Docker

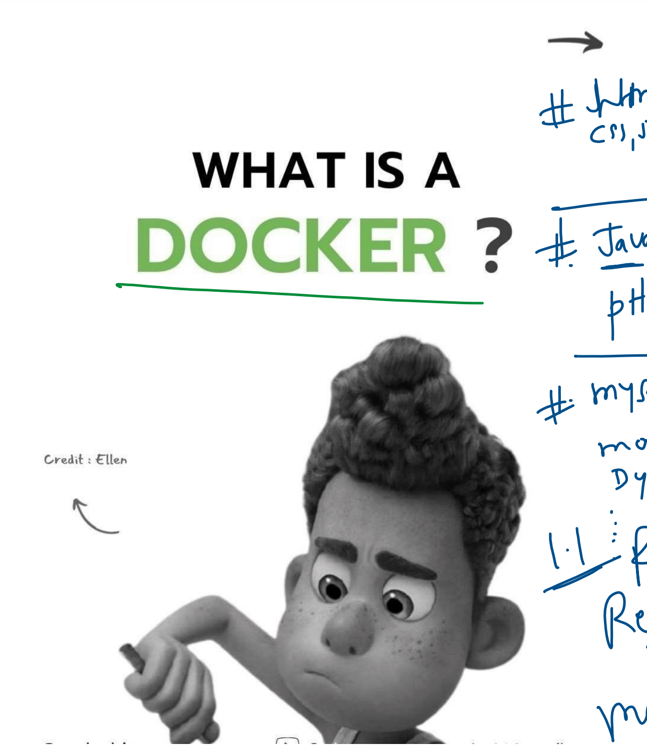
08:50 AM

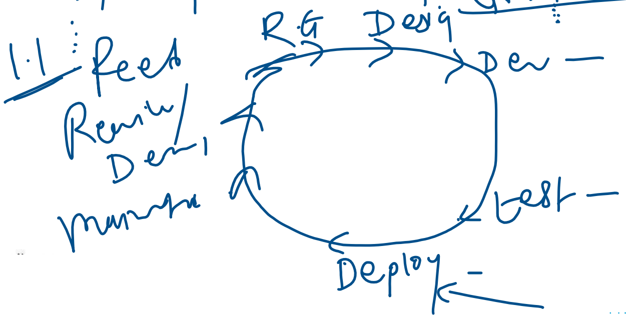
JSP

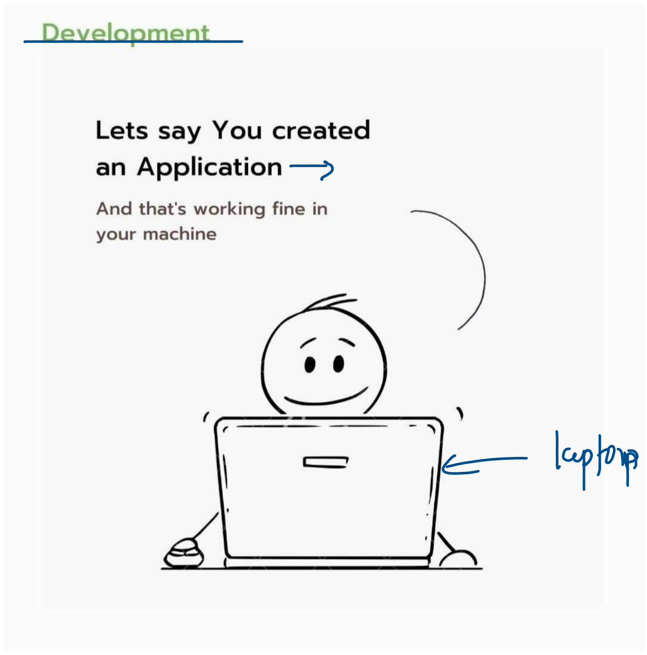
train - ticket sys → C#, Java, nodejs

Dev	Ops
3 tier FE, BE, D	ansible, chef, puppet,
# html, angular, ReactJs, css, js, Vue, nodejs	CI/CD - artifacts
# Java, C#, PY, Ruby, php, Scala, go, R	Jenkins, Bamboo,
# mysql, sql, ~~ps-sql~~, mon… mongodb, postgres, DynoDb, Redis, Couch…	monitoring -
	Dynatrace, splunk, an…
	Automation →
	Devlivery of appn
	Team task → ALM
	JIRA, Rally, Trello

te… Tea… tou…


WHAT IS A DOCKER ?




Development

Lets say You created an Application →

And that's working fine in your machine

VM – Cloud – AWS, Azure, GCP

VM – private/On-premis cloud

TH → Ram, P

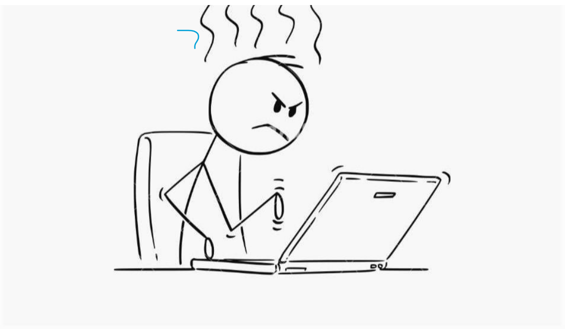
Production

But in Production it doesn't work properly

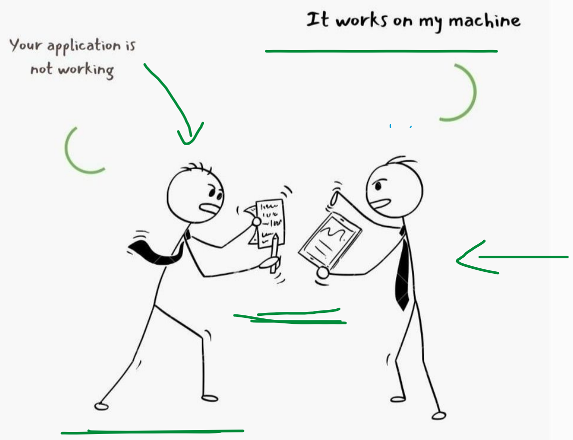
Developers experince it a lot

We'd love your feedback!

We have just two questions for you.



That is when the Developer's famous words are spoken



The Reason could be due to :

- Dependencies
- Libraries and versions
- Framework
- OS Level features
- Microservices

That the developers machine has but not there in the **production environment**

DOCKER

We need a standardized way to package the application with its **dependencies** and deploy it on any environment.

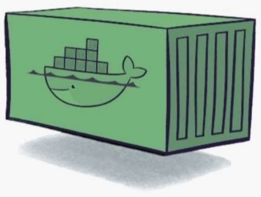
We'd love your feedback!

We have just two questions for you.

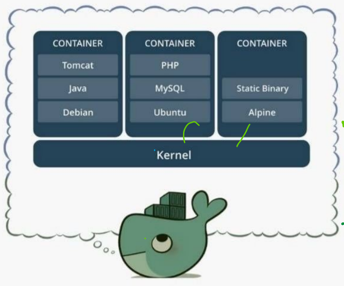
Docker is a tool designed to make it easier to **create**, **deploy**, and **run** applications by using **containers**.

How Does Docker Work?

Docker packages an application and all its **dependencies** in a **virtual container** that can run on any ~~Linux~~ server.

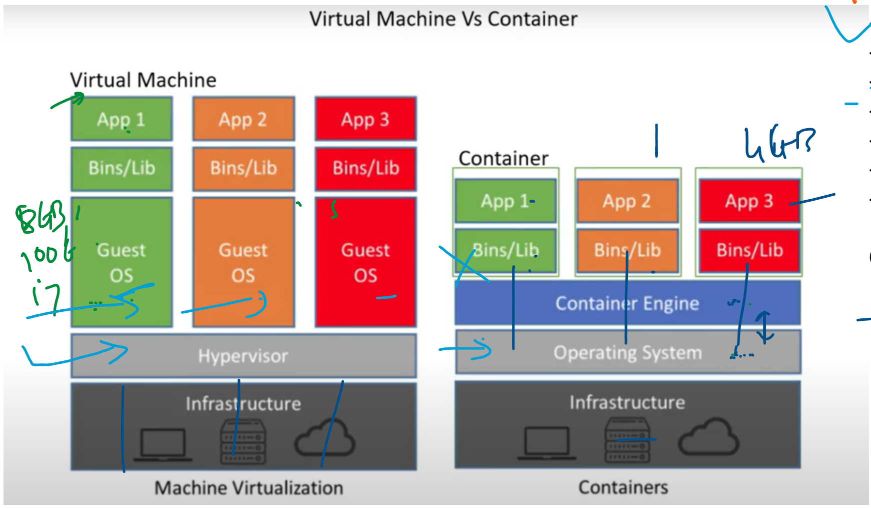


Each container runs as an **isolated process** in the user space and take up **less space** than regular VMs due to their layered architecture.



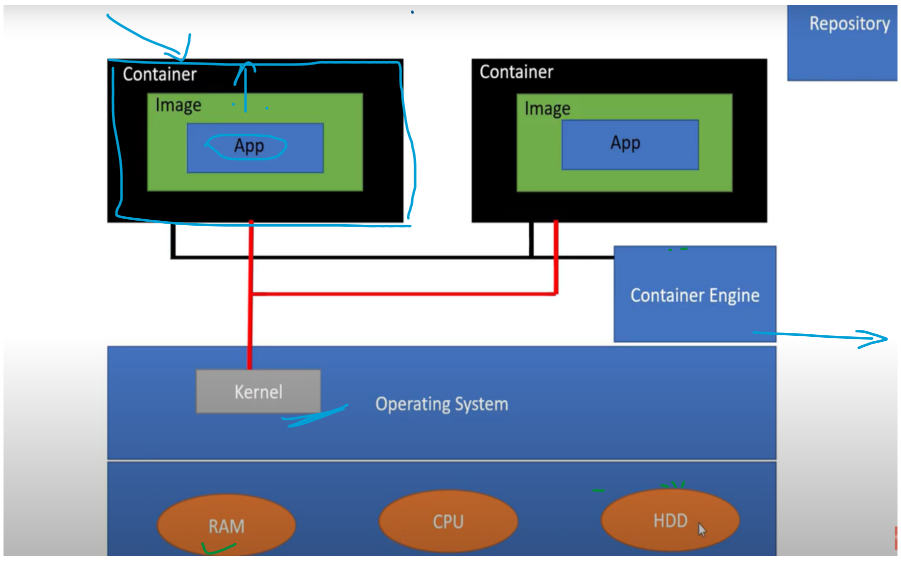
So it will always work the same
regardless of its environment

→ standerise?
→ packageing
→ Application
↳ Required
Rep;l
JDK, Ubun
image



- Physical server
- Hypervisor do virtualization of physical resou
- Separate OS
- Traditional full H/W was not getting consume
- Single phy box converted multiple vm by hyp
- To run app whole OS not required only few E
- OS License, resourse getting west in case of

Container
- Physical machine has RAM,CPU,H/D
- OS Install
- Container eng / docker
- Only required files to run app
- Size in MB, maintaine easy
- Isolation with by creating container
- Each container shares the host OS kerne
- OS getting virtualized into multiple contair



Inside container loads image.
- Img contains sys lib, sys tools, othe to run

Container share os kernel. (interact w h/w, os etc)

Kernal Main features
- Manage RAM Mem, all prog and
- Manage the processor time,
- Manage access and use diff peri comp
-

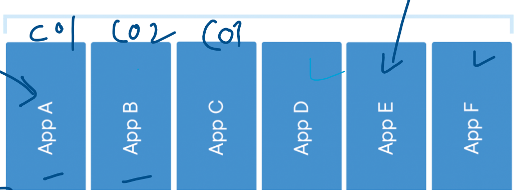
1)
2)
↳ 3)

Container images become containers at runtime and in the case of Docker containers - images become containers when they run on [Docker Engine](). Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

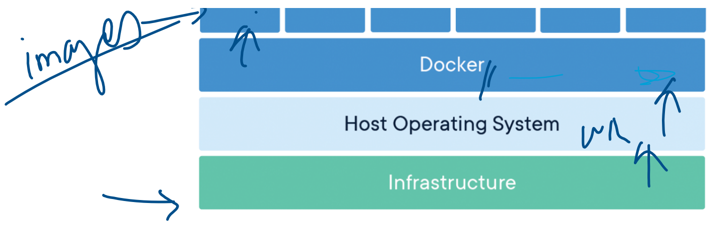
Docker containers that run on I

- **Standard:** Docker created the indu could be portable anywhere
- **Lightweight:** Containers share the therefore do not require an OS per efficiencies and reducing server a
- **Secure:** Applications are safer in c strongest default isolation capabi

Containerized Applications



We'd love your feedback!
We have just two questions for you.



1) without images ? X
running copy of ima

We'd love your feedback!

We have just two questions for you.