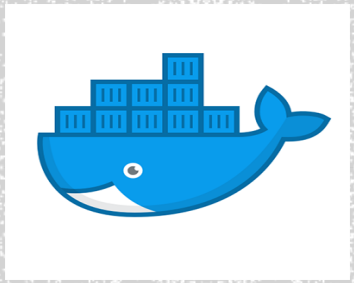
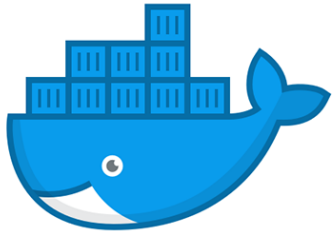
DOCKER



Presented By : Amrit Choudhary

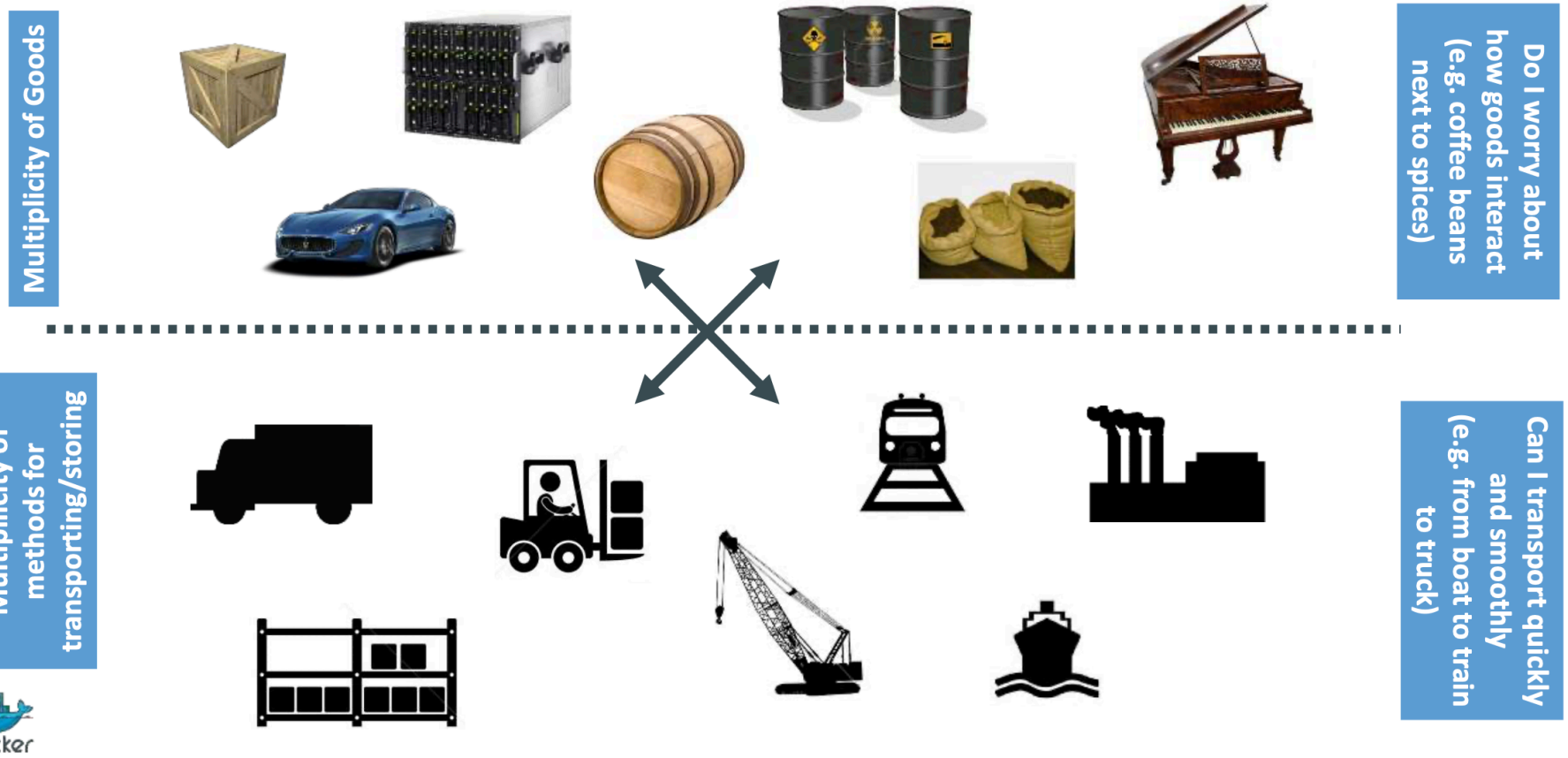
DOCKER



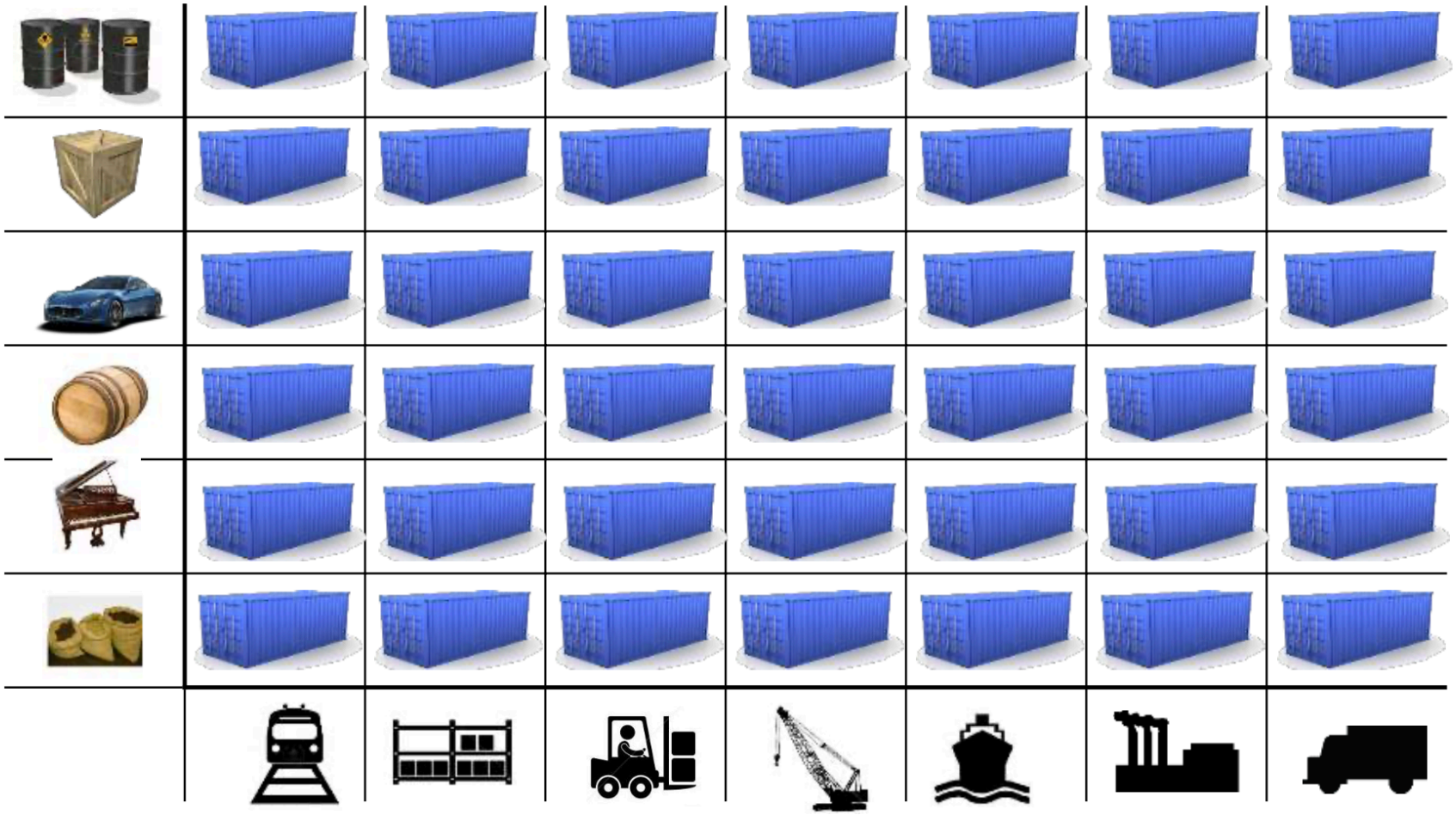
- What's was the Challenge ?
- Solution ?
- Why Docker , containers matter ?
- How they work ?

ANALOGY

Cargo Transport Pre-1960



ANALOGY



ANALOGY

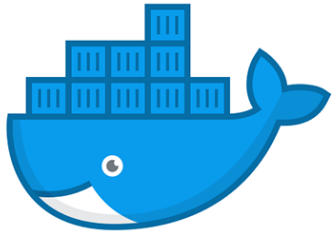
- 90% of all cargo now shipped in a standard container
- Order of magnitude reduction in cost and time to load and unload ships
- Massive reduction in losses due to theft or damage
- Huge reduction in freight cost as percent of final goods (from >25% to <3%)

→ massive globalizations

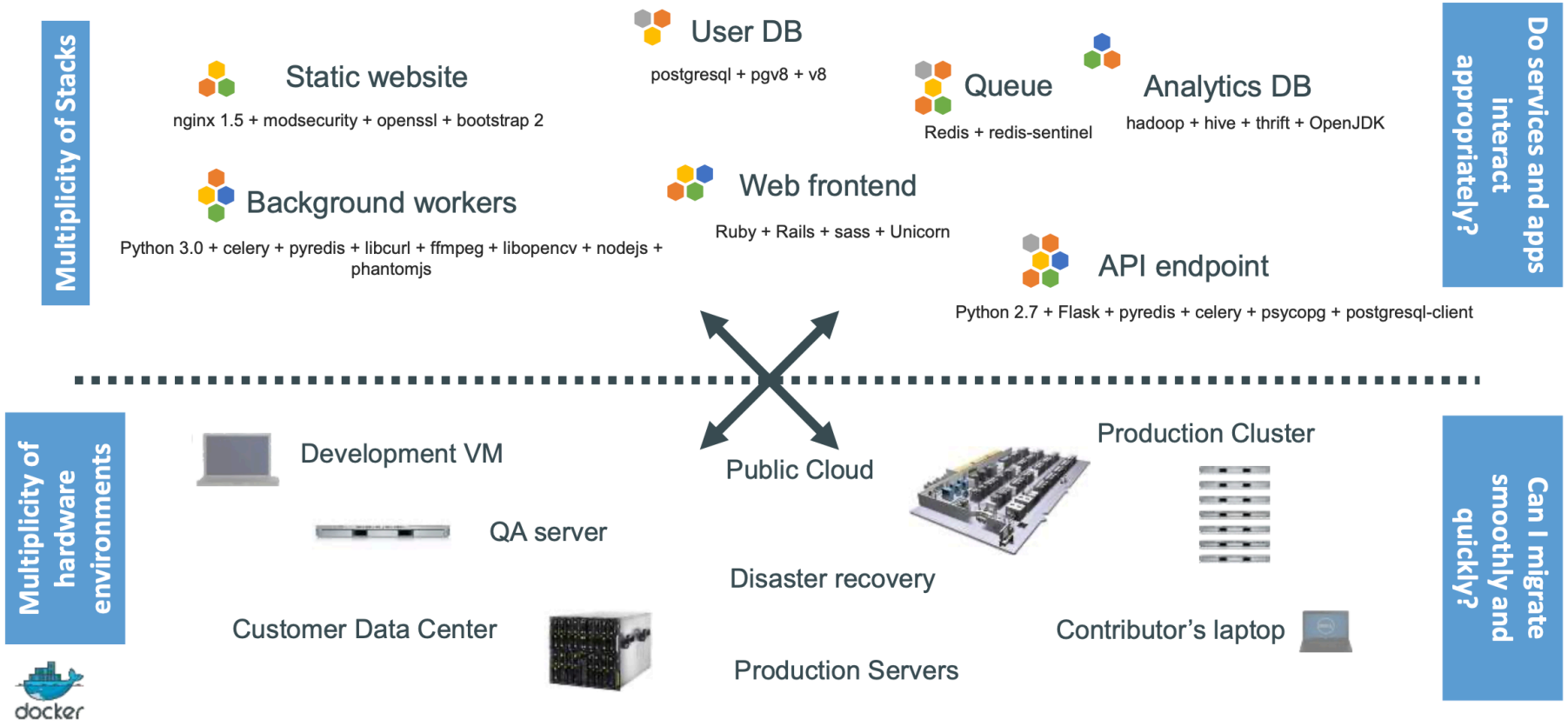
- 5000 ships deliver 200M containers per year

docker

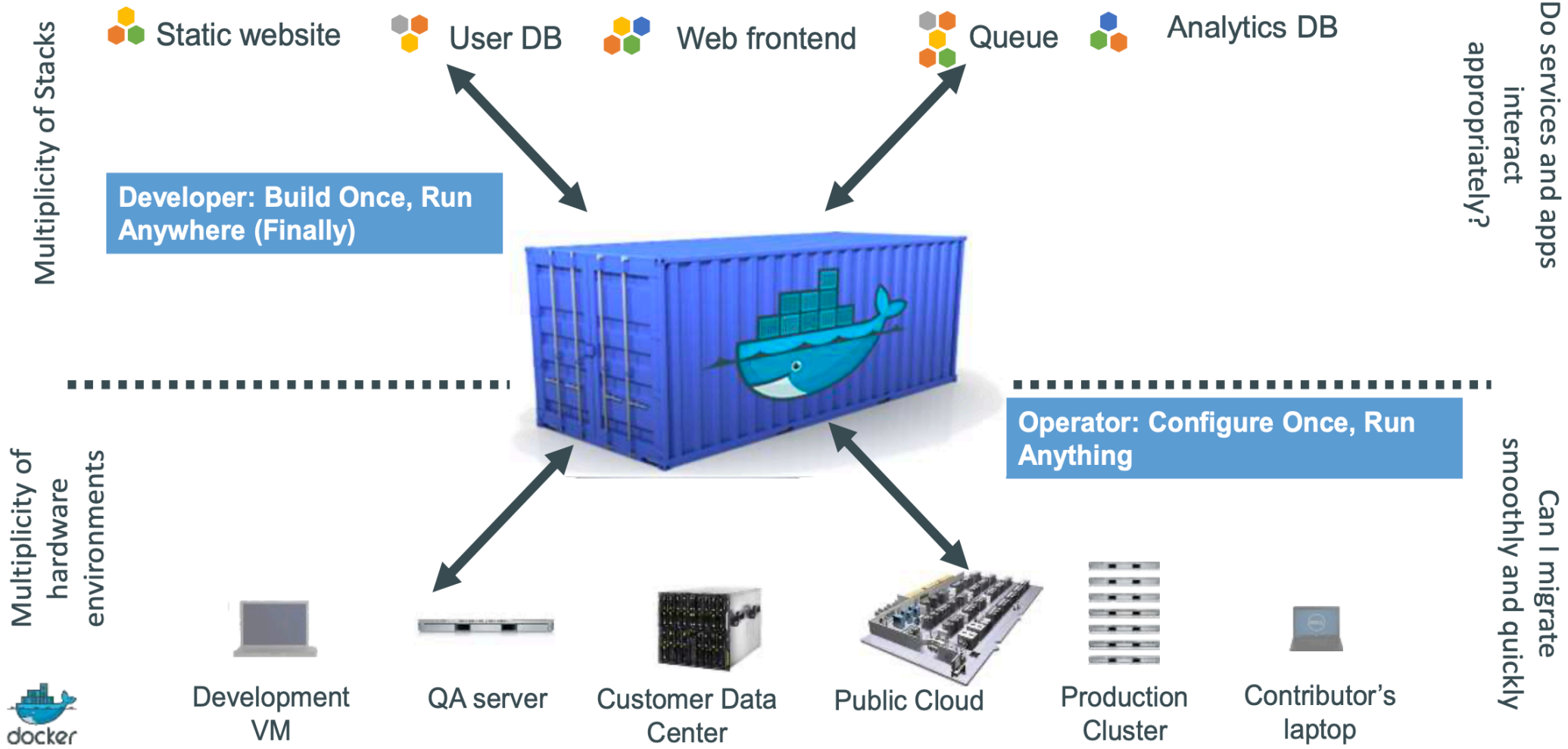
CHALLENGE



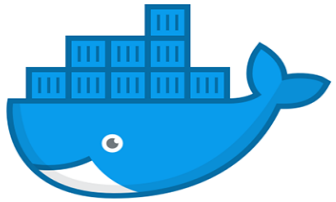
The Challenge



SOLUTION

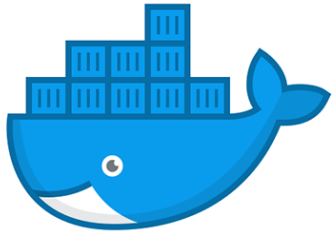

Multiplicity of Stacks
Static website
User DB
Web frontend
Queue
Analytics DB
Do services and apps interact appropriately?
Developer: Build Once, Run Anywhere (Finally)
Operator: Configure Once, Run Anything
Multiplicity of hardware environments
Can I migrate smoothly and quickly
docker
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop

DOCKER/CONTAINER'S MATTER ?



	Physical Containers	Docker
Content Agnostic	The same container can hold almost any type of cargo	Can encapsulate any payload and its dependencies
Hardware Agnostic	Standard shape and interface allow same container to move from ship to train to semi-truck to warehouse to crane without being modified or opened	Using operating system primitives (e.g. LXC) can run consistently on virtually any hardware—VMs, bare metal, openstack, public IAAS, etc.—without modification
Content Isolation and Interaction	No worry about anvils crushing bananas. Containers can be stacked and shipped together	Resource, network, and content isolation. Avoids dependency hell
Automation	Standard interfaces make it easy to automate loading, unloading, moving, etc.	Standard operations to run, start, stop, commit, search, etc. Perfect for devops: CI, CD, autoscaling, hybrid clouds
Highly efficient	No opening or modification, quick to move between waypoints	Lightweight, virtually no perf or start-up penalty, quick to move and manipulate
Separation of duties	Shipper worries about inside of box, carrier worries about outside of box	Developer worries about code. Ops worries about infrastructure.

DOCKER & DOCKER ENGINE

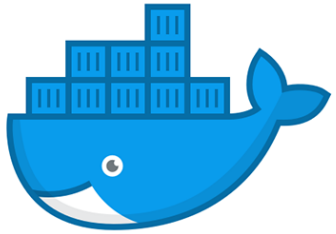


Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers.

Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and therefore use fewer resources than virtual machines.

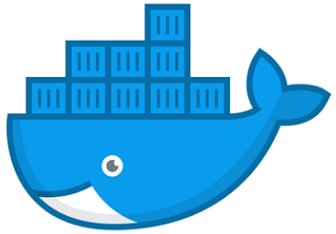
Docker Engine : The software that hosts the containers is called **Docker Engine**

UNDERLYING TECHNOLOGIES



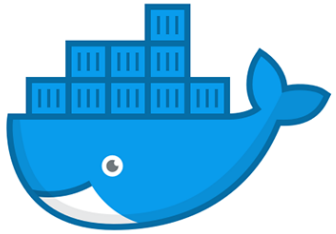
- By default daemon uses “execution driver” to create containers. By default this dockers own runc driver (docker’s own) , legacy support is lxc.
- Runc is tied to the following kernel features :
- i) cgroups : responsible for managing resources used by container (eg. Cpu, memory usage)
- ii) namespaces : responsible for isolating containers; makes sure container’s filesystem, hostname, users, networking, and processes are separated from the rest of the system.

DO DEVELOPERS CARE ?



- Build Once run anywhere
- Don’t worry about package, libraries, dependencies etc. (PAIN POINTS)
- RUN EACH APP in ISOLATED CONTAINERS, enables to run varies versions of libraries, dependencies etc.
- REDUCE/ELIMINATE CONCERNS OF ENV | PLATFORM COMPATIBILITY
- RUNS within a few secs

DO DEVOPS CARE ?



- CONFIGURE ONCE, RUN ANYWHERE …
- Make the entire lifecycle more efficient, consistent, and repeatable
- Increase the quality of code produced by developers
- Eliminate inconsistencies between development, test, production, and customer environments
- Support segregation of duties
- Significantly improves the speed and reliability of continuous deployment and continuous integration systems
- Because the containers are so lightweight, address significant performance, costs, deployment, and portability issues normally associated with VMs