Parul® University

Cloud Computing and Security

Prof. Meet Hudani
Cyber Security Trainer

CHAPTER-2

Cloud Application Architecture and security

What is Cloud Container?

•Containers are packages of software that contain all of the necessary elements to run in any environment. In this way, containers virtualize the operating system and run anywhere,from a private data centre to the public cloud or even on a developer’s personal laptop. From Gmail to YouTube to Search, everything at Google runs in containers.

•Containerization allows our development teams to move fast, deploy software efficiently, and operate at an unprecedented scale.

•Containers are light weight packages of your application code together with dependencies such as specific versions of programming language runtimes and libraries required to run your software services.

•Containers make it easy to share CPU, memory, storage, and network resources at the operating systems level and offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run.

•Containers rely on virtual isolation to deploy and run applications that access a shared OS kernel without the need for VMs.

•Containers hold all the necessary components, such as files, libraries and environment variables, to run desired software without worrying about platform compatibility. The host OS constrains the container’s access to physical resources so a single container cannot consume all of a host’s physical resources.

•Cloud containers is that they are designed to virtualize a single application.
. •For example, you have a MySQL container, and that’s all it does—it provides a virtual instance of that application.

Parul® University

Role of Containers in Cloud Computing?

•Containers are a common option for deploying and managing software in the cloud.

•Containers are used to abstract applications from the physical environment in which they are running. A container packages all dependencies related to a software component, and runs the main isolated environment.

•Containerized applications are easier to migrate to the cloud. Containers also make it easier to leverage the extensive automation capabilities of the cloud—they can easily be deployed, cloned or modified using APIs provided by the container engine or orchestrator.

What are containers used for?

➢ **Agile development**

•Containers allow your developers to move much more quickly by avoiding concerns about dependencies and environments.

➢ **Efficient operations**

•Containers are light weight and allow you to use just the computing resources you need. This lets you run your applications efficiently.

➢ **Run anywhere**

•Containers are able to run virtually anywhere.Wherever you want to run your software, you can use containers.

Use Cases of Containers in the Cloud

•The following use cases are especially suitable for running containers in the cloud:

- **Microservices**—
  - Containers are lightweight ,making them well suited for applications with microservices architectures consisting of a large number of loosely coupled, independently deployable services.

- **DevOps**—
  - Many DevOps teams build applications using a microservices architecture, and deploy services using containers. Containers can also be used to deploy and scale the DevOps infrastructure itself, such CI/CD tools.

- **Hybrid and multi-cloud**
  - For organizations operating in two or more cloud environments, containers are highly useful for migrating workloads. They are a standardized unit that can be flexibly moved between on- premise data centers and any public cloud.

- **Application modernization**—
  - a common way to modernize a legacy application is to containerize it, and move it as is to the cloud (a model known as “lift and shift”).

Parul® University

What is CI/CD tools?

•CI and CD stand for continuous integration and continuous delivery/continuous deployment. In very simple terms, CI is a modern software development practice in which incremental code changes are made frequently and reliably.

•Automated build-and-test steps triggered by CI ensure that code changes being merged in to the repository are reliable. The code is then delivered quickly and seamlessly as a part of the CD process.

•In the software world, the CI/CD pipeline refers to the automation that enables incremental code changes from developers 'desktops to be delivered quickly and reliably to production.

How Do Cloud Containers Work?

•Containers rely on isolation, controlled at the operating system kernel level, to deploy and run applications.

•Containers share the operating system kernel, and do not need to run a full operating system—they only need to run the necessary files, libraries and configuration to run workloads. The host operating system limits the container's ability to consume physical resources.

•In the cloud, a common pattern is to use containers to run an application instance. This can be an individual microservice, or a backend application such as a database or middleware component.

•Containers make it possible to run multiple applications on the same cloud VM, while ensuring that problems with one container do not affect other containers, or the entire VM.

Benefits/Features of containers?

- **Separation of responsibility:**

• Containerization provides a clear separation of responsibility, as developers focus on application logic and dependencies, while IT operations teams can focus on deployment and management instead of application details such as specific software versions and configurations.

- **Workload portability**

• Containers can run virtually any where, greatly easing development and deployment :on Linux, Windows, and Mac operating systems; on virtual machines or on physical servers; on a developer's machine or in data center son-premises, in the public cloud.

- **Application isolation**

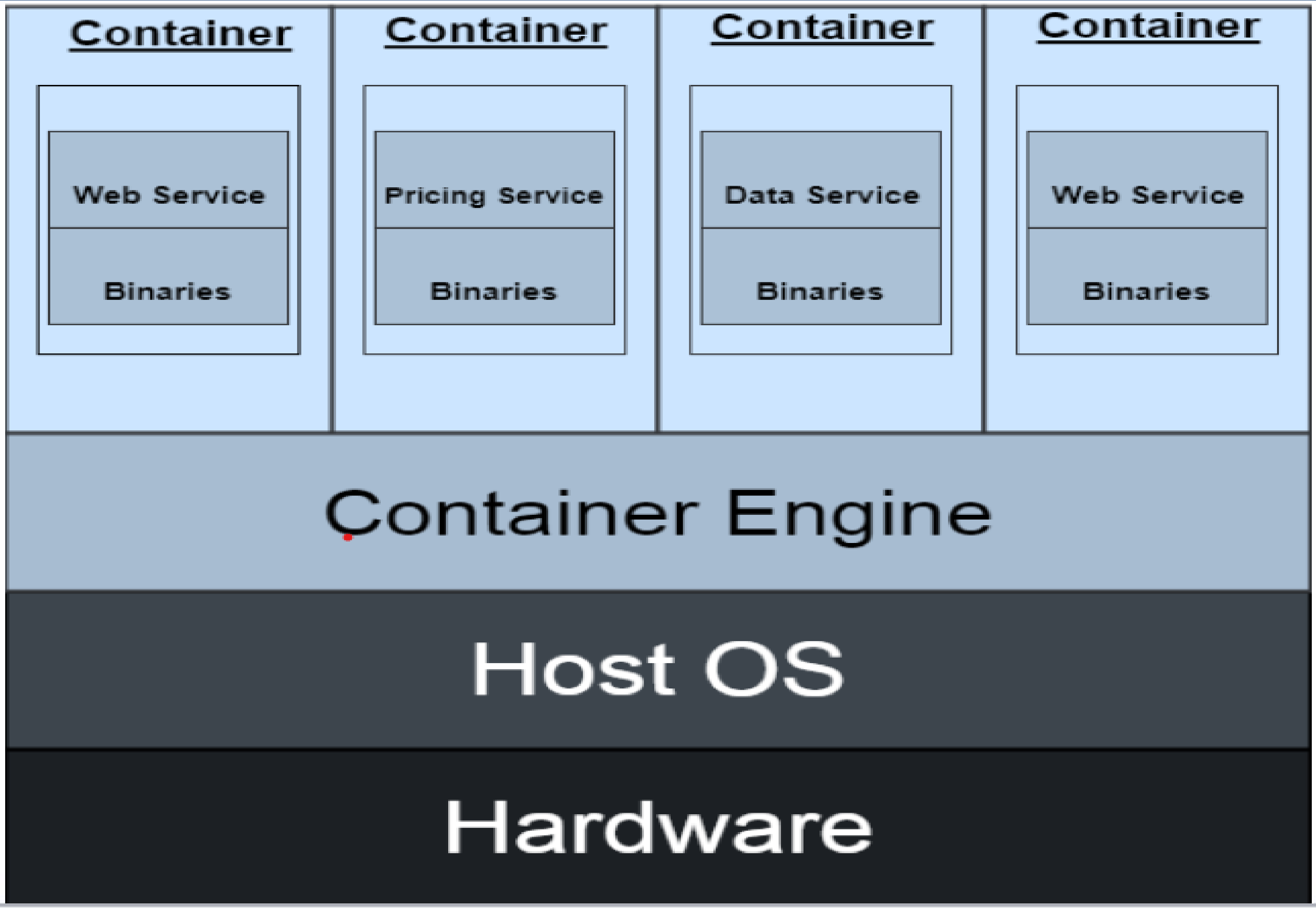
• Containers virtualize CPU, memory, storage, and network resources at the operating system level, providing developers with a view of the OS logically isolated from other applications.

Container Architecture

- A containerized architecture makes it possible to package software and its dependencies in an isolated unit, called a container, which can run consistently in any environment.

- Containers are truly portable, unlike traditional software deployment, in which software could not be moved to another environment without errors and incompatibilities.

- Containers are similar to virtual machines in a traditional virtualized architecture, but they are more light weight–they require less server resources and are much faster to startup.


Container
Container
Container
Container
Web Service
Binaries
Pricing Service
Binaries
Data Service
Binaries
Web Service
Binaries
Container Engine
Host OS
Hardware

Container Architecture

- Technically, a container differs from a virtual machine because it shares the operating system kernel with other containers and applications, while a virtual machine runs a full virtual operating system.

- Containerization helps developers and operations teams manage and automate software development and deployment. Containerization makes it possible to define infrastructure as code(IaC) specifying required infrastructure in a simple configuration file and deploying it as many times as needed. It is especially useful for managing microservices applications, which consist of a large number of independent components.

•**Container Engine**

•**Container Orchestrators**

•**Managed Kubernetes services**

Parul® University

Container Engine

•The container engine (of ten referred to as operating-system-level virtualization)is based on an operating system in which the kernel allows multiple isolated instances. Each instance is called a container, virtualization engine, or “jail”.

•Developers use containers to create a virtual host with isolated resources, and can deploy applications, configurations, and other dependencies with in a container.

•This reduces the administrative over head needed to manage applications, and makes them easy to deploy and migrate between environments. Developers can a loose containers to deploy applications in a hosted environment, more efficiently and with lower resource utilization than virtual machines.

•Examples of container engines include **Docker, CRI-O, Containerd ,and Windows Containers.**

Parul® University

Container Orchestrators

•Container orchestration software allows developers to deploy large numbers of containers and manage them at large scale, using the concept of container clusters.

•Orchestrators help IT admins automate the process of running container instances, provisioning hosts, and connecting containers into functional groups.

•With container orchestration, it is possible to manage the lifecycle of applications or ecosystems of applications consisting of large numbers of containers.

Features of Orchestrators

- Orchestrators can:
- Automatically deploy containers based on policies, application load and environment metrics.

•Identify failed containers or clusters and heal them.

•Manage application configuration.

•Connect containers to storage and manage networking.

•Improve security by restricting access in between containers, and between containers and external systems.

•Examples of orchestrators include **Kubernetes, Mirantis Kubernetes Engine,and OpenShift.**

Managed Kubernetes services

- Managed Kubernetes services add another level of management above container orchestrators. Setting up and managing a tool like Kubernetes is challenging and requires specialized expertise.

•These services allow organizations to provide container images and high-level scaling and operation policies, and automatically creates Kubernetes clusters.

•Clusters can be managed via APIs, web based consoles, or CLI commands. Managed Kubernetes is commonly offered on the public cloud, but there are platforms that can run in a non-premises data center as well.

•Examples of managed Kubernetes services are Amazon Elastic Kubernetes Service (EKS),Google Kubernetes Engine(GKE),Azure Kubernetes Service(AKS)and SUSE Rancher.

Advantages of a Containerized Architecture

•**Lower costs**—on infrastructure operations, because you can run many containers on a single virtual machine.

•**Scalability—**at the micro-service level eliminates the need to scale VMs or instances.

•**Instant replication**—of microservices, enabled through deployment sets and replicas.

•**Flexible routing**—you can set this up between services supported natively by containerization platforms.

•**Resilience**—when a container fails, it's easy to refresh/redeploy with a new container from the same image.

•**Full portability**—between on-premise locations and cloud environments.

•**OS independent—**there is no need to run an OS. All you need is to deploy a container engine on top of a host OS.

- **Fast deployment—**of new containers. You can also quickly terminate old containers using the same environment.

- **Lightweight—**since containers run without an OS,they are significantly lightweight and much less demanding than images.

- **Faster“ready to compute”**—you can start and stop containers within seconds—much faster than VMs

Containers vs. VMs

- You might already be familiar with VMs: a guest operating system such as Linux or Windows runs on top of a host operating system with access to the underlying hardware.
- Containers are often compared to virtual machines (VMs). Like virtual machines, containers allow you to package your application together with libraries and other dependencies, providing isolated environments for running your software services. However, the similarities end here as containers offer a far more lightweight unit for developers and IT Ops teams to work with, carrying a myriad of benefits.
- Containers are much more lightweight than VMs
- Containers virtualize at the OS level while VMs virtualize at the hardware level
- Containers share the OS kernel and use a fraction of the memory VMs require

Docker

- Docker is a software platform that allows you to build, test, and deploy applications quickly.
- Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run.
- Docker works by providing a standard way to run your code. Docker is an operating system for containers. Similar to how a virtual machine virtualizes (removes the need to directly manage) server hardware, containers virtualize the operating system of a server.
- Docker is installed on each server and provides simple commands you can use to build, start, or stop containers.

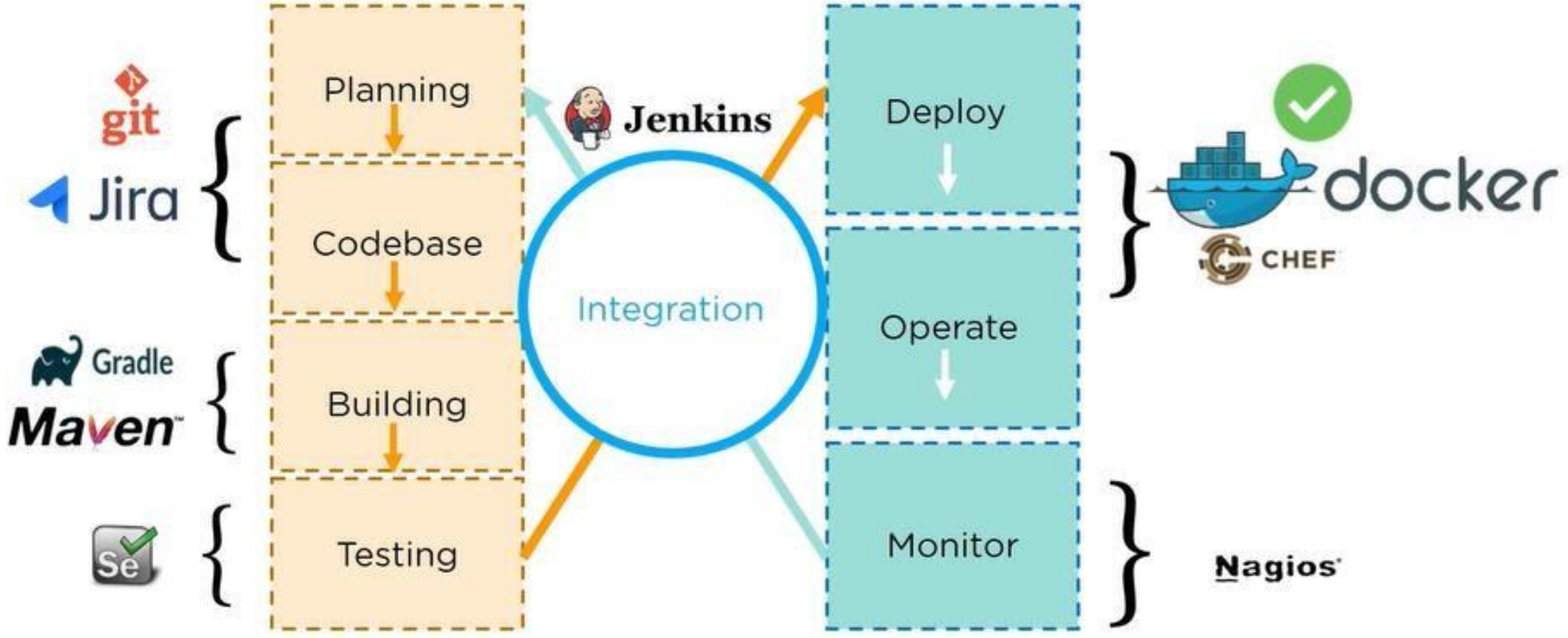
Docker is a tool which is used to automate the deployment of applications in lightweight containers so that applications can work efficiently in different environments

docker

- Multiple containers run on the same hardware
- Maintains isolated applications
- High productivity
- Quick and easy configuration

Note: Container is a software package that consists of all the dependencies required to run an application

DevOps is a collaboration between development and operation teams which enables continuous delivery of applications and services to our end users



Why Docker?

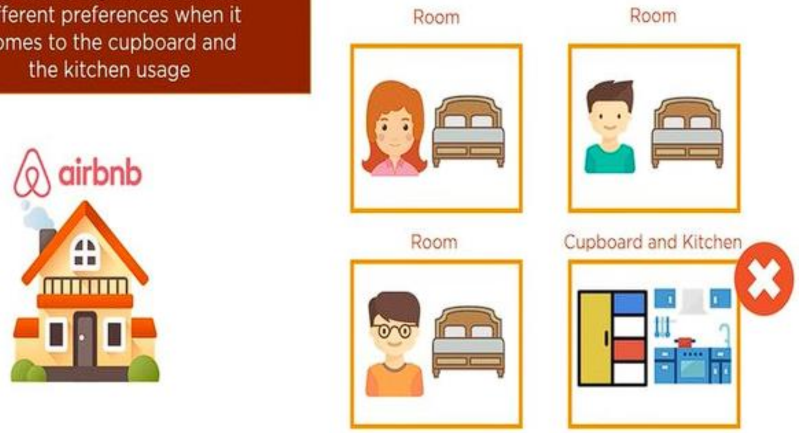
Criteria	Virtual Machine	Docker
OS support	Occupies a lot of memory space	Docker Containers occupy less space
Boot-up time	Long boot-up time	Short boot-up time
Performance	Running multiple virtual machines leads to unstable performance	Containers have a better performance as they are hosted in a single Docker engine
Scaling	Difficult to scale up	Easy to scale up
Efficiency	Low efficiency	High efficiency
Portability	Compatibility issues while porting across different platforms	Easily portable across different platforms
Space allocation	Data volumes cannot be shared	Data volumes can be shared and reused among multiple containers

VS

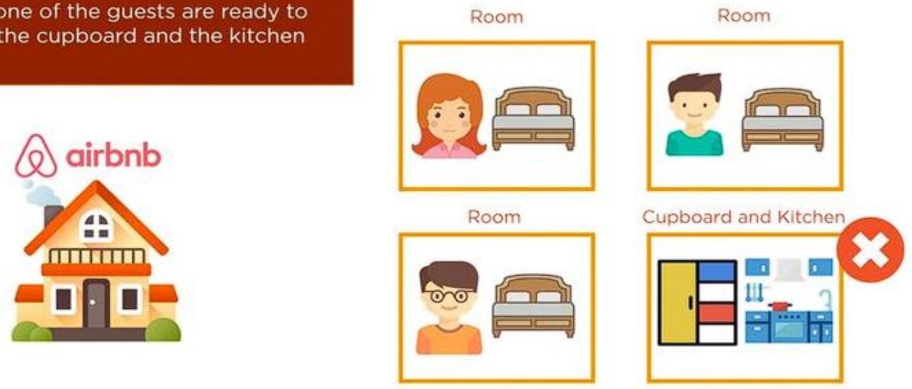
But, in the house, there are 3 rooms and only one cupboard and kitchen

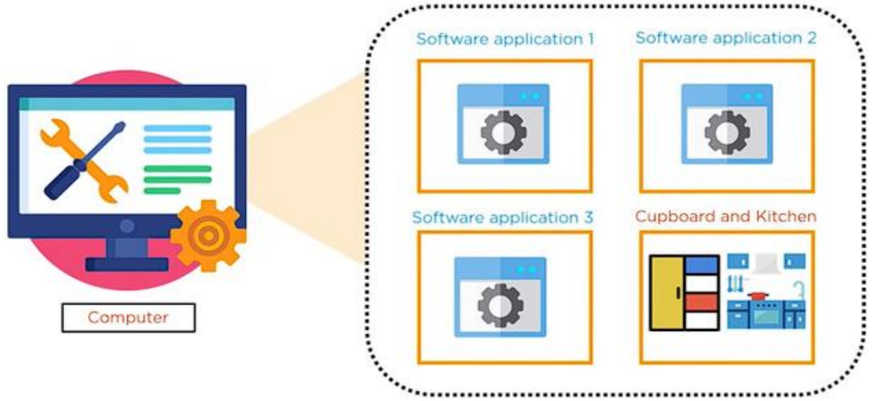
airbnb
Room
Room
Room
Cupboard and Kitchen

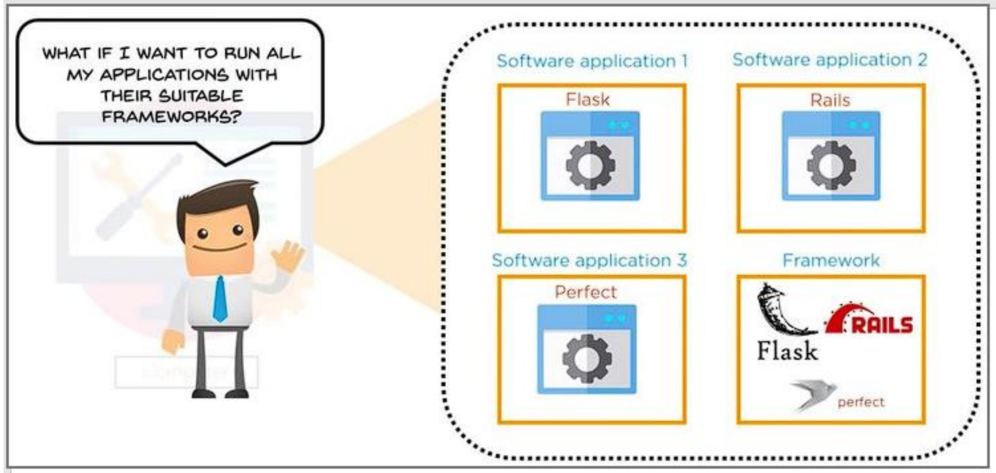
Because every individual has different preferences when it comes to the cupboard and the kitchen usage

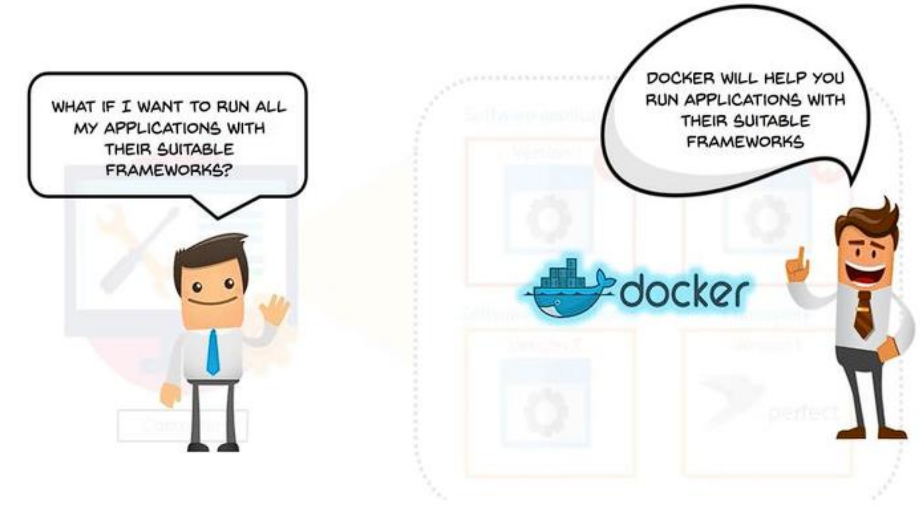


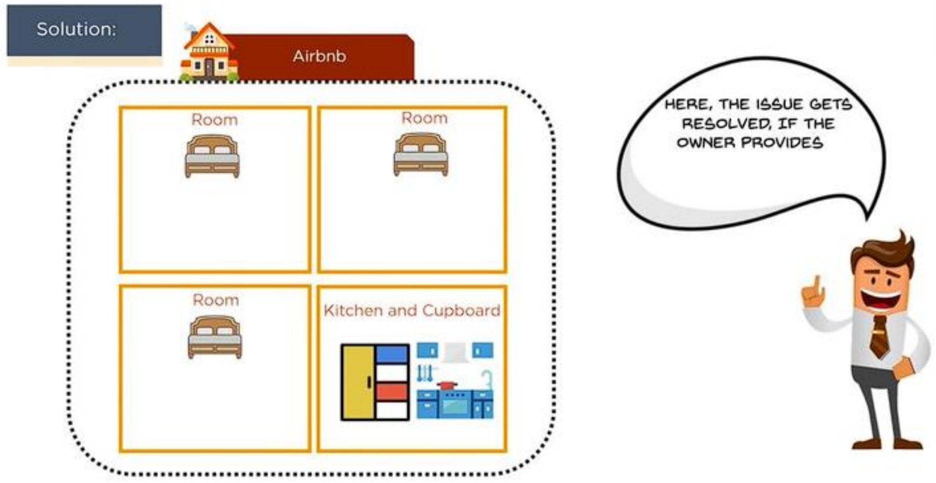
And none of the guests are ready to share the cupboard and the kitchen

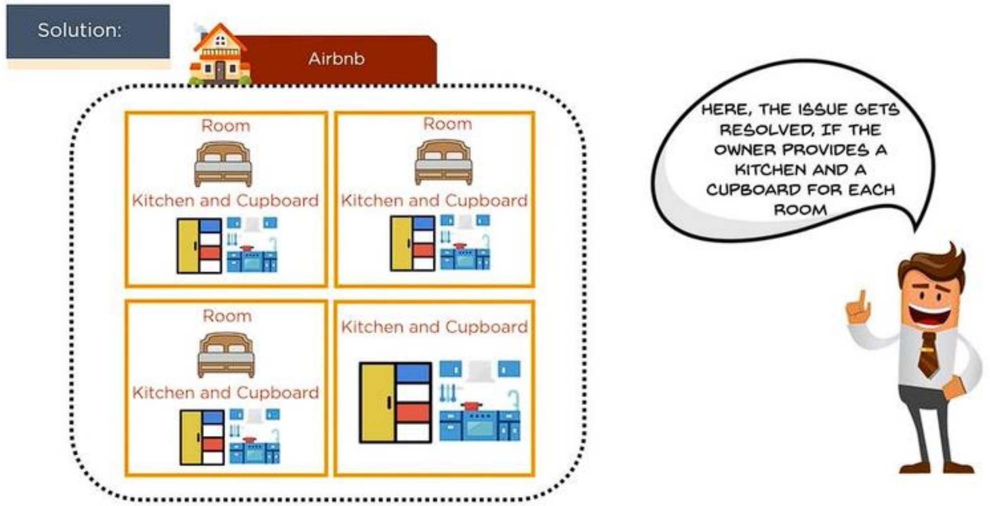


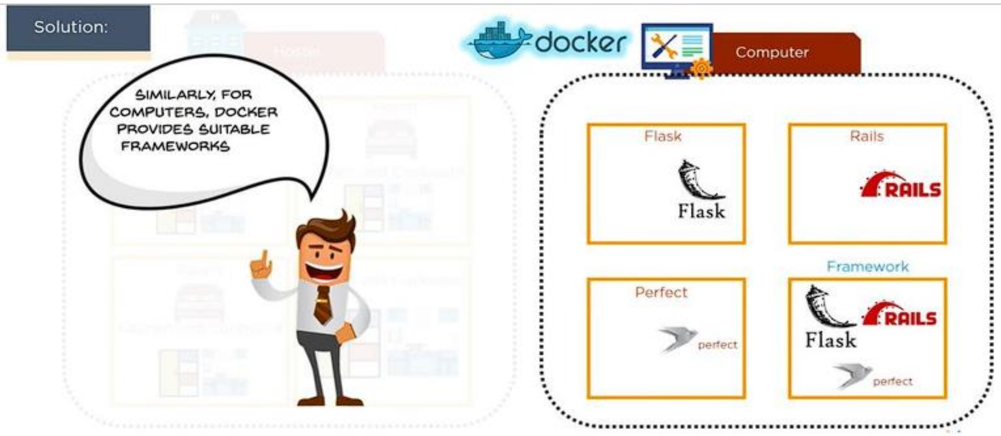


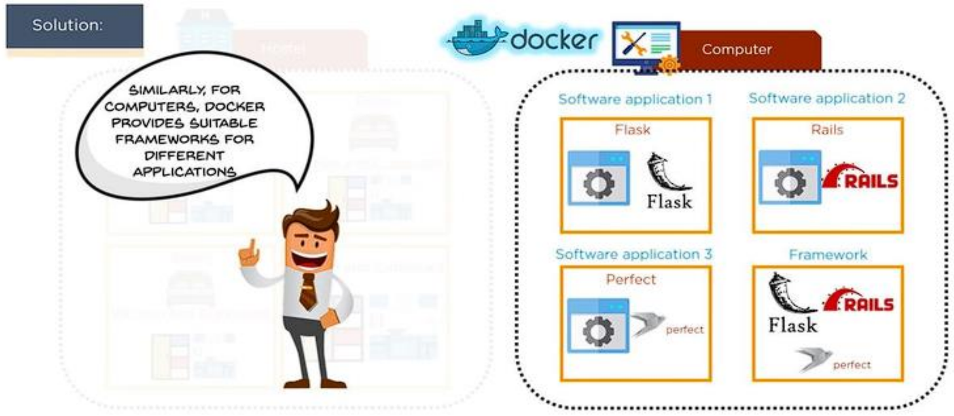

WHAT IF I WANT TO RUN ALL MY APPLICATIONS WITH THEIR SUITABLE FRAMEWORKS?
Software application 1
Flask
Software application 2
Rails
Software application 3
Perfect
Framework
Flask
RAILS
perfect

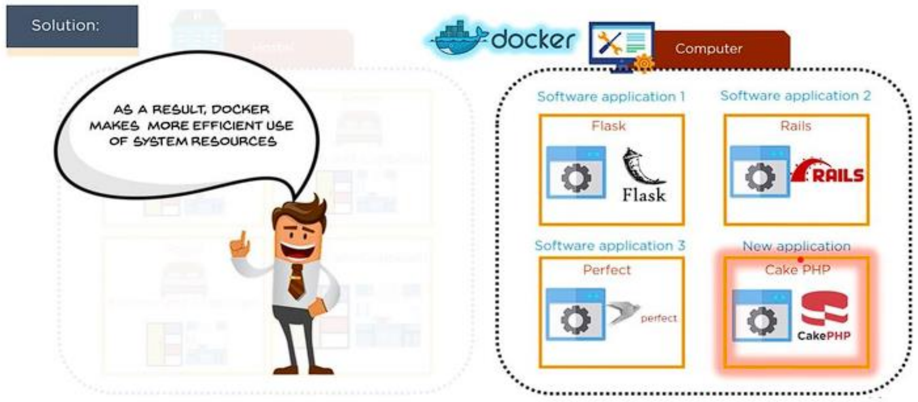

WHAT IF I WANT TO RUN ALL MY APPLICATIONS WITH THEIR SUITABLE FRAMEWORKS?
DOCKER WILL HELP YOU RUN APPLICATIONS WITH THEIR SUITABLE FRAMEWORKS
docker


Solution:
Airbnb
Room
Room
Room
Kitchen and Cupboard
HERE, THE ISSUE GETS RESOLVED, IF THE OWNER PROVIDES


Solution:
Airbnb
Room
Kitchen and Cupboard
Room
Kitchen and Cupboard
Room
Kitchen and Cupboard
Kitchen and Cupboard
HERE, THE ISSUE GETS RESOLVED, IF THE OWNER PROVIDES A KITCHEN AND A CUPBOARD FOR EACH ROOM


Solution:
SIMILARLY, FOR COMPUTERS, DOCKER PROVIDES SUITABLE FRAMEWORKS
docker
Computer
Flask
Rails
Perfect
Framework


Solution:
SIMILARLY, FOR COMPUTERS, DOCKER PROVIDES SUITABLE FRAMEWORKS FOR DIFFERENT APPLICATIONS
docker
Computer
Software application 1
Flask
Software application 2
Rails
Software application 3
Perfect
Framework


Solution:
AS A RESULT, DOCKER MAKES MORE EFFICIENT USE OF SYSTEM RESOURCES
docker
Computer
Software application 1
Flask
Software application 2
Rails
Software application 3
Perfect
New application
Cake PHP

Docker Engine

- Docker Engine is an open source containerization technology for building and containerizing your applications. Docker Engine acts as a client-server application with:
  1) A server with a long-running daemon process dockerd.
  2) APIs which specify interfaces that programs can use to talk to and instruct the Docker daemon.
  3) A command line interface (CLI) client docker.
- The CLI uses Docker APIs to control or interact with the Docker daemon through scripting or direct CLI commands. Many other Docker applications use the underlying API and CLI. The daemon creates and manage Docker objects, such as images, containers, networks, and volumes.

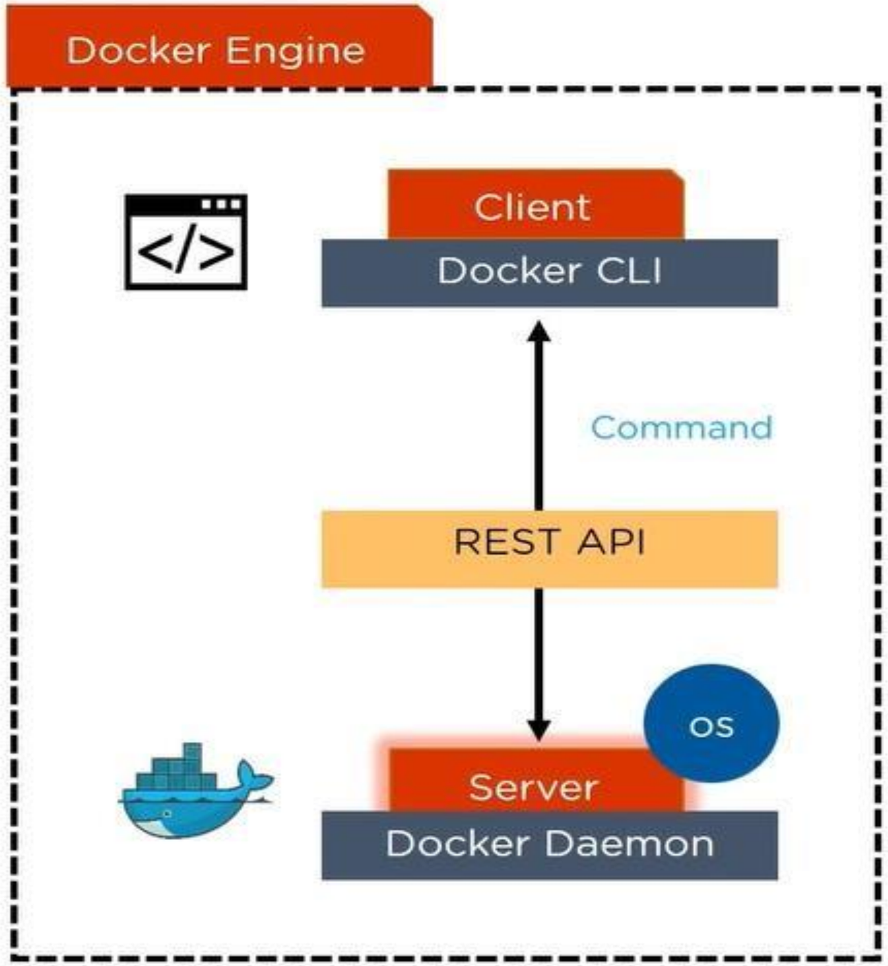
Docker Engine

- Docker Engine is the underlying client-server technology that supports the tasks and workflows involved in building, shipping and running containerized applications using Docker's components and services.
- Used alone, the term Docker can refer either to Docker Engine or to the company Docker Inc.

➢ **Components of Docker Engine**

- Docker Engine is an open source technology comprising a server with a daemon process called dockerd, a REST API and a client-side command-line interface (CLI) called docker.
- The engine creates a server-side daemon process that hosts images, containers, networks and storage volumes. The CLI lets users interact with the Docker daemon via the API.

How does Docker work?

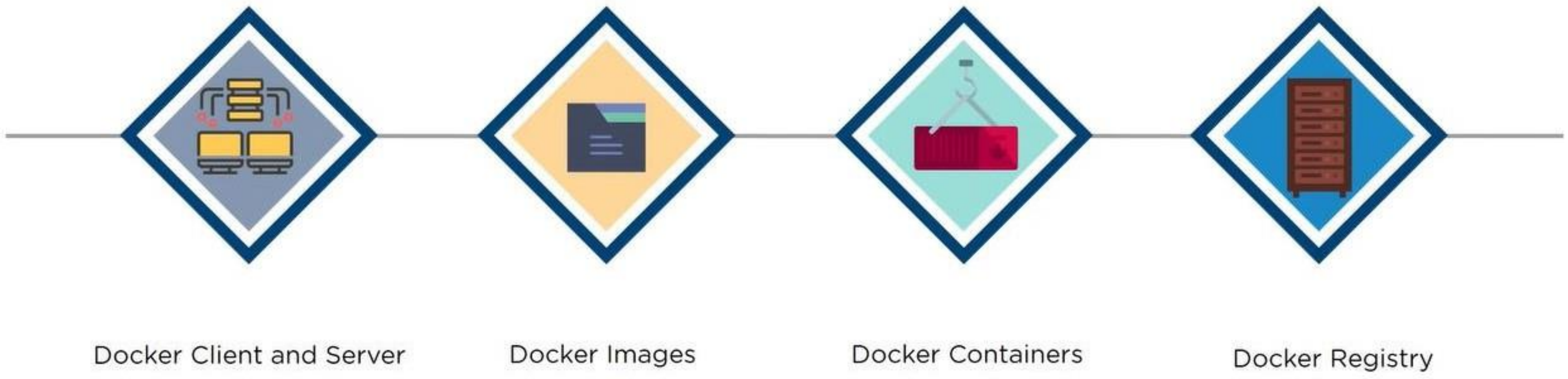


- Docker Engine or Docker is the base engine installed on your host machine to build and run containers using Docker components and services
- It uses a client-server architecture
- Docker Client and Server communicate using Rest API

  What happens here?

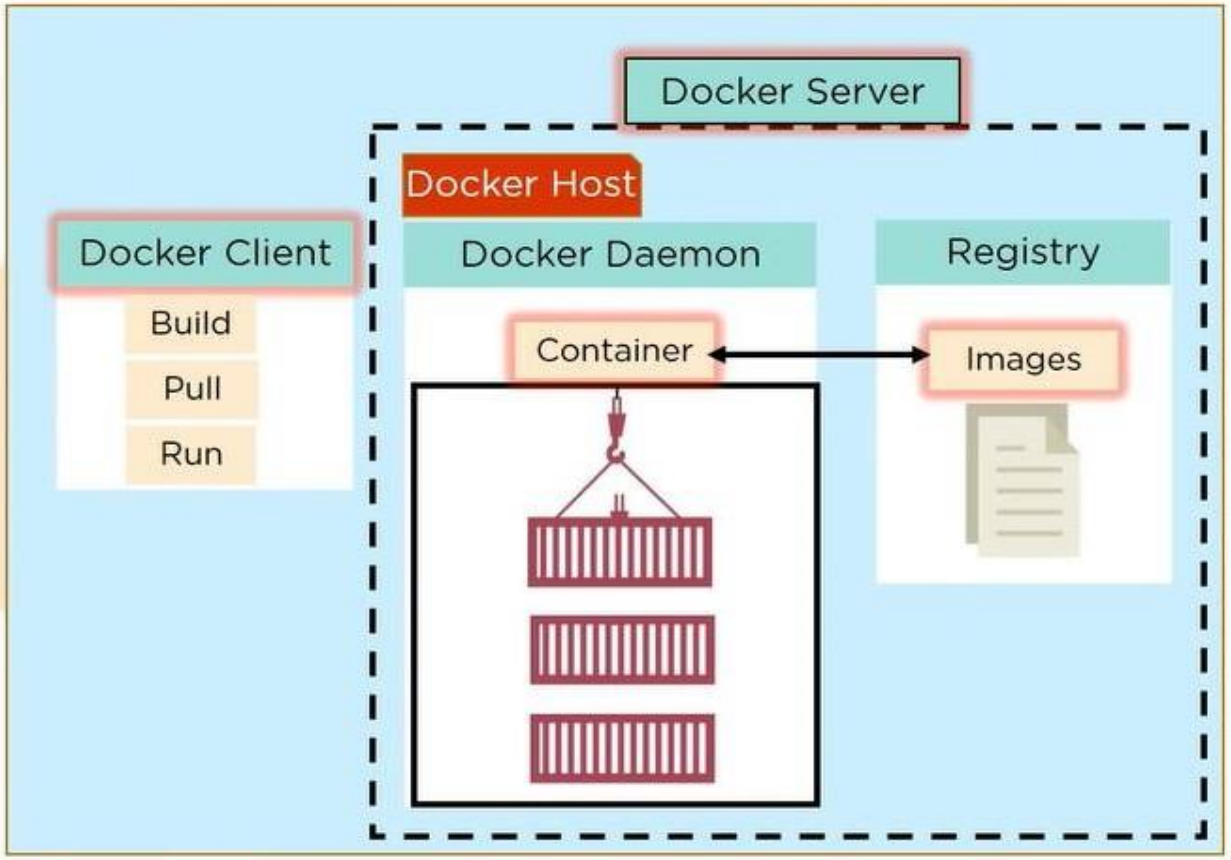
- Docker Client is a service which runs a command. The command is translated using REST API and is sent to the Docker Daemon (server)
- Then, Docker Daemon checks the client request and interacts with the operating system in order to create or manage containers

Components of Docker



Docker Client and Server

- Docker Client is accessed from the terminal and a Docker Host runs the Docker Daemon and registry
- A user can build Docker Images and run Docker Containers by passing commands from the Docker Client to Docker server

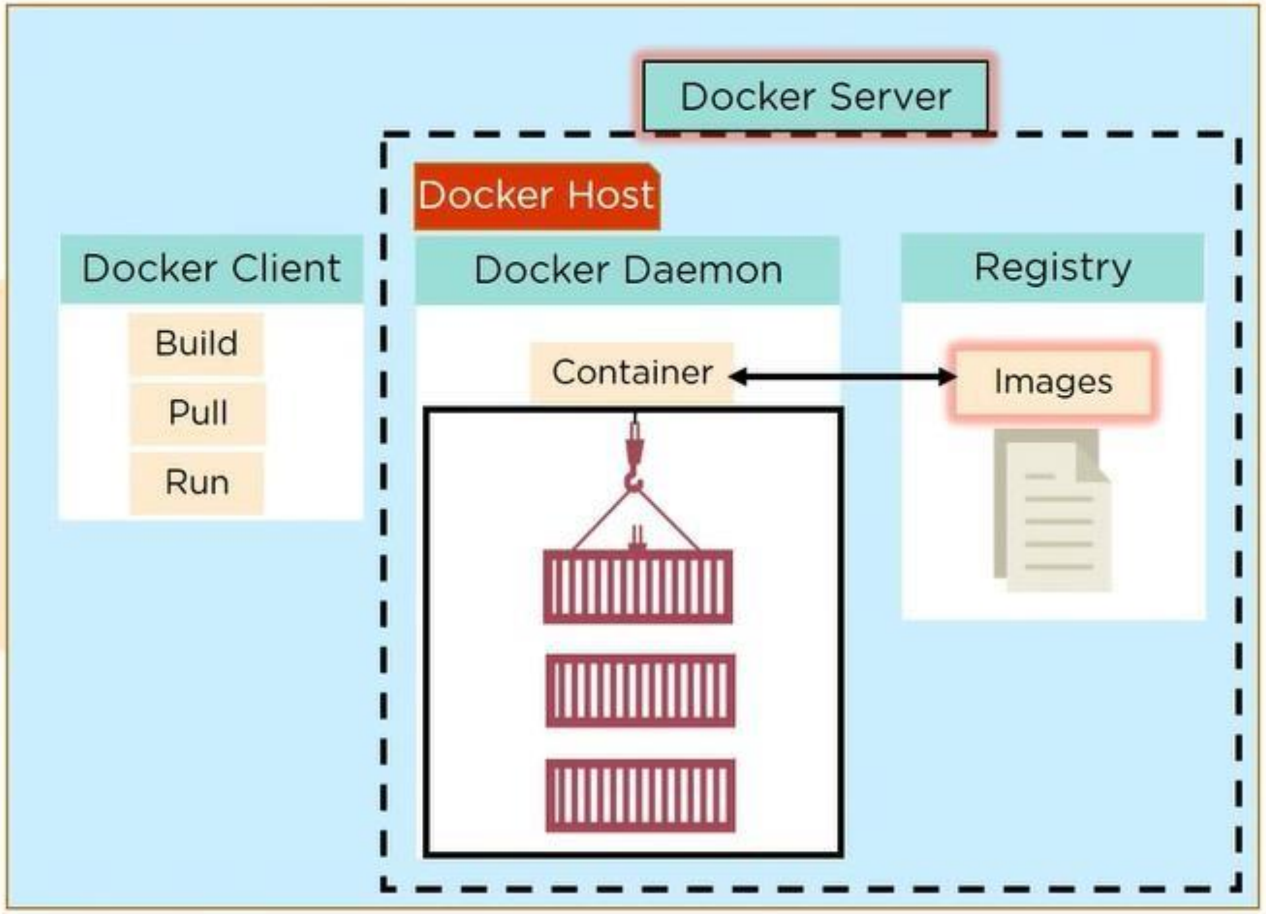


Parul® University

DIGITAL LEARNING CONTENT

Docker Image

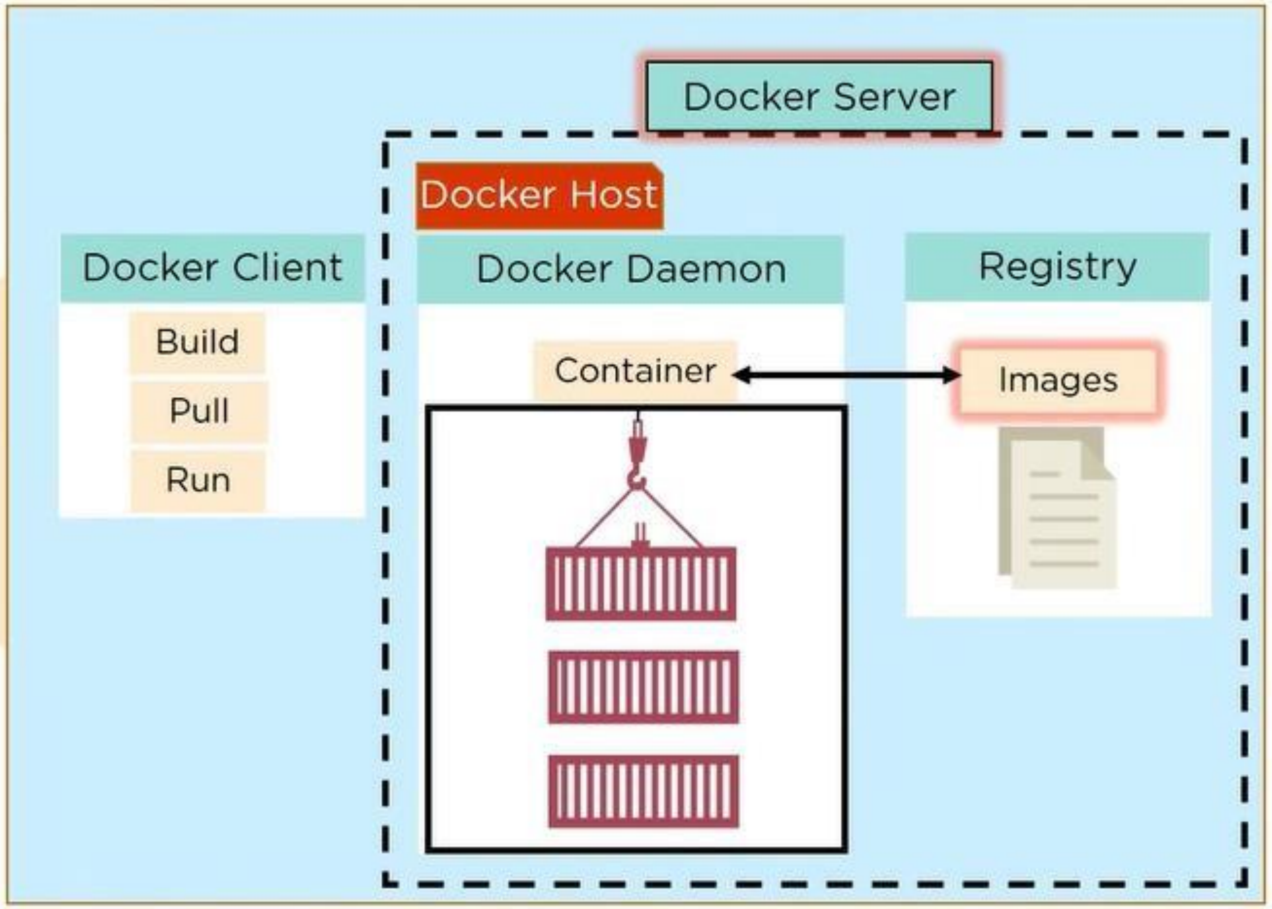
- Docker Image is a template with instructions, which is used for creating Docker Containers
- A Docker Image is built using a file called Docker File
- Docker Image is stored in a Docker Hub or in a repository (like registry.hub.Docker.com.)



Docker Image

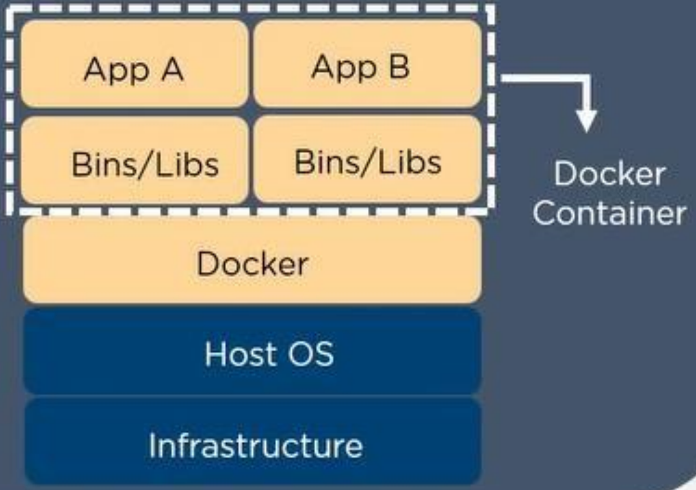
Syntax to create a Docker Container using Docker Image:

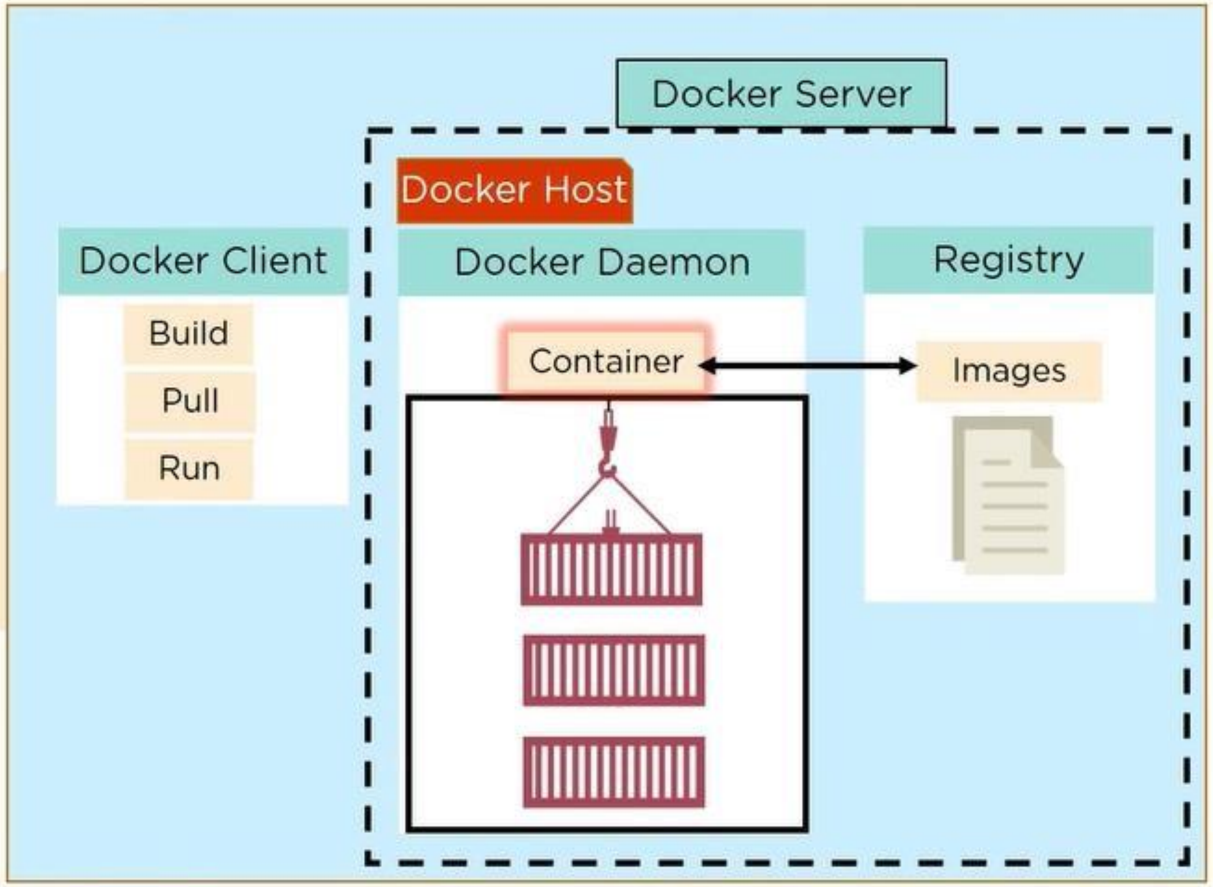
```
Docker container create [OPTIONS] IMAGE [COMMAND] [ARG...]
```



Docker Container

- Docker Container is a standalone, executable software package which includes applications and their dependencies



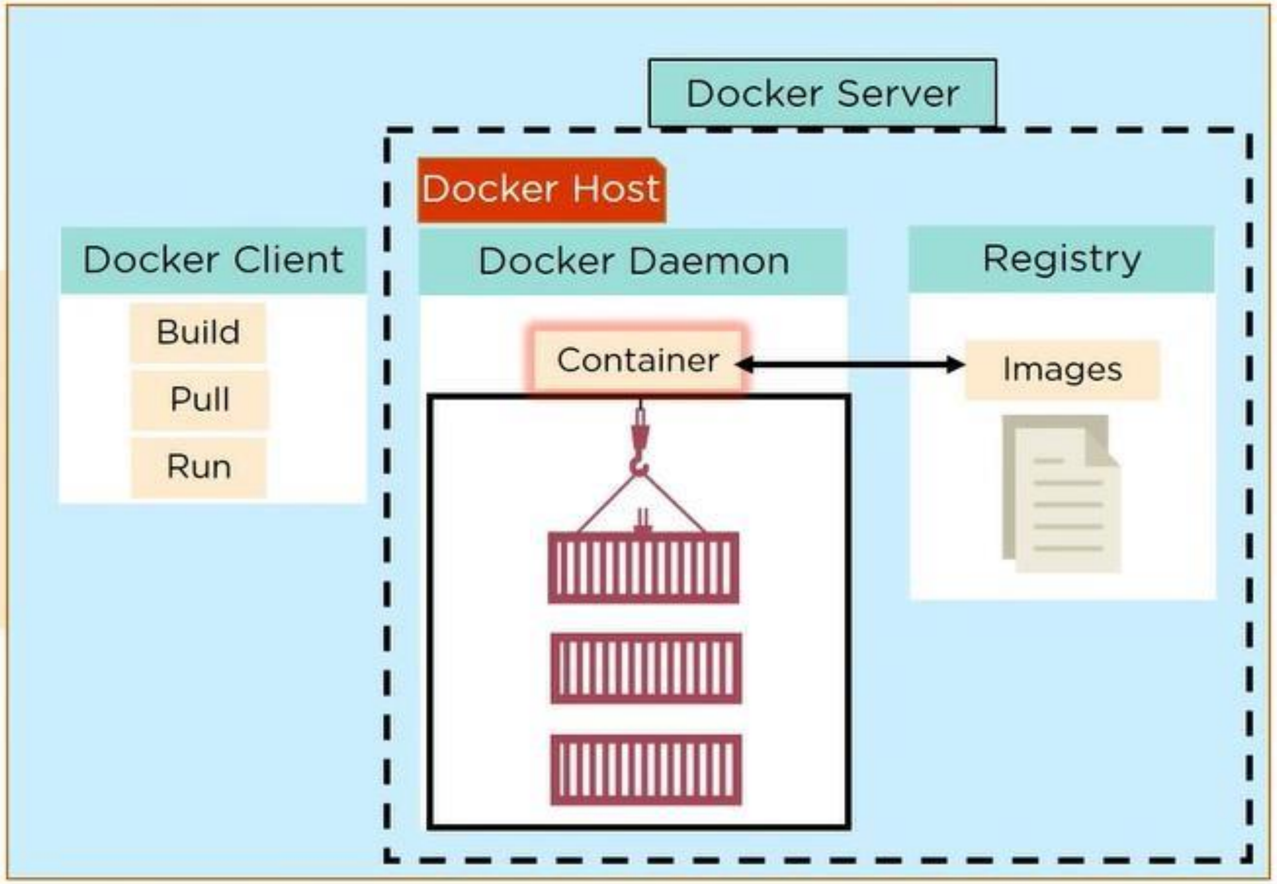


Parul® University

DIGITAL LEARNING CONTENT

Docker Container

- Docker Container is a standalone, executable software package which includes applications and their dependencies
- Numerous Docker Containers run on the same infrastructure and share operating system (OS) with its other containers
- Here, each application runs in isolation

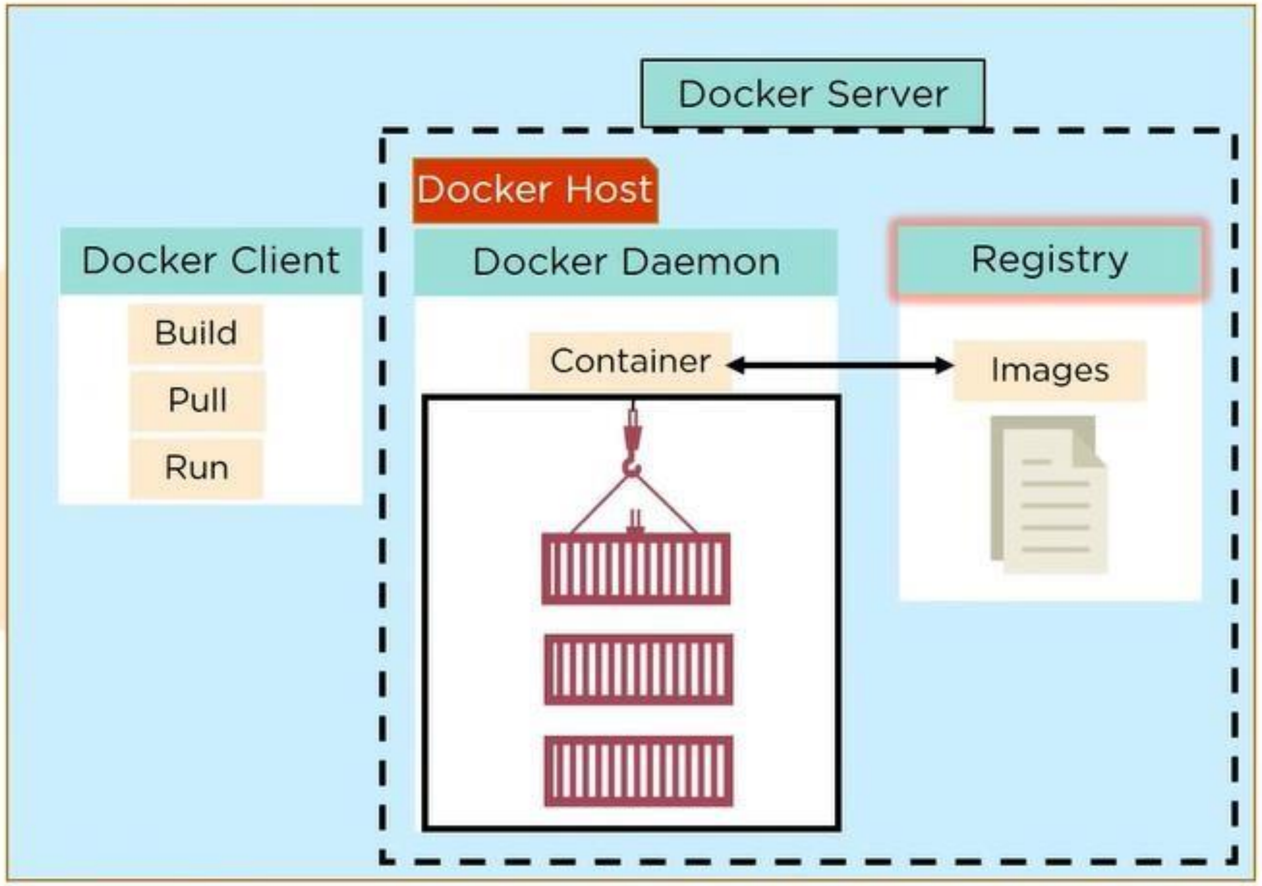


Parul® University

DIGITAL LEARNING CONTENT

Docker Registry

- Docker Registry is an open source server-side service used for hosting and distributing images
- Docker also has its own default registry called Docker Hub
- Here, images can be stored in either public or private repositories
- Pull and Push are the commands used by users in order to interact with a Docker Registry



Parul® University

DIGITAL LEARNING CONTENT

Docker Registry

- Docker Registry is an open source server-side service used for hosting and distributing images
- Docker also has its own default registry called Docker Hub
- Here, images can be stored in either public or private repositories
- Pull and Push are the commands used by users in order to interact with a Docker Registry

In order to build a container, pull command is used to get a Docker Image from the Docker repository

With push command, a user can store the Docker Image in Docker Registry

Docker pull <image>:<tag>: pulls an image from DTR

Docker push <image>:<tag>: pushes an image to DTR

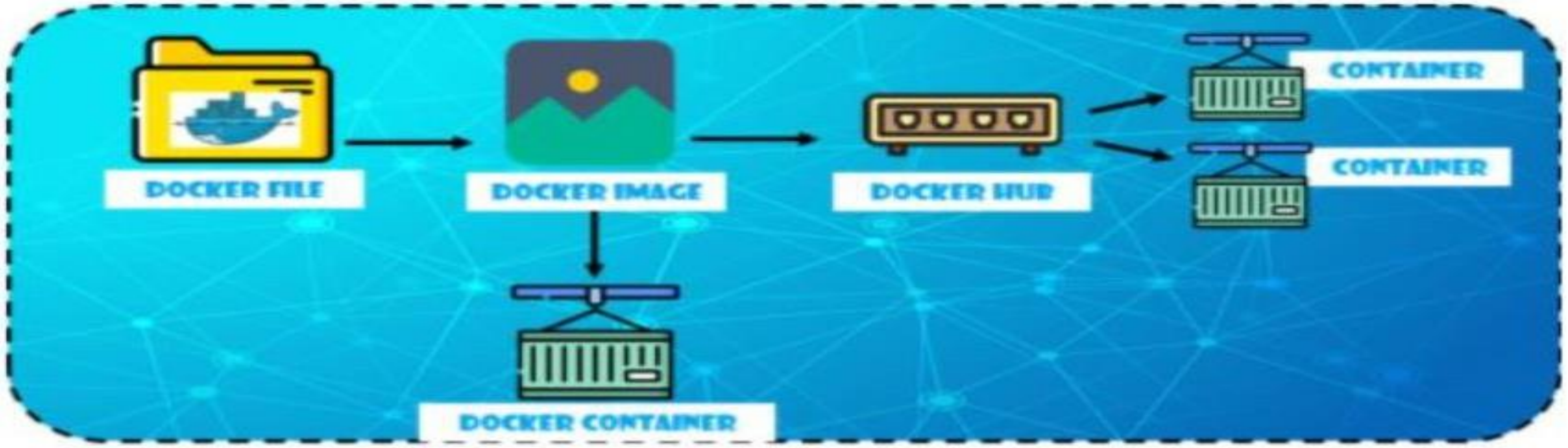
Docker Network

- Docker networking enables a user to link a Docker container to as many networks as he/she requires. Docker Networks are used to provide complete isolation for Docker containers. A user can add containers to more than one network.
- **How Does Docker Networking Work?**
- For a more in-depth understanding, let’s have a look at how Docker Networking works. Below is a diagrammatic representation of the Docker Networking workflow:

Parul® University

DIGITAL LEARNING CONTENT

Docker Network



- Docker File builds the Docker Image.
- Docker Image is a template with instructions, which is used to build Docker Containers.
- Docker has its own cloud-based registry called Docker Hub, where users store and distribute container images.
- Docker Container is an executable package of an application and its dependencies together.

Docker Network

➢ **Functionalities of the different components:**

- Docker File has the responsibility of building a Docker Image using the build command.
- Docker Image contains all the project’s code.
- Using Docker Image, any user can run the code to create Docker Containers.
- Once Docker Image is built, it’s either uploaded in a registry or a Docker Hub

Container Security Challenges

➢ **Container Malware**

- Malware is malicious code that is deployed within a container. It can sneak into containers at multiple stages of the container lifecycle.
- An attacker who compromises your CI/CD environment could insert malware into the source code repositories that are later used to build container images, for instance. Or,
- Attackers could breach container registry and replace images with tainted ones that contain malware.
- A third type of container malware attack involves tricking users into downloading malicious container images from external sources.

Container Security Challenges

➢ **Insecure Container Privileges**

- When containers are allowed to run with more privileges than they strictly require, security risks result. Insecure privileges are usually caused by problematic configurations with the container orchestrator.
- For example, containers orchestrated by Kubernetes may have more privileges than they should if Kubernetes security con
- Containers with Sensitive Data
- Containers are not intended to be used to store data. But sometimes, organizations make the mistake of storing sensitive information inside container images.

Container Security Challenges

- For example, Vine’s entire source code was exposed when someone discovered a container registry that Vine thought was private, but which was publicly accessible, and which turned out to be hosting images that contained the source code.

Container Security Challenges Solution

➢ **Container Images**

- A container image is a file that contains the code required to run a container. The image is not the container itself, but rather a blueprint on which a running container will be based. Thus, if the contents of a container image include malware or sensitive data, the containers that are created from the image will be insecure.
- you should scan your internal source code to help ensure that malware doesn't make its way into your container images.

Container Security Solution

➢ **Container Registries**

- After container images are created, they are usually stored in a container registry, from
  which users can download them.
- There are a few best practices to follow to address registry security. First, you should enforce access controls to ensure that only authorized users can access the images in your registry. Doing so helps prevent accidental data leaks that may occur if images contain private applications or data.

Container Security Solution

➢ **Container Runtime Environment**

- The final stage of the container lifecycle is runtime. This is when containers are deployed into a live environment using container images that were downloaded from a registry.
- Runtime security is one of the most complex aspects of container security because it involves multiple moving pieces, which can vary depending on which type of container application stack you use. In most cases, however, runtime security is based on securing:

Container Security Solution

- **The container runtime:** This is the process on the server that actually executes containers. You should ensure that your runtime software is up-to-date and patched against known security vulnerabilities.
- **The orchestrator:** The container orchestrator deploys and manages containers. Most orchestrators offer a variety of tools to help restrict containers' privileges and minimize security risks, but you should also use third-party monitoring and analysis tools to help detect security issues at the orchestrator level.
- **Nodes:** Kubernetes nodes are the servers that host containers. You need to secure the node operating system, user accounts, networking configurations, and other resources in order to ensure that a breach at the node level doesn't allow attackers to impact your container environment.

What is a web service?

- A Web service include any software, application, or cloud technology that provides standardized web protocols (HTTP or HTTPS) to communicate, and exchange data messaging – usually XML (Extensible Markup Language) over the internet.

- A web service comprises these essential functions:

➢ Available over the internet or intranet networks
➢ Standardized XML messaging system
➢ Independent of a single operating system or programming language
➢ Self-describing via standard XML language

Cont….

- A web service supports communication among numerous apps with HTML, XML, WSDL, SOAP, and other open standards. XML tags the data, SOAP transfers the message, and WSDL describes the service’s accessibility.

- A web service uses for example two sets of java, .NET, or PHP apps providing a way for these applications to communicate over a network.

- On one side, for example, a java app interacts with the java, .net, and PHP apps on the other end by way of the web service communicating an independent language.

Technologies and the processes required when deploying web services

1. **XML-RMC**(Remote Procedure call): It is the most basic XML protocol to exchange data between a wide variety of devices on a network. It uses HTTP to quickly and easily transfer data and communication other information from client to server.
2. **UDDI** (Universal Description, Discovery, and Integration): It is an XML-based standard for detailing, publishing, and discovering web services. It's basically an internet registry for businesses around the world.
3. **SOAP**: It is is an XML-based Web service protocol to exchange data and documents over HTTP or SMTP (Simple Mail Transfer Protocol). It is an XML-based messaging protocol for exchanging information among computers. SOAP is an application of the XML specification.

Types of Web Services

- A SOAP message is an ordinary XML document containing the following elements −

•**Envelope** − Defines the start and the end of the message. It is a mandatory element.

•**Header** − Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end-point. It is an optional element.

•**Body** − Contains the XML data comprising the message being sent. It is a mandatory element.

•**Fault** − An optional Fault element that provides information about errors that occur while processing the message.

4. **Web Services Description Language** (WSDL): It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

➢ **Features of WSDL**:

•WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.

•WSDL definitions describe how to access a web service and what operations it will perform.

•WSDL is a language for describing how to interface with XML-based services.

Advantages of web services

1.Scalability: Web services can be scaled easily to handle increased demand by adding more resources or servers.

2.Reusability: Web services lets you package specific functions as services that can be used in different applications. This saves time, avoids repeating code, and makes it easier to manage and update the code.

3.Standardization: Web services uses standardized protocols like HTTP, SOAP, which ensures consistent communication and compatibility across various platforms and technologies.

4. Cost Efficiency: Developing and maintaining web services can lead to cost savings due to the reuse of existing functionalities, reduced development time, and improved resource allocation.

5. Global Reach: Web services can be accessed by clients from anywhere with an internet connection, enabling businesses and applications to have a global reach.

Disadvantages of web services

1. Latency: Due to the nature of internet communication, web services can introduce latency or delays in data transmission, impacting real-time applications or those requiring rapid responses.

2. Complexity: Implementing and managing web services can be complex, especially when dealing with different protocols, data formats, and security measures. This complexity can increase development time and maintenance efforts.

3. Dependency on Network: Web services heavily rely on network connectivity. If the network experiences issues or downtime, it can disrupt communication between applications and services.

4. Privacy Concerns: Transmitting sensitive data over the internet using web services raises privacy concerns, especially if proper encryption and security measures are not implemented

5. Lack of Control: When relying on third-party web services, you might have limited control over updates, changes, or downtimes, which could affect your application's performance and availability.

Parul® University

References

1. https://www.konverge.co.in/virtualization-in-cloud-computing-need-types-and-importance/

2. https://www.tutorialspoint.com/soap/what_is_soap.htm

3. https://www.yarddiant.com/blog/web-development/what-are-the-pros-and-cons-of-web-services.html#:~:text=While%20web%20services%20offer%20numerous,calls%20within%20a%20single%20application.

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in