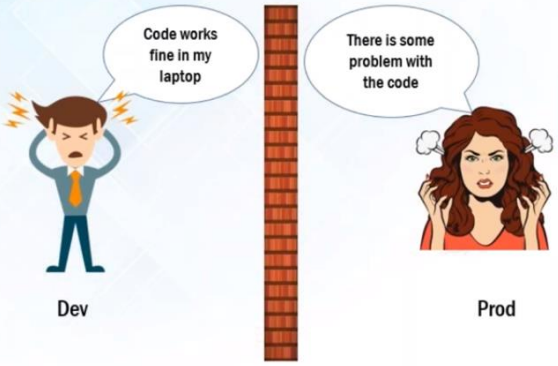
Docker Uses

Problems Before Docker

edureka!

An application works in developer's laptop but not in testing or production. This is due to difference in computing environment between Dev, Test and Prod.

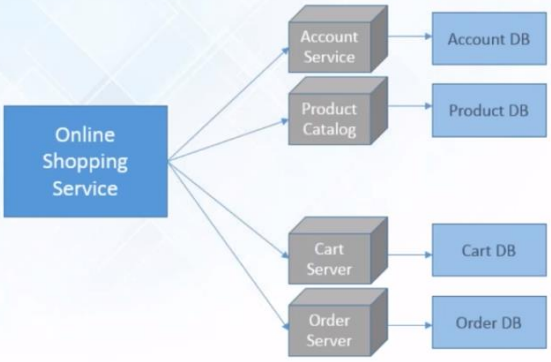


In Dev there can be a software that is upgraded and in Prod the old version of software might be present

Problems Before Docker

edureka!

The idea behind microservices is that some types of applications become easier to build and maintain when they are broken down into smaller, composable pieces which work together. Each component is developed separately, and the application is then simply the sum of its constituent components.

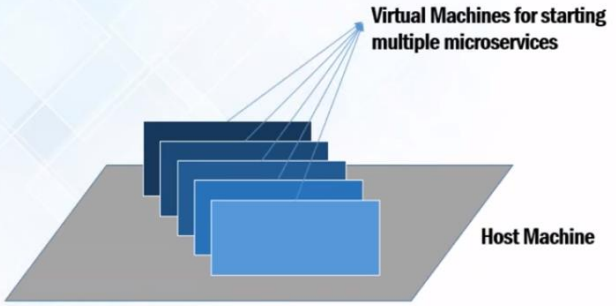


For example imagine an online shop with separate microservices for user-accounts, product-catalog order-processing and shopping carts

Problems Before Docker

edureka!

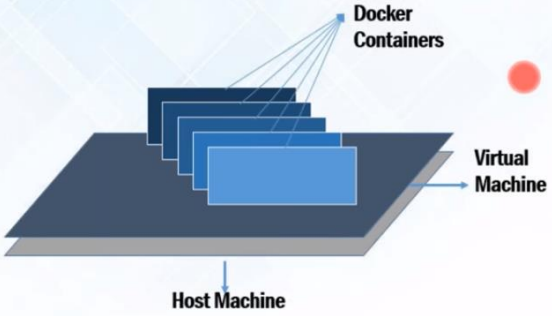
Developing an application requires starting several of microservices in one machine. So if you are starting five of those services you require five VMs on that machine.



How Docker Solves These Problems

edureka!

You can run several microservices in the same VM by running various Docker containers for each microservice.

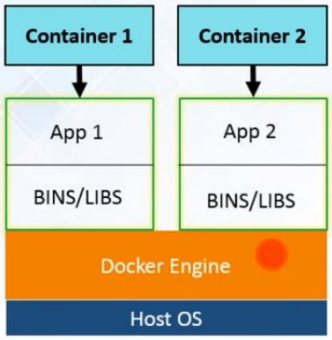


Provides a consistent computing environment throughout the whole SDLC.

What Is Docker?

edureka!

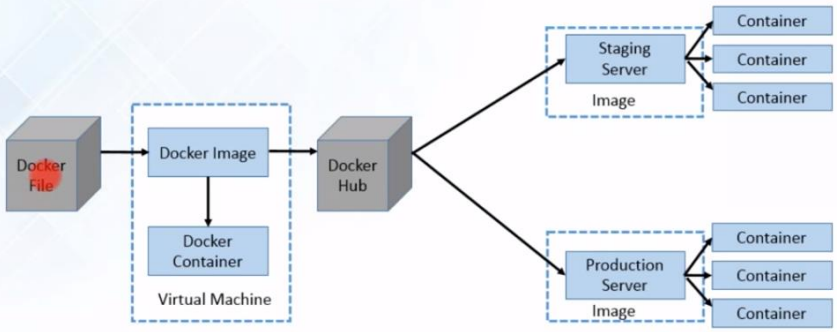
- Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.
- Docker containers are lightweight alternatives to Virtual Machines and it uses the host OS.
- You don't have to pre-allocate any RAM in containers.



Docker In A Nutshell

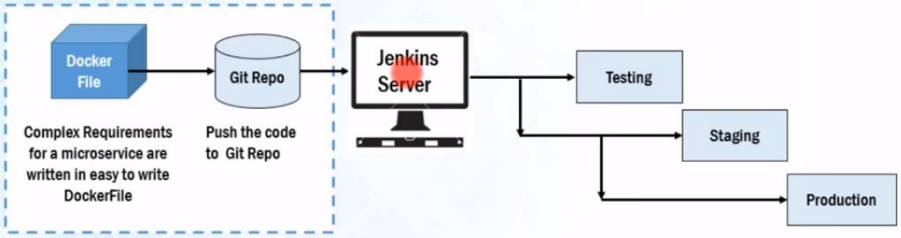
edureka!

- Docker file builds a Docker image and that image contains all the project's code
- You can run that image to create as many Docker containers as you want
- Then this Image can be uploaded on Docker hub, from Docker hub any one can pull the image and build a container



Docker Example

edureka!



- Create complex requirements for a microservice within an easy-to-write Dockerfile.
- Push the code up to the Git Repo.

- CI server pull it down and build the exact environment that will be used in production to run the test suite without needing to configure the CI server at all.
- Deploy it out to a staging environment for testers.
- Roll exactly what you had in development, testing, and staging into production

Docker Registry

edureka!

- Docker Registry is a storage component for Docker Images
- We can store the Images in either Public / Private repositories
- Docker Hub is Docker's very own cloud repository

Why Use Docker Registries?

- Control where your images are being stored
- Integrate image storage with your in-house development workflow

Docker Images & Containers

edureka!

run

Docker Images

Docker Containers

- Read Only Template Used To Create Containers
- Built By Docker Users
- Stored In Docker Hub Or Your Local Registry

- Isolated Application Platform
- Contains Everything Needed To Run The Application
- Built From One Or More Images

Docker Compose

edureka!

Docker Compose makes it easier to configure and run applications made up of multiple containers. For example: imagine being able to define three containers—one running a web app, another running postgres, and a third running redis—all in one YAML file and then running those three connected containers with a single command.

Containers

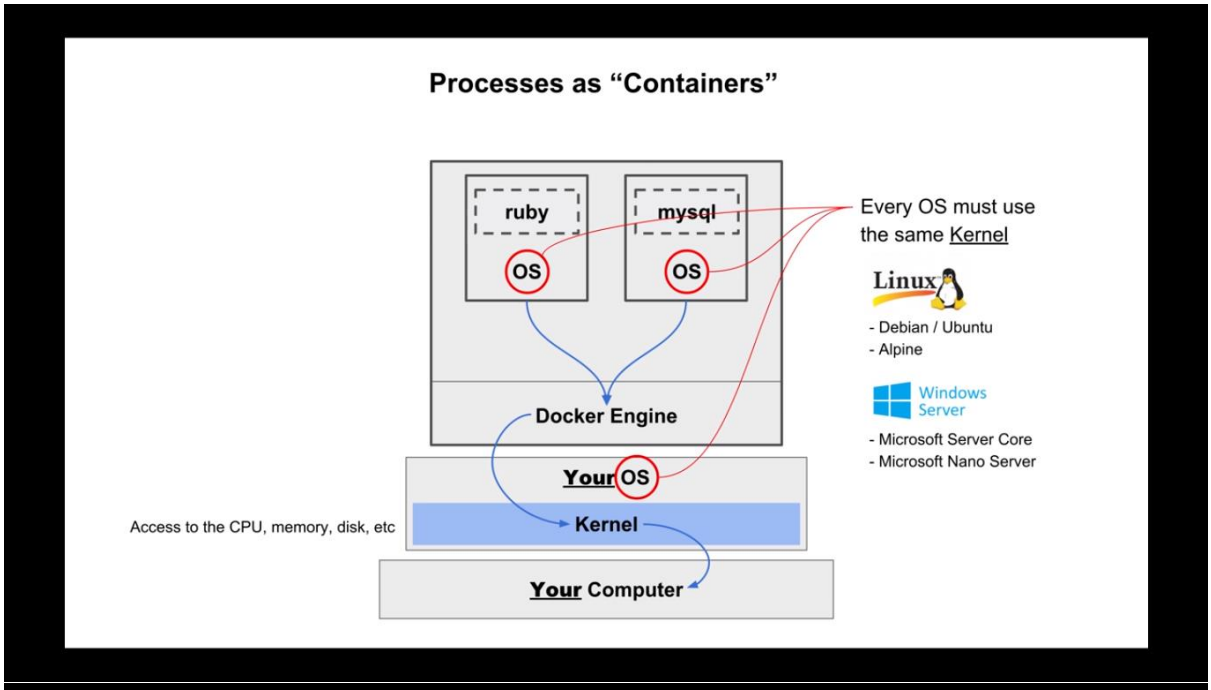
web app

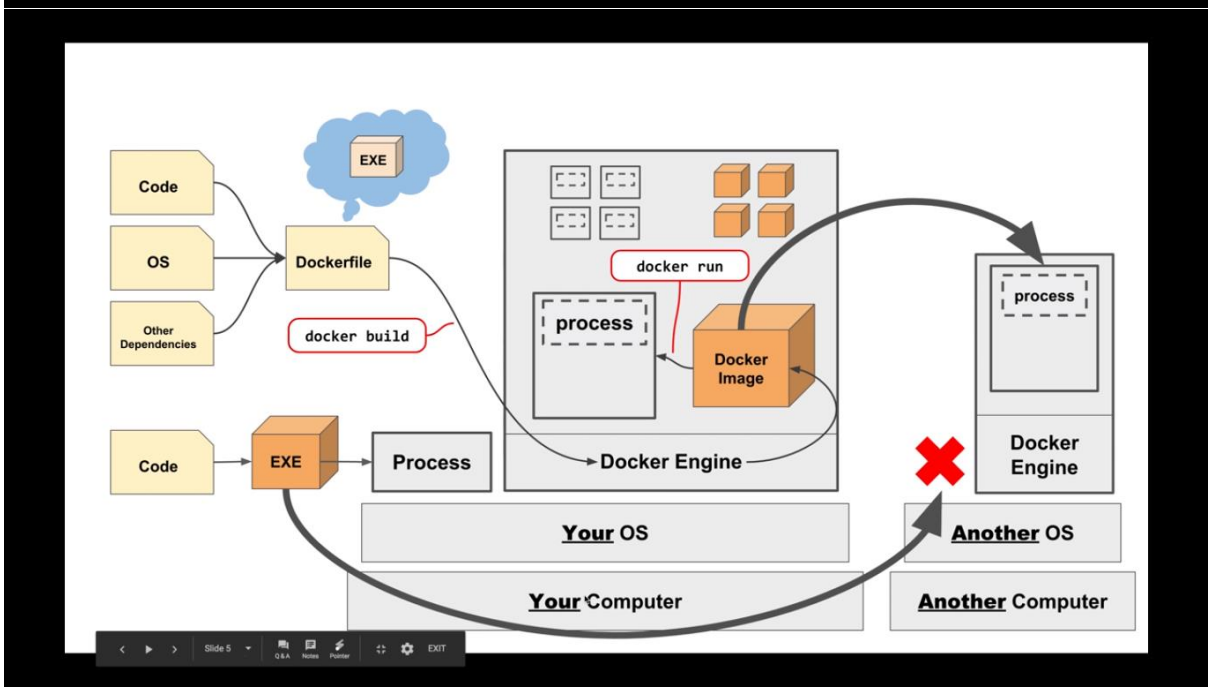
postgres

redis

Docker Compose File

You can run these three containers with a single command



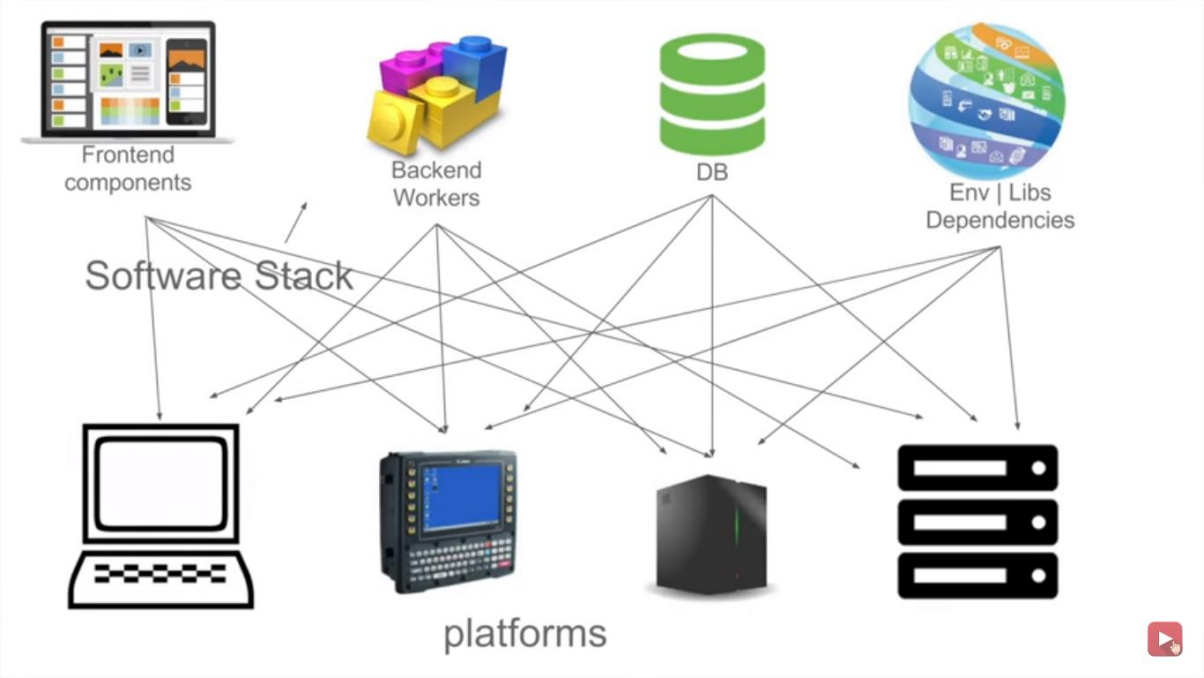


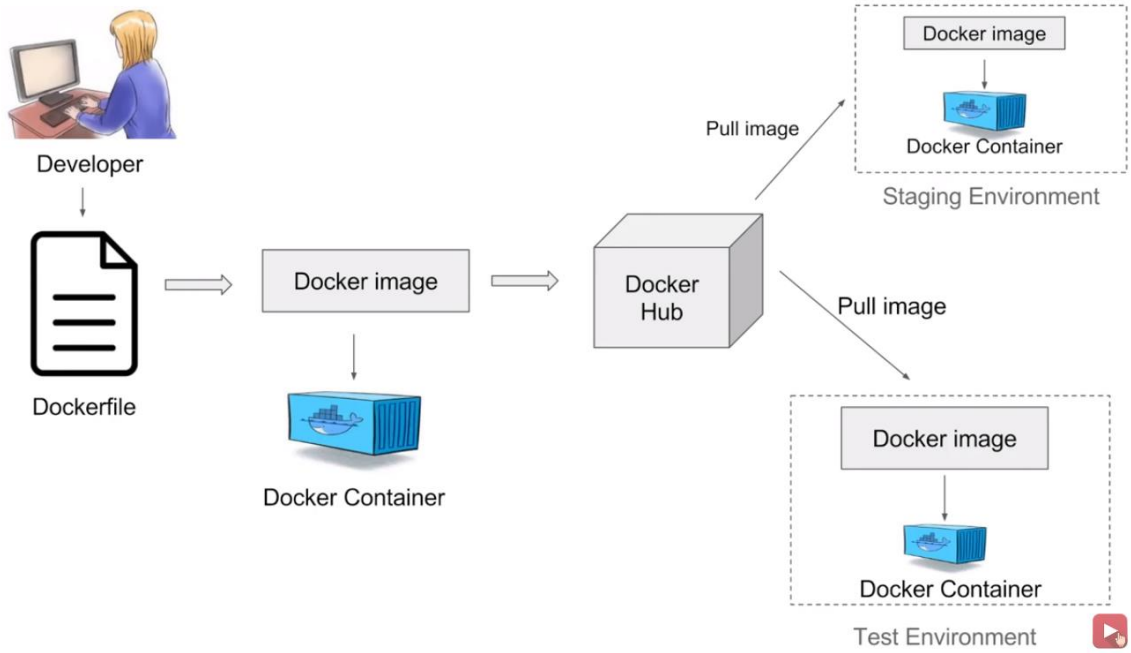
What is Docker ?

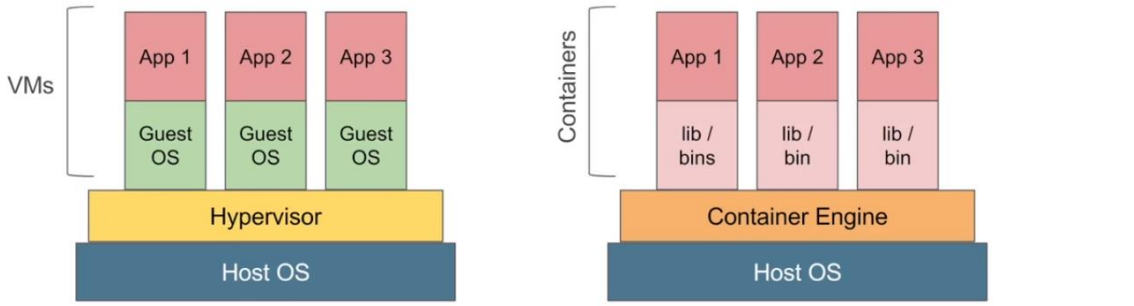
- Docker is the world’s leading software container platform.
- Docker makes the process of application deployment very easy and efficient and resolves a lot of issues related to deploying applications.
- Docker is a tool designed to make it easier to deploy and run applications by using containers.
- Docker gives you a standard way of packaging your application with all its dependencies in a container.
- Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

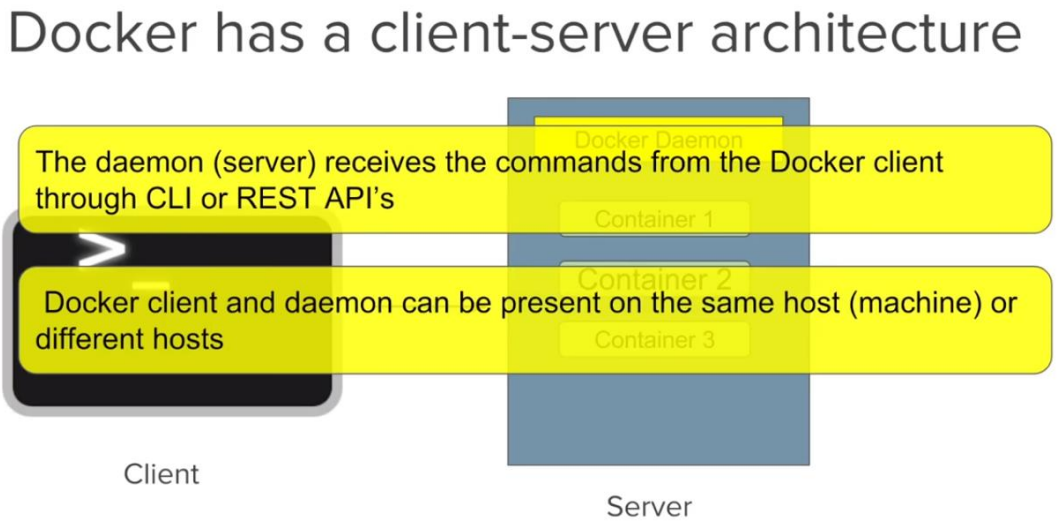
- Understand Docker with analogy of the Shipping industry

  How a real world problem was resolved using containers

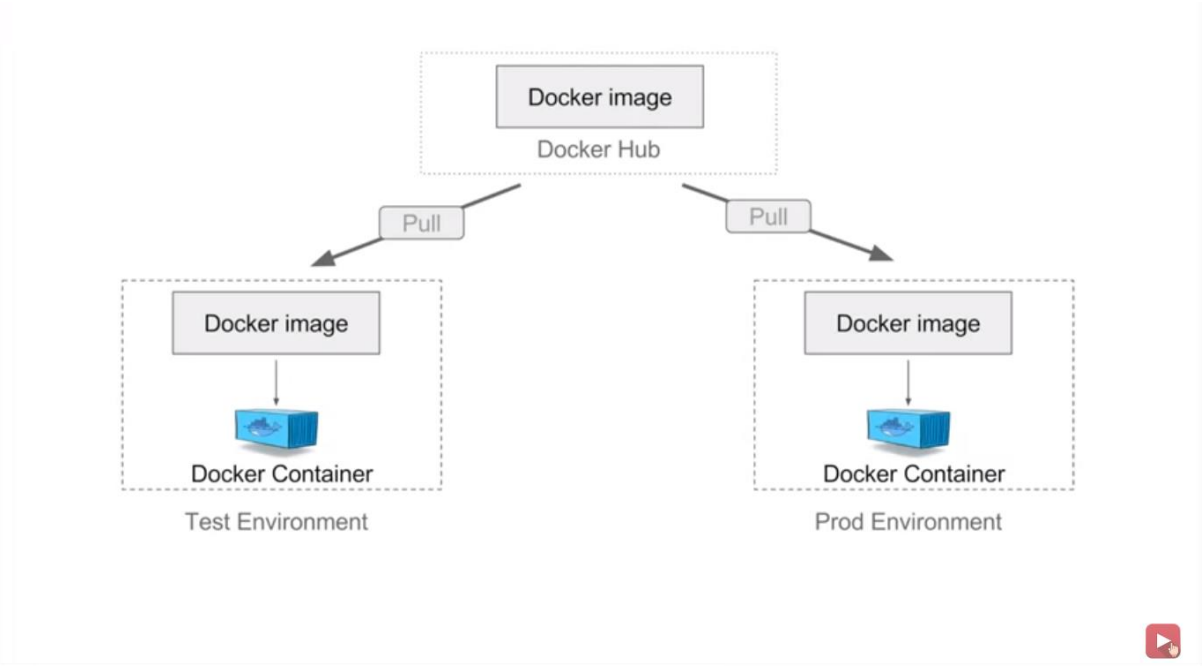


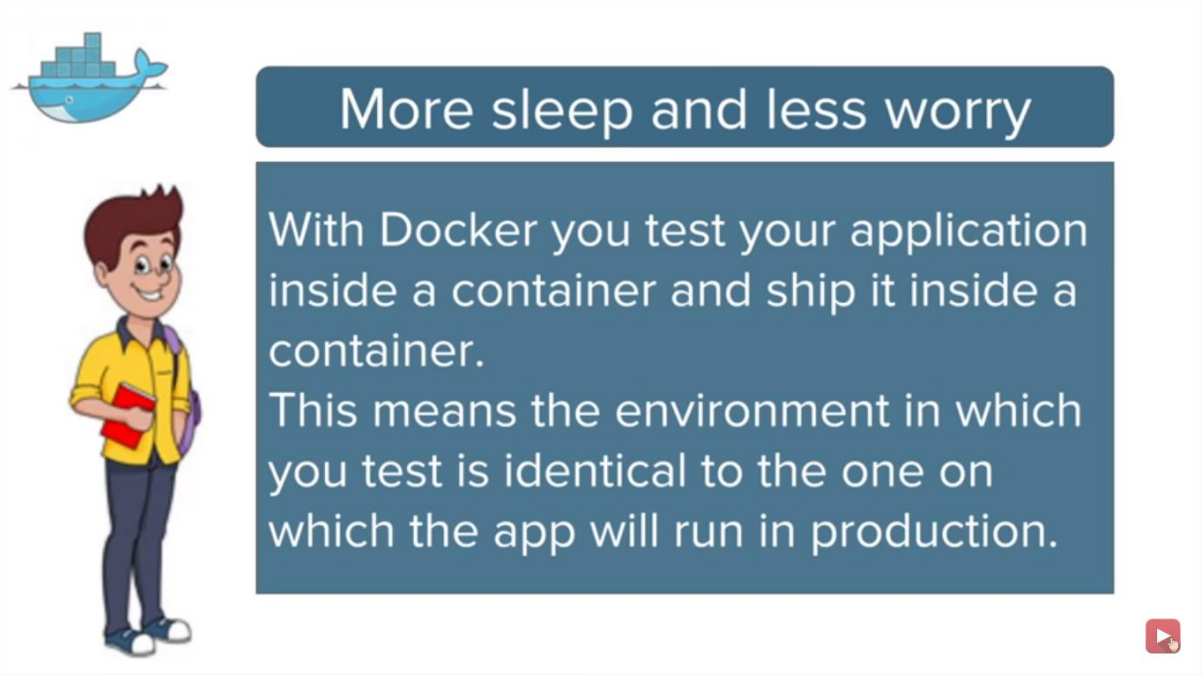






Benefits of using Docker Build app only once No worries that the application will not perform the same way it did on testing env Portability Version Control Isolation Productivity Docker simplifies DevOps.


Docker image
Docker Hub
Pull
Pull
Docker image
Docker Container
Test Environment
Docker image
Docker Container
Prod Environment

- 
More sleep and less worry
With Docker you test your application inside a container and ship it inside a container.
This means the environment in which you test is identical to the one on which the app will run in production.

Portability

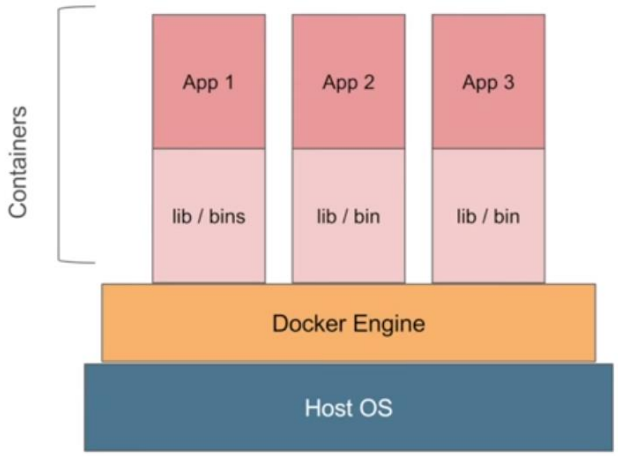
Docker containers can run on any platform.
It can run on your local system, Amazon ec2, Google Cloud platform, Rackspace server, VirtualBox..etc.

A container running on AWS can easily be ported to VirtualBox

Version Control

Like Git, Docker has in-built version control system

Docker containers work just like GIT repositories, allowing you to commit changes to your Docker images and version control them



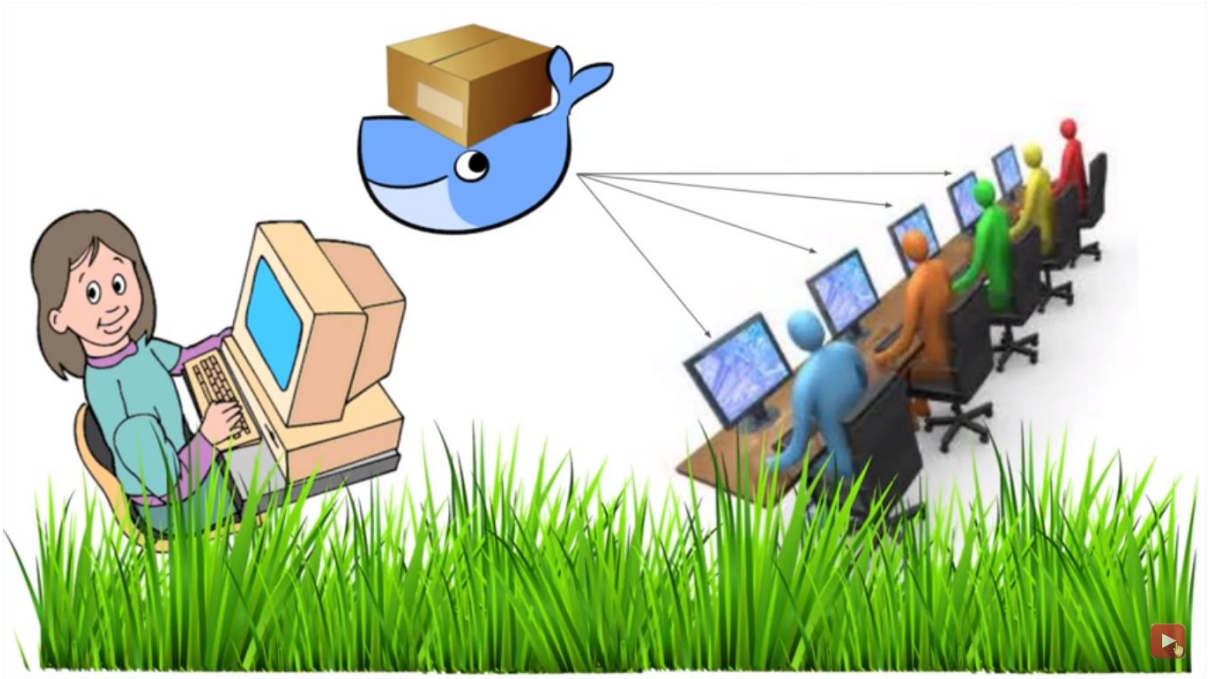
Every app works in its own container and does not interferes with other apps

Isolation

With Docker every application works in isolation in its own container and does not interferes with other applications running on the same system.

So multiple containers can run on same system without interference.

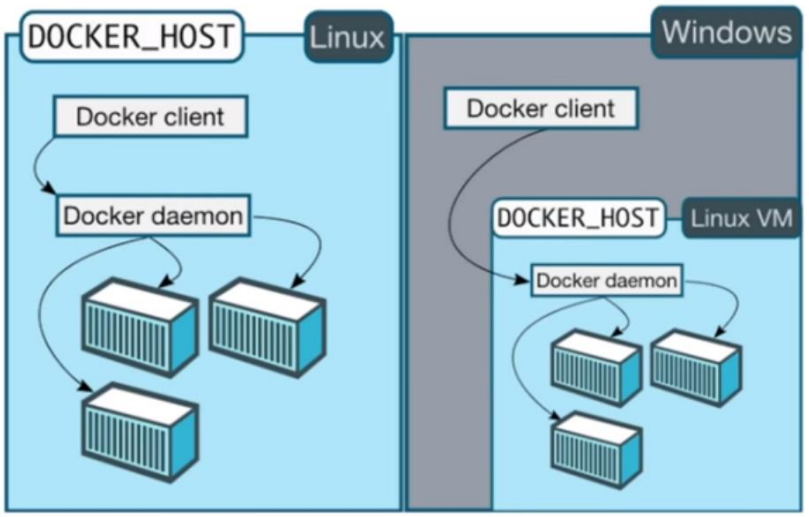
For removal also you can simply delete the container and it will not leave behind any files or traces on the system.



Productivity

Docker allows faster and more efficient deployments without worrying about running your app on different platforms.

It increases productivity many folds.



References - https://www.wedidknow.xyz/2017/03/docker-ecosystem-how-to-manage-your.html

Containers and images created with Docker for Windows are shared between all user accounts on machines where it is installed. This is because all Windows accounts use the same VM to build and run containers.

Basic Commands-

: docker version

: docker -v

: docker infod

: docker --help

: docker login

————————————

Images

: docker images

: docker pull

: docker rmi

————————————

Containers

: docker ps

: docker run

: docker start

: docker stop

————————————

System

: docker stats

: docker system df

: docker system prune

- Dangling images- which are not associated with running container
- A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.

Features of Containers:

- Are lightweight
- Fewer resources are used
- Booting of containers is very fast
- Can start, stop, kill, remove containers easily and quickly
- Operating System resources can be shared within Docker
- Containers run on the same machine sharing the same Operating system Kernel, this makes it faster