Docker

Docker

- A centralized platform for packaging, deploying, and running applications.
- Challenge - developers faced was that a particular code would run in the developer's system but not in the user's system.
- Aim : to develop docker was to help developers to
  - develop applications easily,
  - ship them into containers, and
  - can be deployed anywhere.

- First released in March 2013.
- used in the Deployment stage of the software development life cycle
  - can efficiently resolve issues related to the application deployment.

What is Docker?

- an open-source centralized platform designed to create, deploy, and run applications.
- uses container on the host's operating system to run applications.
- allows applications to use the same Linux kernel as a system on the host computer, rather than creating a whole virtual operating system.
- Containers ensure that our application works in any environment like development, test, or production.
- That is it automates the deployment of software applications inside **containers**
- by providing
  - an additional layer of abstraction and
  - automation of **OS-level virtualization** on Linux.

The key benefit of Docker

- it allows users to **package an application with all of its dependencies into a standardized unit** for software development.
- Unlike virtual machines, containers do not have high overhead and hence enable more efficient usage of the underlying system and resources.

Docker components

- Docker client
- Docker server
- Docker machine
- Docker hub
- Docker compose etc.

Docker Containers

- lightweight alternatives of the virtual machine.
- allow developers to package up the application with all its libraries and dependencies, and ship it as a single package.
- The advantage of using a docker container
  - We don't need to allocate any RAM and disk space for the applications.
  - It automatically generates storage and space according to the application requirement.

Virtual Machine

- A software that allows us to install and use other operating systems (Windows, Linux, and Debian) simultaneously on our machine.
- The OS in which virtual machine runs are called virtualized operating systems.
- These virtualized OSs can run programs and preforms tasks that we perform in a real operating system.

Containers Vs. Virtual Machine

Containers	Virtual Machine
Integration in a container is faster and cheap.	Integration in virtual is slow and costly.
No wastage of memory.	Wastage of memory.
It uses the same kernel, but different distribution.	It uses multiple independent operating systems.

Why Docker

- designed to benefit both the Developer and System Administrator.
- reasons to use Docker -
  - allows us to easily install and run software without worrying about setup or dependencies.
  - Developers use Docker to eliminate machine problems, i.e. "but code is worked on my laptop." when working on code together with co-workers.
  - Operators use Docker to run and manage apps in isolated containers for better compute density.
  - Enterprises use Docker to securely built agile software delivery pipelines to ship new application features faster and more securely.
  - Since docker provides a great platform for development, we can efficiently increase our customer's satisfaction.

Advantages of Docker

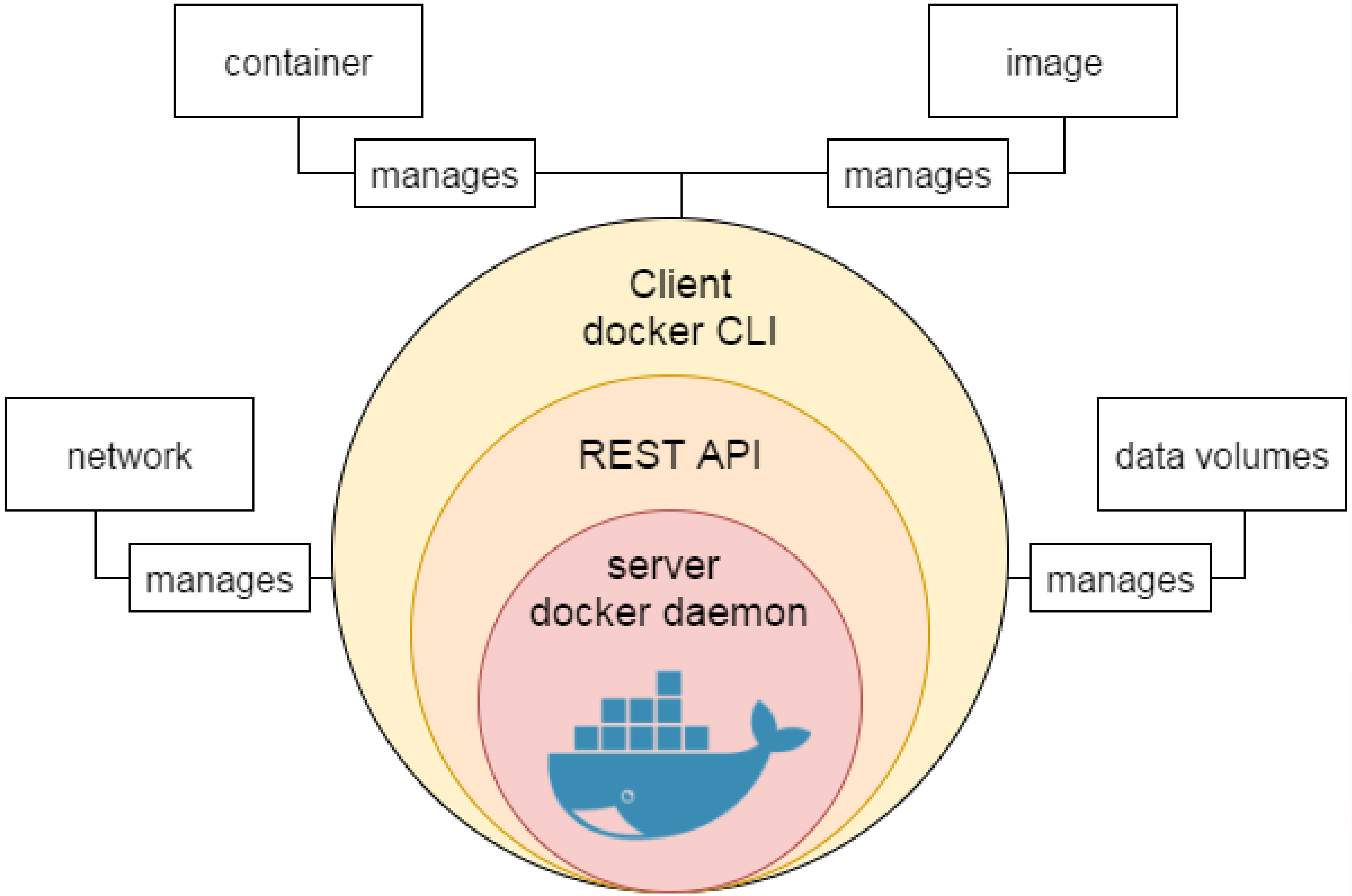
- Fast : runs the container in seconds instead of minutes.
- Uses less memory.
- Provides lightweight virtualization.
- Does not a require full operating system to run applications.
- Uses application dependencies to reduce the risk.
- Allows us to use a remote repository to share your container with others.
- Provides continuous deployment and testing environment.

Disadvantages of Docker

- increases complexity due to an additional layer.
- it is difficult to manage large amount of containers.
- Some features such as container self -registration, containers self-inspects, copying files form host to the container, and more are missing in the Docker.
- not a good solution for applications that require rich graphical interface.
- Does not provide cross-platform compatibility means if an application is designed to run in a Docker container on Windows, then it can't run on Linux or vice versa.

Docker Engine

- A client server application that contains the following major components.
  - A server which is a type of long-running program called a daemon process.
  - The REST API is used to specify interfaces that programs can use to talk to the daemon and instruct it what to do.
  - A command line interface client.


container
image
manages
manages
Client
docker CLI
network
REST API
data volumes
server
docker daemon
manages
manages

Docker Features

- Easy and Faster Configuration
- Application Isolation
- Swarm
- Routing Mesh
- Increase productivity
- Services
- Security Management

Easy and Faster Configuration

- A key feature of docker that helps us to configure the system easily and faster.
- We can deploy our code in less time and effort.
- As Docker can be used in a wide variety of environments, the requirements of the infrastructure are no longer linked with the environment of the application.

Increase productivity

- By easing technical configuration and rapid deployment of application.
- Docker not only helps to execute the application in isolated environment but also it has reduced the resources requirement.

Application Isolation

- It provides containers that are used to run applications in isolation environment.
- Each container
  - is independent to another and
  - Allows us to execute any kind of application.

Swarm

- It is a clustering and scheduling tool for Docker containers.
- Swarm uses the Docker API as its front end, which helps us to use various tools to control it.
- It also helps us to control a cluster of Docker hosts as a single virtual host.
- It's a self-organizing group of engines that is used to enable pluggable backends.

Routing Mesh

- It routes the incoming requests for published ports on available nodes to an active container. This feature enables the connection even if there is no task is running on the node.

Services

- A list of tasks that lets us specify the state of the container inside a cluster.
- Each task represents one instance of a container that should be running and Swarm schedules them across nodes.

Security Management

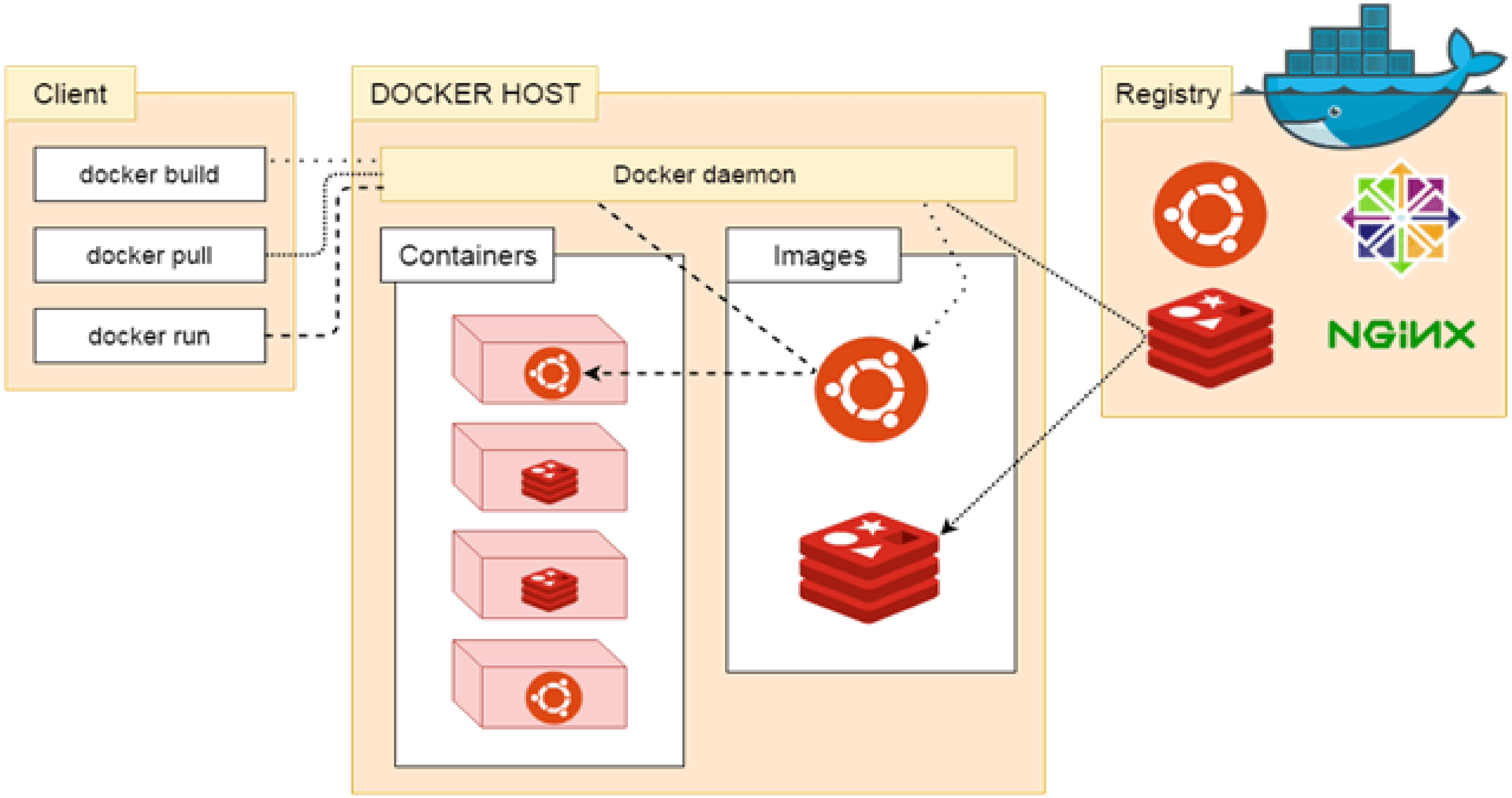
- It allows us to save secrets into the swarm itself and then choose to give services access to certain secrets.
- It includes some important commands to the engine like secret inspect, secret create etc.

Docker daemon?

- runs on the host operating system.
- It is responsible for running containers to manage docker services.
- It communicates with other daemons.
- offers various Docker objects such as images, containers, networking, and storage.

Docker architecture

- Docker follows Client-Server architecture,
- includes the three main components that are
  - Docker Client,
  - Docker Host, and
  - Docker Registry.


Client
docker build
docker pull
docker run
DOCKER HOST
Docker daemon
Containers
Images
Registry
NGINX

Docker Client

- uses commands and REST APIs to communicate with the Docker Daemon (Server).
- When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon.
- Docker daemon receives these commands from the docker client in the form of command and REST API's request.
- Docker Client has an ability to communicate with more than one docker daemon.
- Docker Client uses Command Line Interface (CLI) to run the following commands -
  - docker build
  - docker pull
  - docker run

Docker Host

- used to provide an environment to execute and run applications.
- contains
  - the docker daemon,
  - images,
  - containers,
  - networks, and
  - storage.

Docker Registry

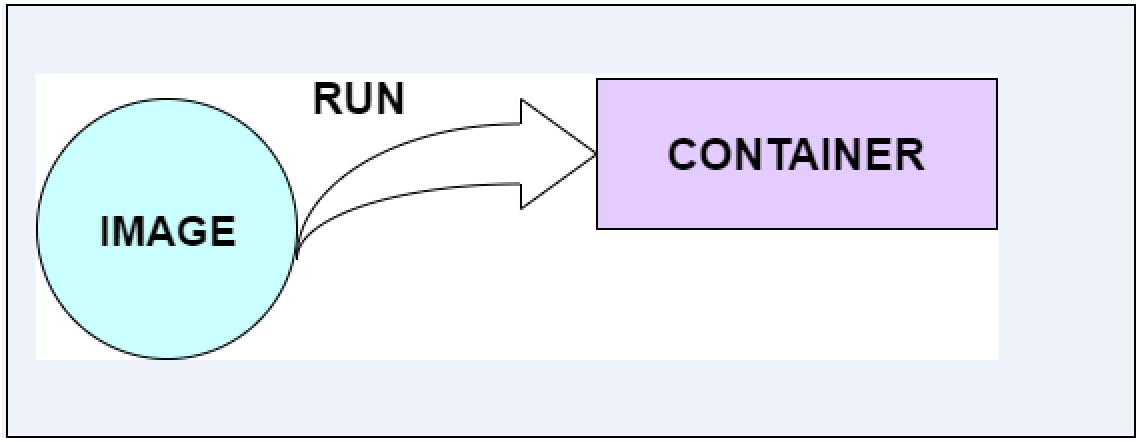
- Docker Registry manages and stores the Docker images.
- two types of registries in the Docker -
  - Public Registry - also called as Docker hub.
  - Private Registry - used to share images within the enterprise.

Docker Objects

- Docker Images
  - **read-only binary templates** used to create Docker Containers.
  - It uses a private container registry to share container images within the enterprise and also uses public container registry to share container images within the whole world. Metadata is also used by docket images to describe the container's abilities.
- Docker Containers
  - Containers are the structural units of Docker, which is used to hold the entire package that is needed to run the application. The advantage of containers is that it requires very less resources.

image vs container

- image is a template, and the container is a copy of that template.



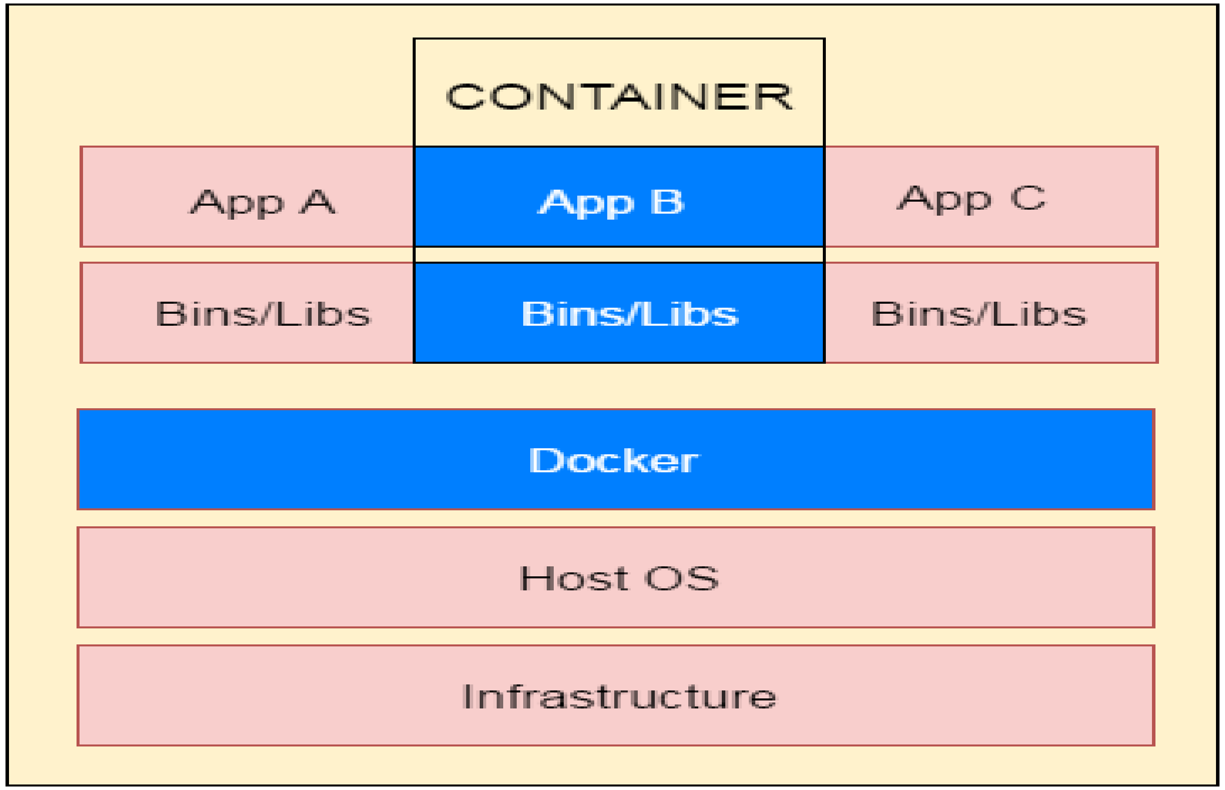
Docker Networking

- Using Docker Networking, an isolated package can be communicated.
- Docker contains the following network drivers -
  - Bridge - Bridge is a default network driver for the container. It is used when multiple docker communicates with the same docker host.
  - Host - It is used when we don't need for network isolation between the container and the host.
  - None - It disables all the networking.
  - Overlay - Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker host.
  - Macvlan - Macvlan is used when we want to assign MAC addresses to the containers.

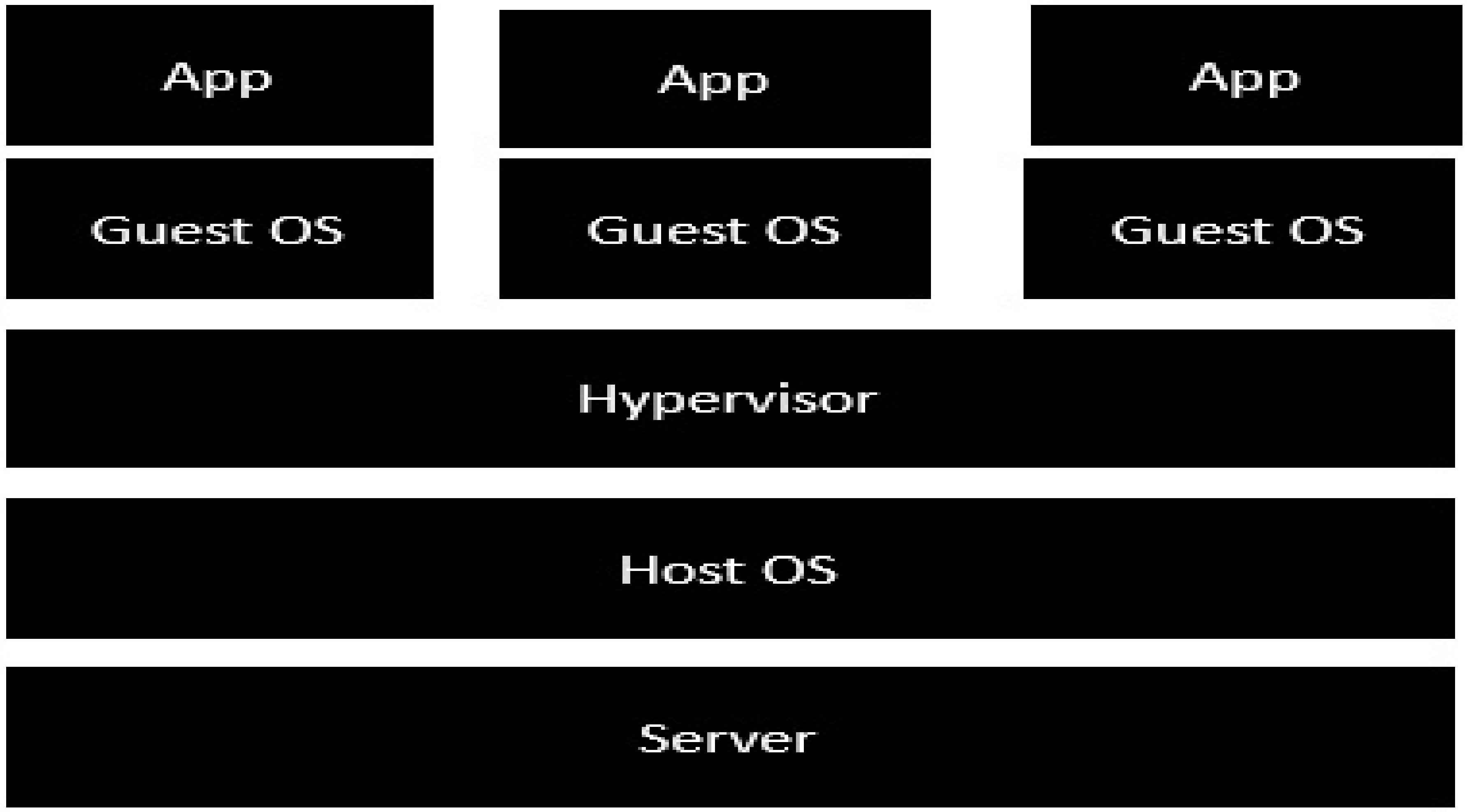
Docker Storage

- used to store data on the container.
- Docker options for the Storage -
  - Data Volume - Data Volume provides the ability to create persistence storage. It also allows us to name volumes, list volumes, and containers associates with the volumes.
  - Directory Mounts - It is one of the best options for docker storage. It mounts a host's directory into a container.
  - Storage Plugins - It provides an ability to connect to external storage platforms.

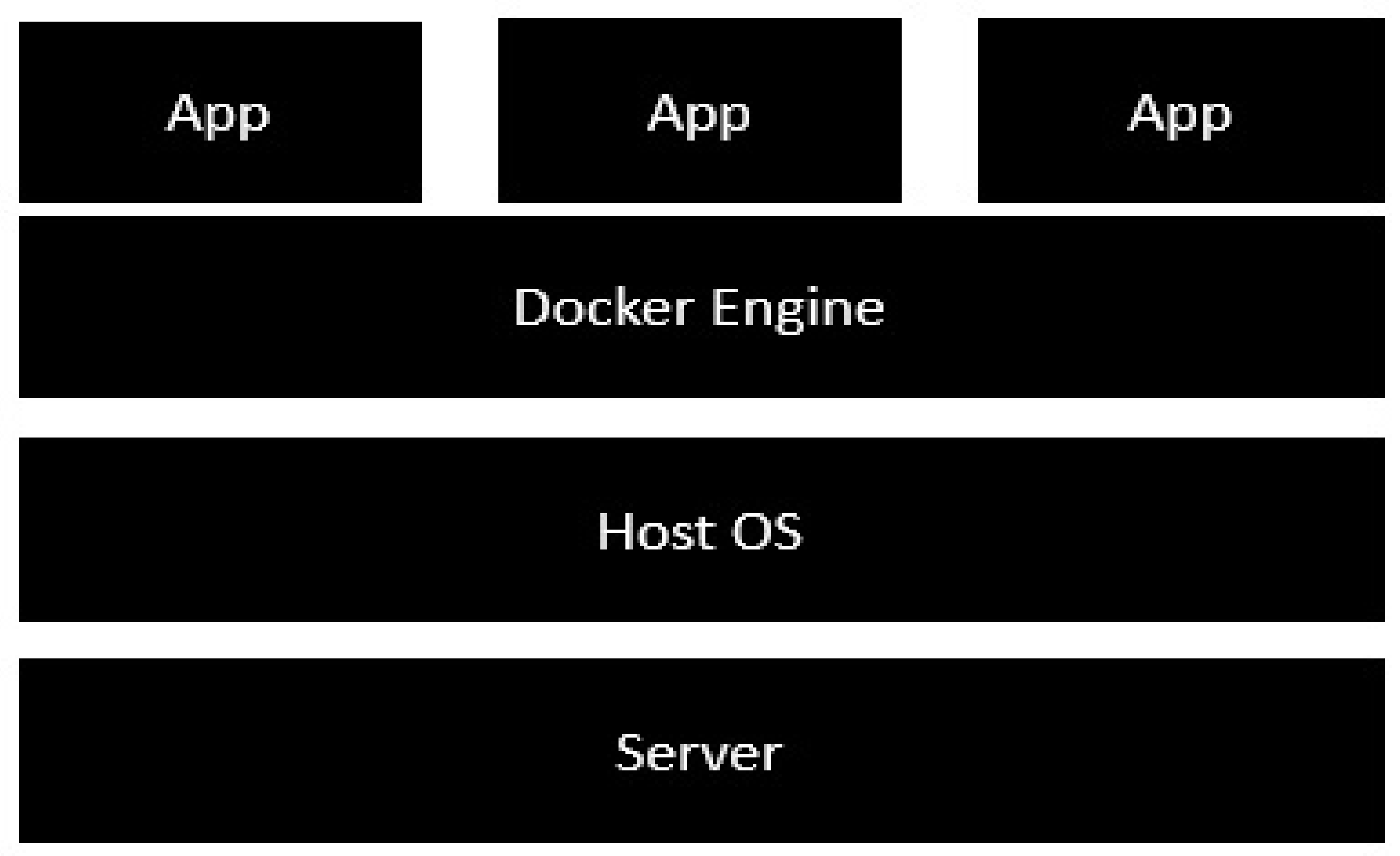
Docker-container



Architecture - virtualization



Docker



Docker Container and Image

- Docker container is a running instance of an image.
  - We can use CLI commands to run, start, stop, move, or delete a container.
  - We can also provide configuration for the network and environment variables.
  - Docker container is an isolated and secure application platform, but it can share and access to resources running in a different host or container.

Docker Container and Image

- An image is a read-only template with instructions for creating a Docker container.
- A docker image is described in text file called a Dockerfile, which has a simple, well-defined syntax.
- An image does not have states and never changes.
- Docker Engine provides the core Docker technology that enables images and containers.
- We can understand container and image with the help of the following command.
- $ docker run hello-world

Docker Container and Image

- command docker run hello-world has three parts.
  - **docker:**
    - docker engine, used to run docker program.
    - tells to the OS that we are running docker program.
  - **run:**
    - subcommand , used to create and run a docker container.
  - **hello-world:**
    - name of an image. We need to specify the name of an image which is to load into the container.

Docker Dockerfile

- A Dockerfile is a text document that contains commands that are used to assemble an image.
- We can use any command that call on the command line.
- Docker builds images automatically by reading the instructions from the Dockerfile.

- The docker build command is used to build an image from the Dockerfile. You can use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

- $ docker build -f /path/to/a/Dockerfile .

Dockerfile Instructions

- not case-sensitive but you must follow conventions which recommend to use uppercase.
- Docker runs instructions of Dockerfile in top to bottom order.
- The first instruction must be FROM in order to specify the Base Image.

Dockerfile instructions

- A statement begin with # treated as a comment. You can use RUN, CMD, FROM, EXPOSE, ENV etc instructions in your Dockerfile.
- FROM
  - This instruction is used to set the Base Image for the subsequent instructions.
  - first instruction.
- LABEL
  - We can add labels to an image to organize images of our project. We need to use LABEL instruction to set label for the image.
- RUN
  - used to execute any command of the current image.
- CMD
  - used to execute application by the image. We should use CMD always in the following form
  - CMD ["executable", "param1", "param2"?]
  - preferred way to use CMD. There can be only one CMD in a Dockerfile. If we use more than one CMD, only last one will execute.

Dockerfile instructions

- COPY
  - used to copy new files or directories from source to the filesystem of the container at the destination.
- **Rules**
  - The source path must be inside the context of the build. We cannot COPY ../something /something because the first step of a docker build is to send the context directory (and subdirectories) to the docker daemon.
  - If source is a directory, the entire contents of the directory are copied including filesystem metadata.
- WORKDIR
  - The WORKDIR is used to set the working directory for any RUN, CMD and COPY instruction that follows it in the Dockerfile. If work directory does not exist, it will be created by default.
  - We can use WORKDIR multiple times in a Dockerfile.

What are containers?

- Containers take a different approach:
  - by leveraging the low-level mechanics of the host operating system,
  - containers provide most of the isolation of virtual machines at a fraction of the computing power.

Container

- A container is simply another process on our machine
  - that has been isolated from all other processes on the host machine.
  - This isolation leverages kernel namespaces and cgroups

Container

- When running a container, it uses an isolated filesystem.
- This custom filesystem is provided by a **container image**.
- As the image contains the container’s filesystem
  - it must contain everything needed to run an application
    - all dependencies,
    - configuration,
    - scripts,
    - binaries, etc.
  - The image also contains other configuration for the container, such as environment variables, a default command to run, and other metadata.

Why use containers?

- Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run.
- This decoupling allows container-based applications to be deployed easily and consistently, regardless of whether the target environment is a private data center, the public cloud, or even a developer's personal laptop.
- This gives developers the ability to create predictable environments that are isolated from the rest of the applications and can be run anywhere.
- From an operations standpoint, apart from portability containers also give more granular control over resources giving your infrastructure improved efficiency which can result in better utilization of your compute resources.

Docker engine

- it is designed to work on various operating systems.
- Can be installed on Windows and Linux systems.
- Some Docker commands on the Windows OS.

Docker Compose

- A tool that was developed to help define and share multi-container applications.
- With Compose, we can create a YAML file to define the services and with a single command, can spin everything up or tear it all down.
- Advantage of using Compose :
  - We can define our application stack in a file,
  - keep it at the root of your project repo (it's now version controlled),
  - and easily enable someone else to contribute to our project.
  - Someone would only need to clone our repo and start the compose app.
  - We might see quite a few projects on GitHub/GitLab doing exactly this now.