Docker:

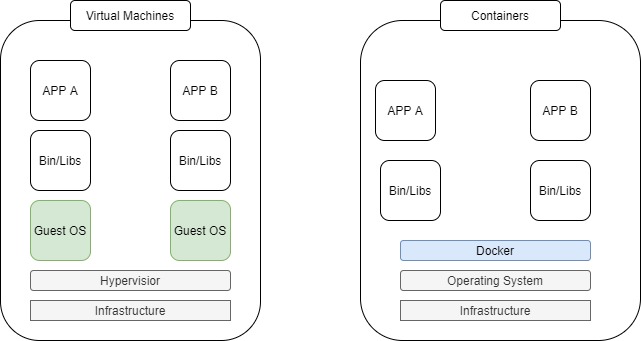
1. What is Docker?

* Docker is a containerization platform that enables you to build, test and deploy software solutions quickly and reliably.
* Docker host can run multiple containers, with degree of isolation between the containers.
* A container is a standard unit which provides a single point of service.
* Multiple containers when provided as a Services, create a scalable and reliable solution.

1. What are containers?

* A running instance of Docker Image.
* Packages all the code and its dependencies.
* OS level virtualization; containers share the operating system of the host.

1. Is Docker same as Virtual Machine?

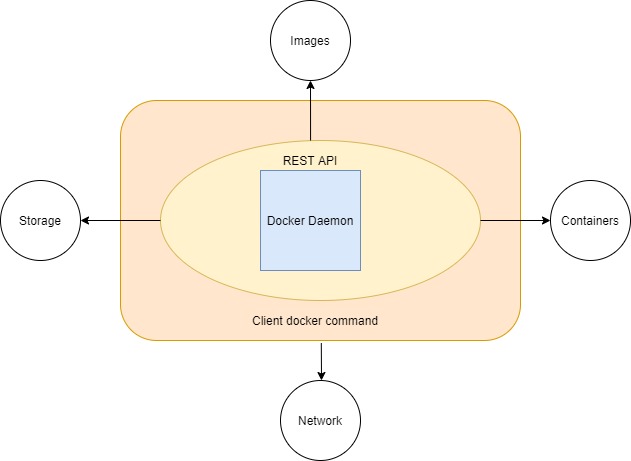


* Docker underlying architecture is fundamentally different from Virtual Machine.
* Virtual Machine run on virtualized hardware resources, with each VM having its own operating system.
* Docker Containers share the host OS Kernel, with each container having its own set of binaries and libraries.
* VM is abstraction of physical hardware, whereas containers are an abstraction at application layer.
* Docker containers are executed with the Docker engine rather than the hypervisor.
* Each VM has a full of OS with its own kernel, memory management and virtual device drivers.
* Docker containers are able to share a host kernel and share application libraries.

1. Why Docker?

* It’s convenient way to package applications, it bundles all dependencies and binaries of application. There is no need for additional software installation for applications.
* Development can create ready-to-run containerized applications.
* Provides consistent computing environment.
* Works well in on-premises as well as cloud environment.
* Lightweight and faster alternative to virtual machines.

1. Docker Architecture?



Docker engine gets installed in on the Docker host.

It’s a client server architecture based application.

Which has the following components:

* A daemon process (dockerd), which is continuous running program.
* A REST API to communicate with the daemon process and provide instructions.
* Client (docker command),command line interface

1. What are Docker Objects?

* Images: Read only template with instructions for creating a Docker container.
* Container: Runnable instance of an image.
* Networks: Network interface to connect to the container to external networks using the host machine’s network connection.
* Volumes: Mechanism for persisting data generated by and used by the containers.
* Registry: Private or public registry of docker image.
* Service: enables multi-host, multi-container deployment.

1. Docker Container format/Architecture?

The default container format is libcontainer and it combines the following:

Namespaces-

* Wraps a global system resource in an abstraction (isolated workspace).
* In short, limits what a container can see (own view of the system)
* Process ID, N/W Interfaces, Inter-process communication, Filesystem mount points

Control groups (cgroups) –

* Share available hardware resources to containers and optionally enforce limits and constraints
* In short, limits how much a container can use (metering & limiting)
* Memory, CPU, block I/O, Network

Union Filesystem (Union FS)-

* Operate by creating layers, making them very lightweight and fast.
* Union mounting is a way of combining multiple directories into one that appears to contain their combined contents.

1. What are Images and containers?

Images-

* Read-only template with instructions for running a Docker container.
* Images are the build or packaging part of Docker’s life cycle.
* Can be consider as a source code for the containers.
* Highly portable and can be shared, stored and updated.
* You can create your own image or you can use those created by others and published in a registry.
* One image can be based on another image, with some additional customization.

Creating Images:

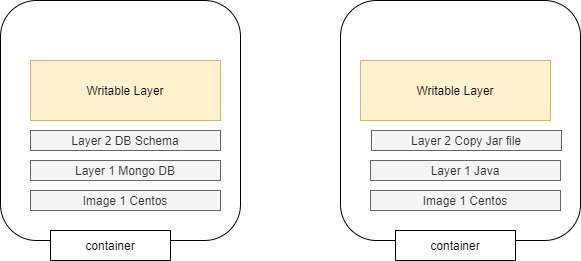
* From Dockerfile providing step by step instructions.
* Using docker commit command.

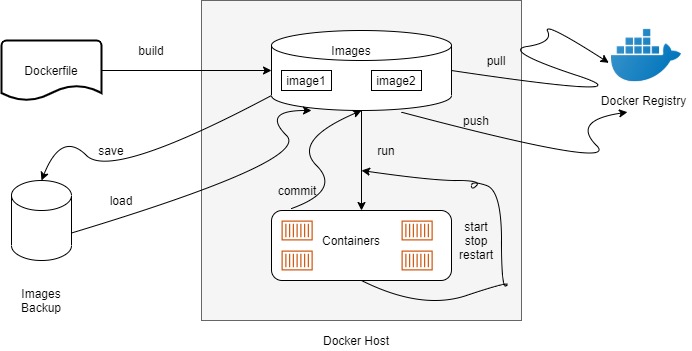
Distributing images:

* Using the default Docker Hub registry.
* You can create a public or private registry
* Save images to archive (tar files) and share

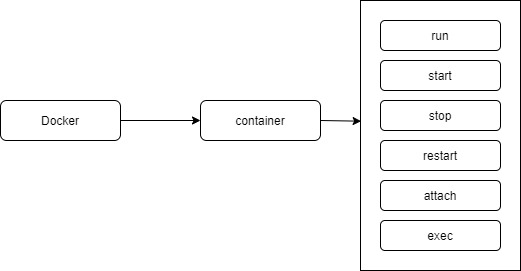
Containers-

* Containers are the running or execution aspect of Docker’s life cycle.
* Contains one or more running processes in a self-contained environment.
* Wraps up an application into its own isolated box
* An application in its container, has no knowledge of any other applications or processes that exists outside of its box.
* A running container can’t touch the original image itself nor the filesystem of the host.
* Changes made in a container are preserved in that container itself and don’t affect the original image.
* Writeable layer is available in container, but not in image.
* All writes to the container that add new or modify existing data are stored here.
* When the container is deleted, the writable layer is also deleted.
* Multiple containers can access the same underlying image and yet have their own data state.





1. What are the different commands for containers?



1. What is container top, stats and inspect commands?

* Docker top: docker top <containerid> OR <containername>: Display the running processes of a container.
* Docker stats command: docker stats <containerid> OR <containername>: Display a live stream of container(s) resource usage statistics.
* Docker inspect command: docker inspect <containerid> OR <containername>: Return low level information of Docker objects. It is in json format.

1. How to delete container?

docker rm <containerid> OR <containername>: Only stop container can remove. We can’t remove a running container.

1. How container delete itself once it’s stopped?

Command docker run –rm command delete the container once it’s stopped.

1. Default network available for container?

Default available networks in docker host is as follows:

1. Bridge

2. Host

3. None

Command docker network ls display all available networks in docker host

Default network for container is Bridge network hence if we don’t specify any network while running container, it used docker0 or Bridge network.

1. What is Host network?

Containers behave just as any other process running in the Docker host.

The host network adds a container on the host’s network stack.

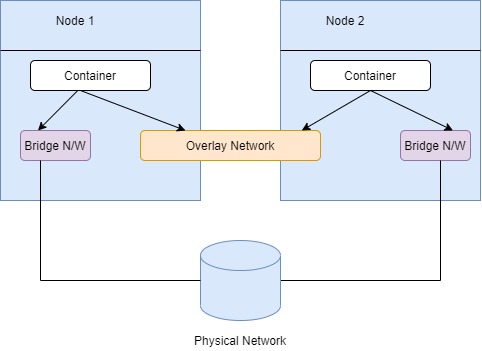
There is no isolation between the host machine and the container

Doesn’t perform any operations on incoming traffic.

Command: docker run –-itd --rm –name=nginx –network=host nginx

1. What is overlay network?

* An overlay network is a computer network that is built on top of another network.
* Creates distributed network among multiple Docker daemon hosts, on top of the host-specific networks.
* Allows containers, connected it, to communicate securely.
* Multiple new networks can be created.
* Networks can be added or removed from containers, without need to restart.
* In case of multiple networks, external connectivity is from the first network.



1. Explain Docker network commands?

New Bridge and overlay networks can be created.

* docker network create –driver=bridge new\_bridge1
* docker network create –driver=bridge –subnet=172.99.11.0/24 new\_bridge2
* docker network ls
* docker network inspect <network-name> OR <network-id>
* docker network rm <network-name> OR <network-id>

Pre-defined network can’t be deleted. Only custom network can be deleted.

We can’t remove any custom network, if that network is bind with active container.

docker network prune: This command removed all custom networks, doesn’t have any active containers.

1. Exercise : Disconnect default bridge network and connect new bridge n/w to container

1. docker run –itd –name=testnginx –net=bridge –p 80 nginx

2. docker inspect <container-id> (check the IP Address and external connectivity)

3. docker network create –driver=bridge –subnet=172.99.101.0/24 –gateway=172.99.101.1 new\_bridge3

4. docker network disconnect bridge <container-id>

5. docker network connect new\_bridge3<container-id>

1. How to launch container in desired networks?

Docker only allows a single network to be specified with the docker run command.

Docker run –itd –name=<container-name> --network=<network> -p 80 nginx

Further networks can then be added to the running container.

docker network connect <networked OR networkname> <containerid OR containername>

If a container needs to be connected to multiple networks before it runs, then all networks should be connected to a created container and start later.

docker create –itd --name=<container-name> --network=<network> -p 80 nginx

docker network connect <networked OR networkname> <containerid OR containername>

docker start <containerid OR containername>

1. What are the different repositories?

There are two different types of repositories.

1. User Repository: contains images contributed by docker user. Image name contains user name.

2. Top level Repository: It’s managed by docker and it has only image name.

1. What is Dockerfile?

* The Docker file contains a series of instructions paired with arguments.
* Each instruction should be in upper-case and be followed by an argument.
* Instructions in Dockerfile are processed from the top down.
* Each instruction adds a new layer to the image and then commits the image.

Upon running, changes made by an instructions make it to the container.

If Dockerfile stops for some reason, you will be left with a working image.

Images are built using docker build command.

Build context is a set of files at either

Specified location path: directory on your local file system.

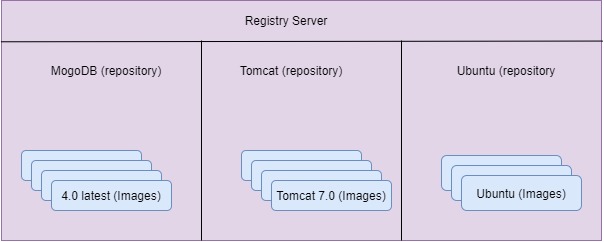
URL: Git Repository location

The build is run by the Docker daemon, not by the CLI.

First thing build process does, is send the entire context (recursively) to the daemon.

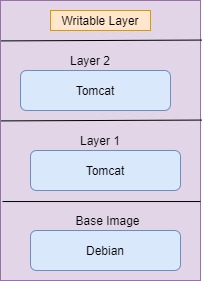
1. What is docker repositories and registry?

* Repositories hold images.
* Registry stores repositories.



1. Explain Docker Image Layered file system?

* Docker uses UnionFS for implementing the layered docker images.
* The resulting file system contains files and subdirectories from any or all of the underlying file systems.
* Any update on the image, adds a new layer.
* All changes made to the running container are written to the writeable layer.



1. What are the different docker images cli commands?

Listing Image:

1. Listing docker images: docker images OR docker image ls

2. Get full length of docker image id: docker image ls –no-trunc

3. Filter the docker images: docker image ls --filter “before=<Image ID>”

4. Get the dangling images: docker image ls –filter “dangling=true”

5. Format the output: docker image ls --format=”{{.ID}}:{{.Repository}}”

Deleting Image:

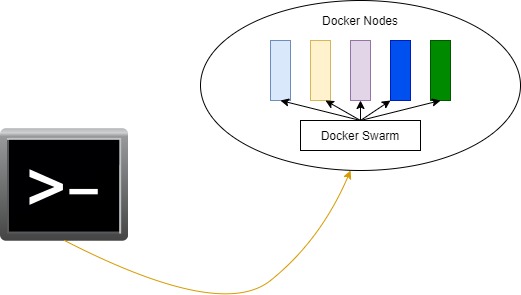
1. What is docker compose?

* Compose is a tool for defining and running multi-container Docker applications.
* With compose, you use a YAML file to configure your application’s services.
* With a single command, you create and start all the services from your configuration.

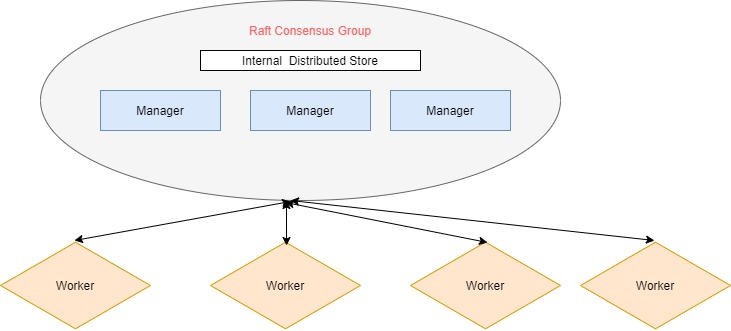
Compose is a basically three step process:

* Build container image as part of Compose OR use a pre-created image.
* Define the services that makeup the application.
* Start the entire application with a single command.

1. Docker swarm concepts?



* Multi-container, multi-machine applications.
* A swarm is a group of machines that are running Docker and joined into a cluster.
* Docker CLI to create and manager a swarm; also to deploy application services on swarm
* Cluster management integrated with Docker Engine.
* Secure by default
* Built using swarmkit



* A swarm consists of multiple Docker hosts which run in swarm mode and act as managers and workers.
* Manager node handles cluster management and orchestration functions.
* Worker node run swarm services.
* By default, manager nodes also run services as worker nodes.
* Standalone containers can still run on any of the Docker hosts participating in the swarm.

**Raft Implementation:**

* Managers maintain a consistent internal state of the entire swarm.
* If the swarm loses the quorum of managers, the swarm can’t perform management tasks.
* You should maintain an odd number of managers to support manager node failures.
* N manager cluster tolerates the loss of at most (N-1)/2 managers.

**Deploy services to a swarm:**

Service is a definition of tasks to execute on Manager or Worker nodes.

Declarative Model for services.

Scaling

Desired state reconciliation

Service Discovery

Rolling Updates

Load Balancing

Internal DNS Component