Daemon?

Communication between Docker client and Docker Daemon happens with the combination of Rest API, socket.IO, and TCP.

What is a Dockerfile?

Docker can build images by reading the instructions given in a file called Dockerfile. A Dockerfile is a text document that contains all the commands in a sequence. This "docker build -t <Dockerfile/path> .", command will execute several instructions written in Dockerfile in a sequence to create Docker images. The common commands used in Dockerfile are: FROM, MAINTENANCE, COPY, RUN, and CMD.

How do you run multiple containers using a single service?

By using an orchestration tool like docker-compose, you can run multiple containers using a single service. All docker-compose files use yaml language.

What is Docker Compose?

Docker Compose contains information about the services, networks, and volumes for setting up the Docker application. So, you can use Docker Compose to create separate containers, and host them and get them to communicate with each other. Make sure that each container should expose a port for communicating with others. Docker Compose is written in YAML format.

What does the volume parameter do in a docker run command?

The volume parameter syncs a directory in a container with a host directory.

For example:

docker run -v nginx-sites:/etc/nginx/sites-available nginx

This command mounts the nginx-sites directory in the host to the /etc/nginx/sites-available directory. In this way, you can sync nginx sites without restarting the container they’re in. Also, you can protect your data that is generated in your container using a directory in the host. Otherwise, if you delete your container, your data that was generated and stored in your container will naturally be deleted.

When you use the volume parameter, you can use the same data that was generated in a previous container using the same command.

What is the use of the docker save and docker load commands?

A Docker image can be exported as an archive via the docker save command. For example:

docker save -o <container-export-path>.tar <container-name>

The exported Docker image can then be imported to another Docker host via the docker load command:

docker load -i <container-path>.tar

Note that this does not export data from any containers that were based on the image, just the image itself.

What is the main difference between the approaches of Docker and standard hypervisor virtualization?

With standard virtualization using a hypervisor like vSphere, an operating system is necessary for each app. A host operating system is at the bottom of your infrastructure, and a hypervisor has to be installed on your host OS. Then on top of the hypervisor, you install operating systems for each of your applications.

With Docker, the Docker daemon sits between your host operating system and your Docker images, in place of a hypervisor. Docker images reuse parts of the host operating system—thus a separate OS is not necessary for each app—but your apps are still isolated like they would be with a standard hypervisor.

What is the default Docker network driver, and how can you change it when running a Docker image?

Docker provides different network drivers like bridge, host, overlay, and macvlan. bridge is the default.

Sometimes you might want to use Docker Swarm or connect your containers to your host network directly. In these cases, you’ll need to change your default network driver.

First, you have to create a new network with the new network driver by using the --driver or -d parameter with your docker network create command. Then you’ll need to run your Docker image with the --network parameter to use your newly-created network.

What are a Docker container’s possible states, and what do they mean?

**Created**: If your docker container is newly created, you will see this state for your container. In this state, the container is not yet started.

**Restarting**: When you restart your docker container—or container restarts itself due to a problem—you will see this state.

Docker has four different restart policies. The default is called no. With this policy, the Docker daemon will never try to restart your container (unless you tell it to manually.)

The second policy is on-failure. With this policy, the Docker daemon will try to restart containers if any problem exists, that is, if any startup script returns a non-zero exit code.

The third policy is always. With this policy, the Docker daemon will try restart containers if:

1. Any problem exists,
2. You stop them manually, or
3. The docker daemon was itself stopped and restarted

The fourth policy is unless-stopped, where the Docker daemon will always try to restart containers unless you stop them manually.

**Running**: Running is the main state you’ll see for containers. It means it has started, and there is no problem detected with the container itself.

**Paused**: If you temporarily stop your running Docker container via docker pause, this is what you’ll see until you unpause it.

**Exited**: If your container has stopped because of a problem or you stopped your container manually, you will see your container in this state, depending on your restart policy as described above.

What is a Docker image? What is a Docker image registry?

A Docker image consists of many layers. Each layer corresponds to a command in an image’s Dockerfile. This image provides isolation for an application when you run a Docker image as a container.

You can run many containers from a single Docker image. Docker images can be built from a Dockerfile.

A Docker image registry is a storage area for Docker images. You can get images from them instead of building them.

An image registry is either public or private. The best-known public registry is Docker Hub.

What is container orchestration and why should we use it?

When you have to manage large and dynamic environments, the docker command alone does not suffice. You will face many problems automating scaling and health checks for containers. In this case, software teams use container orchestration tools like Kubernetes. Such software enables another level of automation:

* Deploy or scale your containers easily, securely, and with high availability
* Provide a service (internally or externally) from a container group
* Move your containers from one host to another when there’s a host-specific problem
* Manage your configuration data—like environment variables—easily

What features are provided by Docker Enterprise Edition instead of Docker Community Edition?

Docker Enterprise Edition provides certified Docker images and plugins. With this certification, Docker Inc. ensures that the images in question pass security and best-practice checks. In other words, they guarantee a certain baseline of reliability.

Docker Enterprise Edition also provides Active Directory or LDAP user integration, continuous vulnerability and security scans, and container app and image management features.

Is there any problem with just using the latest tag in a container orchestration environment? What is considered best practice for image tagging?

If you’re running your image via the latest tag with a container orchestration environment like Kubernetes, it may cause a problem.

The problem is if you push a new image with just the latest tag, you lose your old image and your deployments will use the new image. If the new image has any problem, your deployments might fail, resulting in downtime.

When you use explicit version numbers to tag Docker images instead, you can roll back to old images easily. Also, when you push a new image to your private registry, your deployments will continue to use the old version number due to your tag until you’re ready to switch each of them over.

The best practice of Docker image tagging is to use both types of tagging. First, tag your Docker images with latest and a version number, then push twice, separately for each tag. For example:

docker tag nginx:latest nginx:0.0.1

docker push nginx:latest

docker push nginx:0.0.1

What is Docker Swarm and which network driver should be used with it?

Docker Swarm is an open-source container orchestration tool that is integrated with the Docker engine and CLI. If you want to use Docker Swarm, you should use the overlay network driver. Using an overlay network enables the Swarm service by connecting multiple docker host daemons together.

What are the possible ways of using insecure Docker image registries?

In some projects, you might choose private Docker registries rather than Docker Hub or any cloud provider’s registry. This might take the form of deploying a Docker registry server, or perhaps a third-party on-premise registry server like Nexus.

When you want to connect these private registries, your registry should be secured with an SSL certificate in accordance with best practices.

You can also elect to use a private registry insecurely if you want to use self-signed SSL certificates—note, this should only be done for testing purposes. To do this, add your private test registry to an array as the value for the "insecure-registries" key in your daemon.json config fil

What is Docker Compose? What can it be used for?

Docker Compose is a tool that lets you define multiple containers and their configurations via a YAML or JSON file.

The most common use for Docker Compose is when your application has one or more dependencies, e.g., MySQL or Redis. Normally, during development, these dependencies are installed locally—a step that then needs re-doing when moving to a production setup. You can avoid these installation and configuration parts by using Docker Compose.

Once set up, you can bring all of these containers/dependencies up and running with a single docker-compose up command.