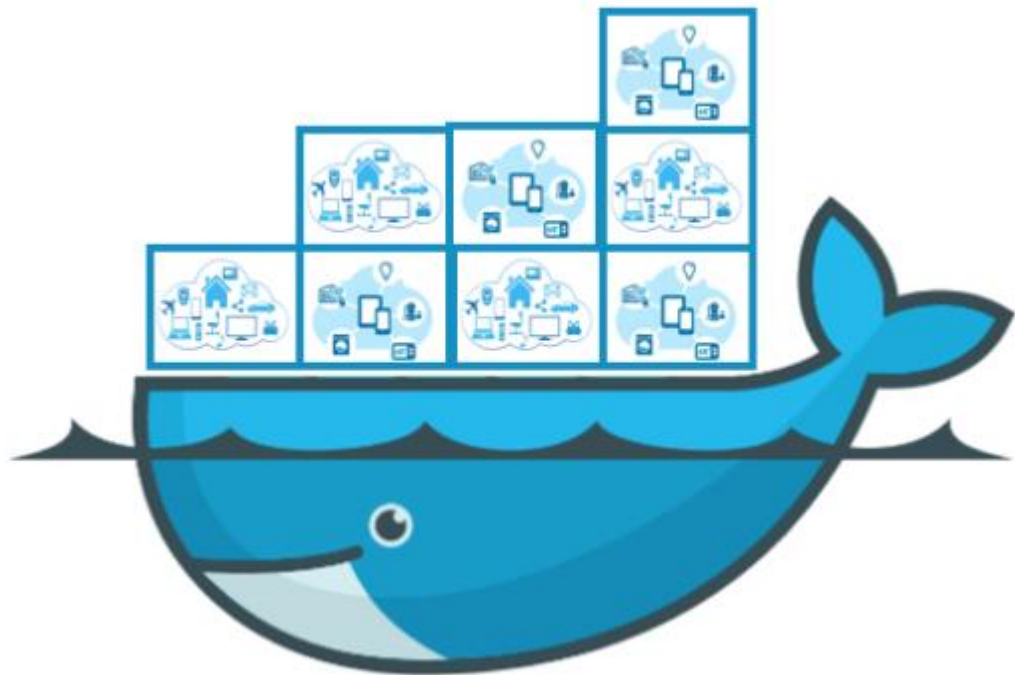


# Docker



# Agenda

- Introduction to Docker
- Docker Setup
- Introduction to Docker Components
- Docker Containers Introduction
- Docker Containers – inside & where to find them

# Software Engineering

"The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"

-- IEEE

# Evolution of IT

## Development Process

Waterfall



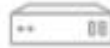
## Application Architecture

Monolithic



## Deployment & Packaging

Physical Servers



## Application Infrastructure

Datacenter



# Evolution of IT

## Development Process

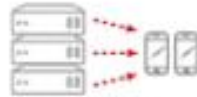
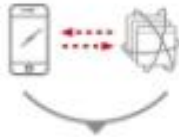
Waterfall



Agile

## Application Architecture

Monolithic



N-Tier

## Deployment & Packaging

Physical Servers



Virtual Servers

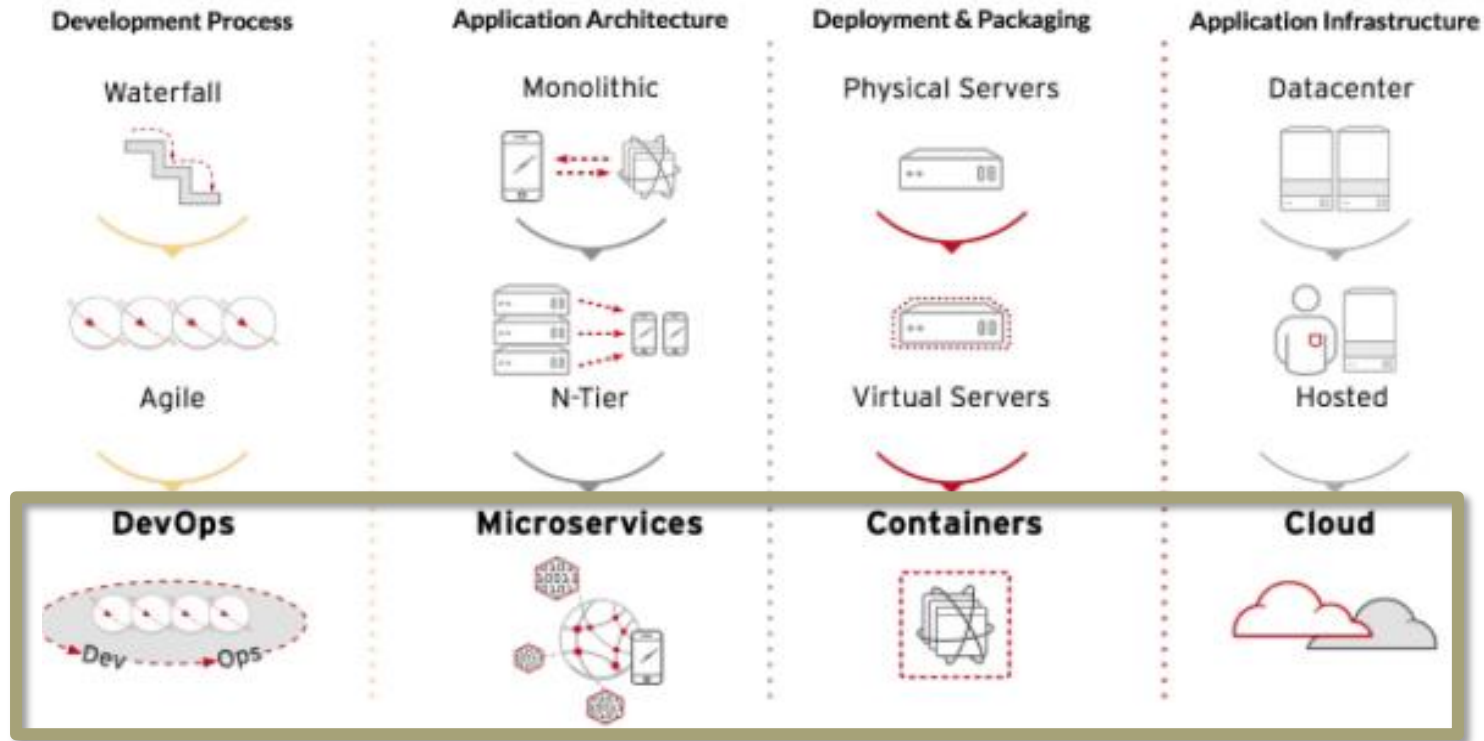
## Application Infrastructure

Datacenter



Hosted

# Evolution of IT



# Containers ?



# Cargo Transport Pre-1960

Multiplicity of Goods



Do I worry about  
how goods interact  
(e.g. coffee beans  
next to spices)

Multiplicity of  
methods for  
transporting/storing
















Can I transport quickly  
and smoothly  
(e.g. from boat to train  
to truck)






















































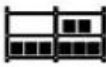





# Then we have NxN Matrix

	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							

# Solution: Intermodal Shipping Container



# This eliminated the NXN Problem....

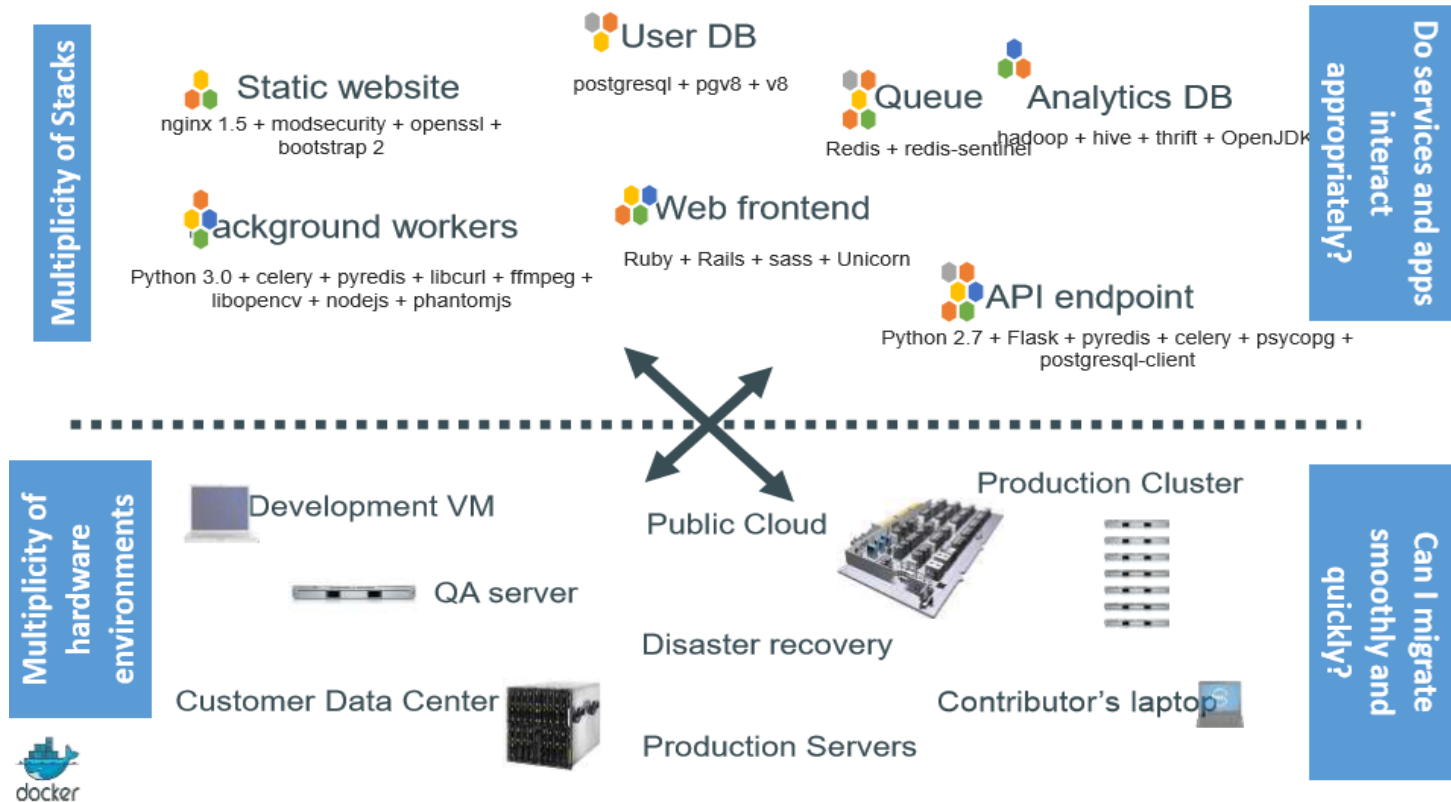
# Intermodal Shipping Container Ecosystem



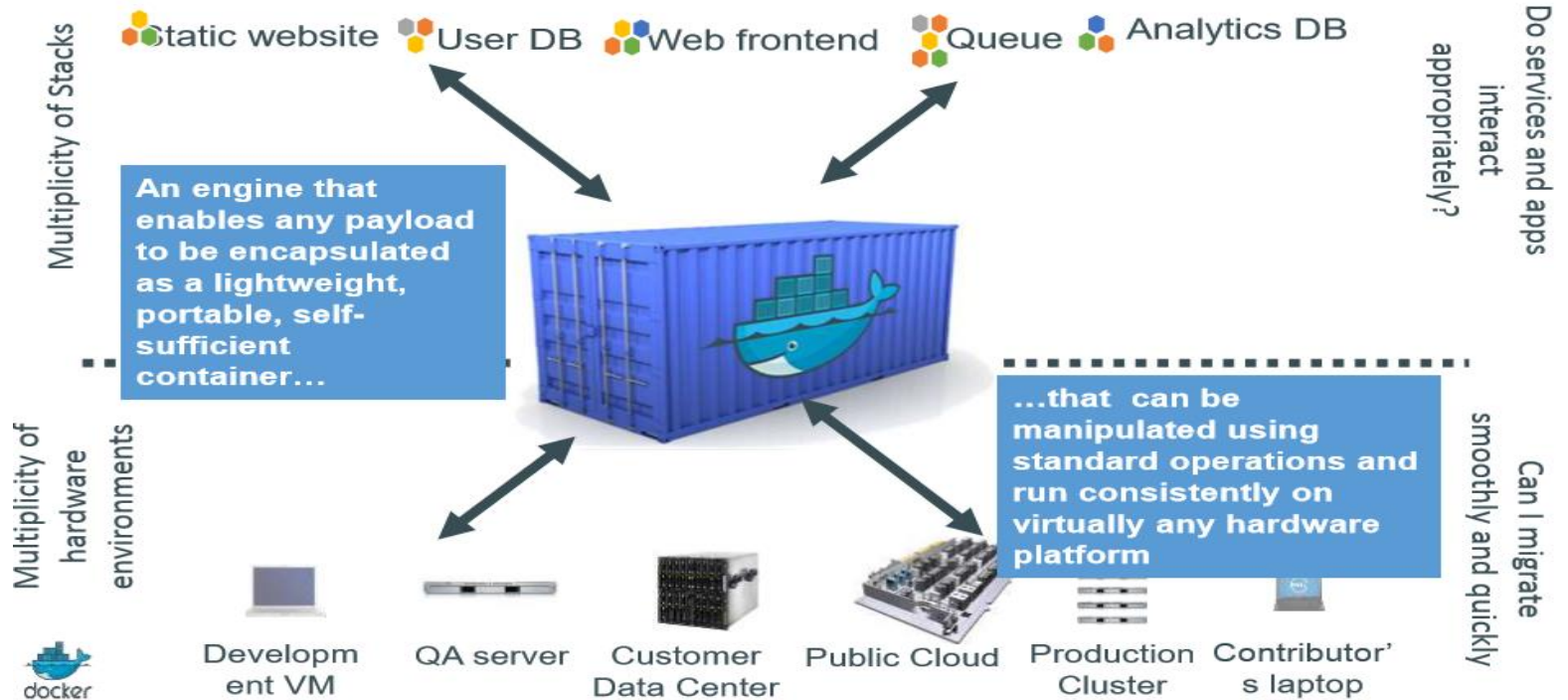
- 90% of all cargo now shipped in a standard container
- Order of magnitude reduction in cost and time to load and unload ships
- Massive reduction in losses due to theft or damage
- Huge reduction in freight cost as percent of final goods (from >25% to <3%)

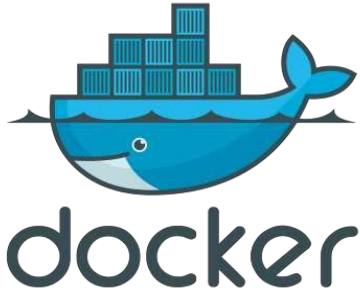
# **Why We Need Software CONTAINERS ?**


# Present Software delivery mechanism



# We need a shipping container system





Brings the power of creating  a for code & all of its dependencies together, which can be shipped across the environments.

## So Docker CONTAINER

Is an Isolated run time environment of a service/code with all dependencies, but shares OS and appropriate bins / libraries



# What is Docker

The Docker is Enterprise container platform that delivers immediate value to your business by reducing the infrastructure and maintenance costs of supporting your existing application portfolio while accelerating your time to market for new solutions.

Docker is a software platform to create, deploy and manage virtualized application containers on a common operating system (OS), with an ecosystem of associated tools.

# Existing System

- Monolithic applications
- long development cycles
- Standard Environment
- Slowly scaling up

# The deployment problem

Multiplicity of Stacks

Do services and apps  
interact  
appropriately?



Static website

~~nginx~~ 1.5 + ~~modsecurity~~ + ~~openssl~~ + bootstrap 2



Background workers

Python 3.0 + celery + ~~redis~~ + ~~libcurl~~ + ~~flup~~ + ~~libpcre~~ + ~~codecs~~ + ~~phantomjs~~



User DB

~~postgresql~~ + pg+8 + +8



Web frontend

Ruby + Rails + sass + Unicorn



Queue

~~Rails~~ + ~~redis~~ + ~~sidekiq~~



Analytics DB

~~hadoop~~ + ~~hive~~ + ~~thrift~~ + ~~OpenDK~~



API endpoint

Python 2.7 + Flask + ~~redis~~ + celery + ~~pycrypto~~ + ~~postgresql-client~~



# Current Need....

- Decoupled services
- Fast, iterative improvements
- Multiple environments
- Quickly scaling out

# Intermodal shipping containers



# Still WHY ??



# How it Works ??



## Developers

Tooling that is simple to use, yet powerful and delivers a great user experience so you can focus on what you love - writing great code.



## IT Operations

Docker delivers an enterprise-ready container platform to deploy and run applications in a way that makes the best sense for your customers and business.



## Business Leader

Docker provides a platform to drive your digital transformation by accelerating new innovation while dramatically driving down your existing IT costs.

# It Makes a difference

## The Docker Enterprise Difference

Leading companies rely on our container platform to build, manage and secure all their applications from traditional applications to cutting-edge microservices — and deploy them anywhere.

### Choice

Eliminates risk by enabling you to start big or small, with legacy or new applications, using any operational model, on any OS, across any infrastructure, whether it be on prem or across multiple clouds with the same Docker experience throughout.

### Agility

Unifies processes across any architecture, while aligning with existing IT operations so organizations can get applications to market faster, reduce TCO and ease the adoption of new technology as business needs evolve over time.

### Security

Incorporates security at every step of the application delivery lifecycle without getting in your way or adding extra cost. Applications receive greater protection while maintaining performance, improving governance and enabling policy-driven automation.



# A Survey....

- According to a 2018 [IDC](#) (International Data Corporation) [survey on containers](#), 85% of container adopters are using containers for production apps and 83% of container deployers use containers on multiple public clouds (3.7 clouds on average) with 55% of containers running on-premises and 45% run in the public cloud. As organizations continue to scale their container environments in dynamic IT environments they require a solution to address their security, management, and governance needs while maintaining operational efficiency and developer agility.

# History....

- Docker was released as open source in March 2013.
- On March 13, 2014, with the release of version 0.9, Docker dropped LXC as the default execution environment and replaced it with its own lib container library written in the [Go](#) programming language.]
- On September 19, 2013, [Red Hat](#) and Docker announced a collaboration around [Fedora](#), [Red Hat Enterprise Linux](#), and [OpenShift](#).
- In November 2014 Docker container services were announced for the [Amazon Elastic Compute Cloud](#) (EC2).
- On November 10, 2014, Docker announced a partnership with [Stratoscale](#).
- On December 4, 2014, [IBM](#) announced a strategic partnership with Docker that enables Docker to integrate more closely with the IBM Cloud.
- On June 22, 2015, Docker and several other companies announced that they are working on a new vendor and operating-system-independent standard for software containers.
- As of October 24, 2015, the project had over 25,600 [GitHub](#) stars (making it the 20th most-starred GitHub project), over 6,800 forks, and nearly 1,100 contributors.
- A May 2016 analysis showed the following organizations as main contributors to Docker: The Docker team, [Cisco](#), [Google](#), [Huawei](#), [IBM](#), [Microsoft](#), and [Red Hat](#).
- On October 4, 2016, Solomon Hykes announced [InfraKit](#) as a new self-healing container infrastructure effort for Docker container environments.
- A January 2017 analysis of [LinkedIn](#) profile mentions showed Docker presence grew by 160% in 2016.
- The software has been downloaded more than 13 billion times as of 2017.

# Editions

- Docker is available in two editions:
- Community Edition (CE)
- Enterprise Edition (EE)
- Docker Community Edition (CE) is ideal for individual developers and small teams looking to get started with Docker and experimenting with container-based apps.
- Docker Enterprise Edition (EE) is designed for enterprise development and IT teams who build, ship, and run business critical applications in production at scale.

Capabilities	Community Edition	Enterprise Edition Basic	Enterprise Edition Standard	Enterprise Edition Advanced
Container engine and built in orchestration, networking, security	✓	✓	✓	✓
Certified infrastructure, plugins and ISV containers		✓	✓	✓
Image management			✓	✓
Container app management			✓	✓
Image security scanning				✓

# Versions..

18.03.1-ee-3 (2018-08-30)	17.06.2-ee-5 (2017-11-02)	Docker v17.12
18.03.1-ee-2 (2018-07-10)	17.06.2-ee-4 (2017-10-12)	Docker v17.09
18.03.1-ee-1 (2018-06-27)	17.06.2-ee-3 (2017-09-22)	Docker v17.06
17.06.2-ee-16 (2018-07-26)	17.06.1-ee-2 (2017-08-24)	Docker v17.03
17.06.2-ee-15 (2018-07-10)	17.06.1-ee-1 (2017-08-16)	Docker v1.13
17.06.2-ee-14 (2018-06-21)	17.03.2-ee-9 (2018-08-30)	Docker v1.12
17.06.2-ee-13 (2018-06-04)	17.03.2-ee-8 (2017-12-13)	Docker v1.11
17.06.2-ee-12 (2018-05-29)	17.03.2-ee-7 (2017-10-04)	Docker v1.10
17.06.2-ee-11 (2018-05-17)	17.03.2-ee-6 (2017-08-24)	Docker v1.9
17.06.2-ee-10 (2018-04-27)	17.03.2-ee-5 (20 Jul 2017)	Docker v1.8
17.06.2-ee-9 (2018-04-26)	17.03.2-ee-4 (01 Jun 2017)	Docker v1.7
17.06.2-ee-8 (2018-04-17)	17.03.1-ee-3 (30 Mar 2017)	Docker v1.6
17.06.2-ee-7 (2018-03-19)	17.03.1-ee-2 (28 Mar 2017)	Docker v1.5
17.06.2-ee-6 (2017-11-27)		Docker v1.4

<https://docs.docker.com/ee/supported-platforms/>

## Pre-Reqs to install

- 1. Must be 64-bit
- 2. Kernal be more than 3.10

`uname -r`

`uname -m`

**Let us Install!!!**

## Introduction to Docker

- Docker is an open-source project that automates deployment and execution of template applications inside software containers
- Solomon Hykes started Docker as an internal project within dotCloud, a platform-as-a-service company

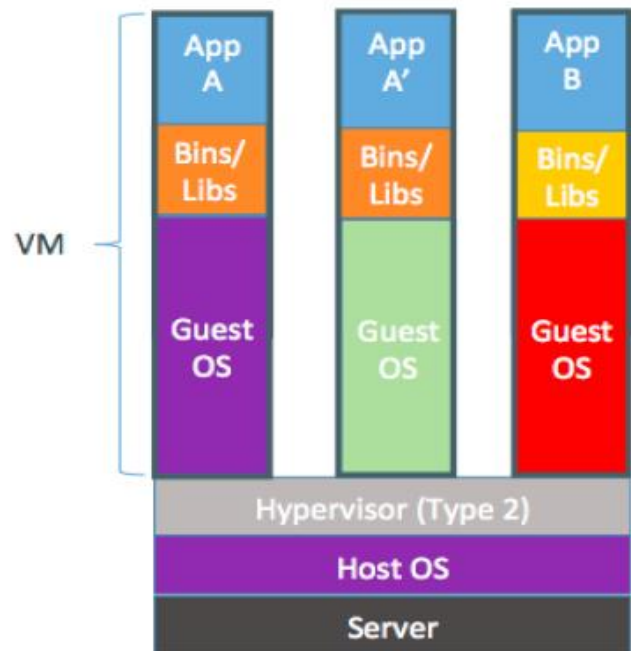
Docker was released as open source software in March 2013

Redat, IBM and Microsoft are using.

- The fundamental philosophy in Docker is “Develop; Ship; and Run Any Application, Anywhere”
- Docker helps in rapidly assembling applications from components eliminating friction and dependencies that come while shipping code to varied environments
- Docker allows code to be tested and deployed into production as fast as possible

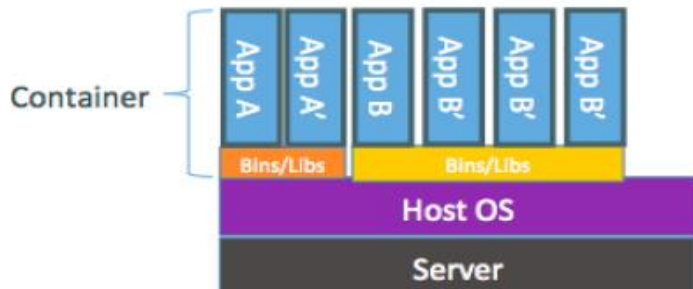


# Containers = cheaper than VMs



Containers are isolated, but share OS kernel and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart



# The origins of the Docker Project

- dotCloud was operating a PaaS, using a custom container engine.
- This engine was based on OpenVZ (and later, LXC) and AUFS.
- It started (circa 2008) as a single Python script.
- By 2012, the engine had multiple (~10) Python components.  
(and ~100 other micro-services!)
- End of 2012, dotCloud refactors this container engine.
- The codename for this project is "Docker."

# Docker becomes a platform

- The initial container engine is now known as "Docker Engine."
- Other tools are added:
  - Docker Compose (formerly "Fig")
  - Docker Machine
  - Docker Swarm
  - Kitematic
  - Docker Cloud (formerly "Tutum")
  - Docker Datacenter
  - etc.
- Docker Inc. launches commercial offers.

# About Docker Inc.

- Docker Inc. used to be dotCloud Inc.
- dotCloud Inc. used to be a French company.
- Docker Inc. is the primary sponsor and contributor to the Docker Project:
  - Hires maintainers and contributors.
  - Provides infrastructure for the project.
  - Runs the Docker Hub.
- HQ in San Francisco.
- Backed by more than 100M in venture capital.

# Introduction to Docker Components

Docker Hub ( public repo )

Docker Trusted Registry ( Private repo )

Docker Engine

Docker Container

Docker Image

Docker Swarm

Docker Services

Docker Compose / Docker Stack

Universal Control Pane ( UCP )

**Docker Image:** A package with all the dependencies and information needed to create a container. An image includes all the dependencies (such as frameworks) plus deployment and execution configuration to be used by a container runtime. Usually, an image derives from multiple base images that are layers stacked on top of each other to form the container's filesystem. An image is immutable once it has been created.

**Docker Container:** An instance of a Docker image. A container represents the execution of a single application, process, or service. It consists of the contents of a Docker image, an execution environment, and a standard set of instructions. When scaling a service, you create multiple instances of a container from the same image. Or a batch job can create multiple containers from the same image, passing different parameters to each instance.

**Repository (repo):** A collection of related Docker images, labeled with a tag that indicates the image version. Some repos contain multiple variants of a specific image, such as an image containing SDKs (heavier), an image containing only runtimes (lighter), etc. Those variants can be marked with tags. A single repo can contain platform variants, such as a Linux image and a Windows image.

**Docker Hub:** A public registry to upload images and work with them. Docker Hub provides Docker image hosting, public or private registries, build triggers and web hooks, and integration with GitHub and Bitbucket.

**Docker Trusted Registry (DTR):** A Docker registry service (from Docker) that can be installed on-premises so it lives within the organization's datacenter and network. It is convenient for private images that should be managed within the enterprise. Docker Trusted Registry is included as part of the Docker Datacenter product.

**Compose:** A command-line tool and YAML file format with metadata for defining and running multi-container applications. You define a single application based on multiple images with one or more .yaml files that can override values depending on the environment. After you have created the definitions, you can deploy the whole multi-container application with a single command (docker-compose up) that creates a container per image on the Docker host.

**Swarm :** A collection of Docker hosts exposed as if it were a single virtual Docker host, so that the application can scale to multiple instances of the services spread across multiple hosts within the cluster. Docker clusters can be created with Docker Swarm, Mesosphere DC/OS, Kubernetes, and Azure Service Fabric. (If you use Docker Swarm for managing a cluster, you typically refer to the cluster as a swarm instead of a cluster.)



**Orchestrator:** A tool that simplifies management of clusters and Docker hosts. Orchestrators enable you to manage their images, containers, and hosts through a command line interface (CLI) or a graphical UI. You can manage container networking, configurations, load balancing, service discovery, high availability, Docker host configuration, and more. An orchestrator is responsible for running, distributing, scaling, and healing workloads across a collection of nodes. Typically, orchestrator products are the same products that provide cluster infrastructure, like Mesosphere DC/OS, Kubernetes, Docker Swarm, and Azure Service Fabric.

**UCP:** Docker Universal Control Plane (UCP) is the enterprise-grade cluster management solution from Docker. You install it on-premises or in your virtual private cloud, and it helps you manage your Docker cluster and applications through a single interface.

- Centralized cluster management
- Deploy, manage, and monitor
- Built-in security and access control

**Tag:** A mark or label you can apply to images so that different images or versions of the same image (depending on the version number or the target environment) can be identified.

**Dockerfile:** A text file that contains instructions for how to build a Docker image.

**Build:** The action of building a container image based on the information and context provided by its Dockerfile, plus additional files in the folder where the image is built. You can build images with the Docker `docker build` command.

**Repository (repo):** A collection of related Docker images, labeled with a tag that indicates the image version. Some repos contain multiple variants of a specific image, such as an image containing SDKs (heavier), an image containing only runtimes (lighter), etc. Those variants can be marked with tags. A single repo can contain platform variants, such as a Linux image and a Windows image.

# Docker Containers

- A container 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.
- Containerized software will always run the same, regardless of the environment.
- Containers isolate software from its surroundings, for example differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure.

# Lets Run our First Container

```
root@ubuntu:~# docker container run -it ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
1be7f2b886e8: Pull complete
Digest:
sha256:e27e9d7f7f28d67aa9e2d7540bdc2b33254b452ee8e60f388875e5b7d9b2
b696
Status: Downloaded newer image for ubuntu:latest
```

```
root@2fc2c56c92e6:/# cat /etc/os-release
NAME="Ubuntu"
VERSION="16.04.3 LTS (Xenial Xerus)"
```

**Note: We are running a container, open it in interactive mode, and running a command**

# What docker really did ?

- Downloaded the image from Hub / Registry
- Generated a new container
- Created a new file system
- Mounted a read/write layer
- Allocated network interface
- Setup IP
- Setup NAT
- Executed bash shell in container

# Let's Try one more ?

```
root@ubuntu:~# docker container run -d -P nginx
```

```
Efe826d8fb9276a6bbcbba826769e344e7e1eaac98f23b449249646e2e73d7c  
6b
```

```
root@ubuntu:~# docker container ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
efe826d8fb92	nginx	"nginx -g 'daemon of..."	6 seconds ago
Up 4 seconds	0.0.0.0:32768->80/tcp	peaceful_chatterjee	

```
root@ubuntu:~#
```

You must see something like: 0.0.0.0:32768->80/tcp

Go to web browser and enter url: <IPAddr>:32768

# Container Commands

<code>docker container ps</code>	→ list all running containers
<code>docker container ps -a</code>	→ list all running & dead containers
<code>docker container commit</code>	→ Create a new image from a container's changes
<code>docker container cp</code>	→ Copy files/folders between a container and the local filesystem
<code>docker container create</code>	→ Create a new container
<code>docker container exec</code>	→ Run a command in a running container
<code>docker container export</code>	→ Export a container's filesystem as a tar archive
<code>docker container inspect</code>	→ Display detailed information on one or more containers
<code>docker container kill</code>	→ Kill one or more running containers
<code>docker container logs</code>	→ Fetch the logs of a container
<code>docker container ls</code>	→ List containers
<code>docker container pause</code>	→ Pause all processes within one or more containers
<code>docker container port</code>	→ List port mappings or a specific mapping for the container
<code>docker container prune</code>	→ Remove all stopped containers
<code>docker container rename</code>	→ Rename a container
<code>docker container restart</code>	→ Restart one or more containers
<code>docker container rm</code>	→ Remove one or more containers
<code>docker container run</code>	→ Run a command in a new container

**Continues.....**