Getting Started with Docker

$whoami

Principal Development Engineer at DellEMC

1st half of my career was in CGI & VMware

2nd half of my career has been in System Integration Testing

Docker Captain (since 2016)

Docker Bangalore Meetup Organizer (8000+ Registered Users)

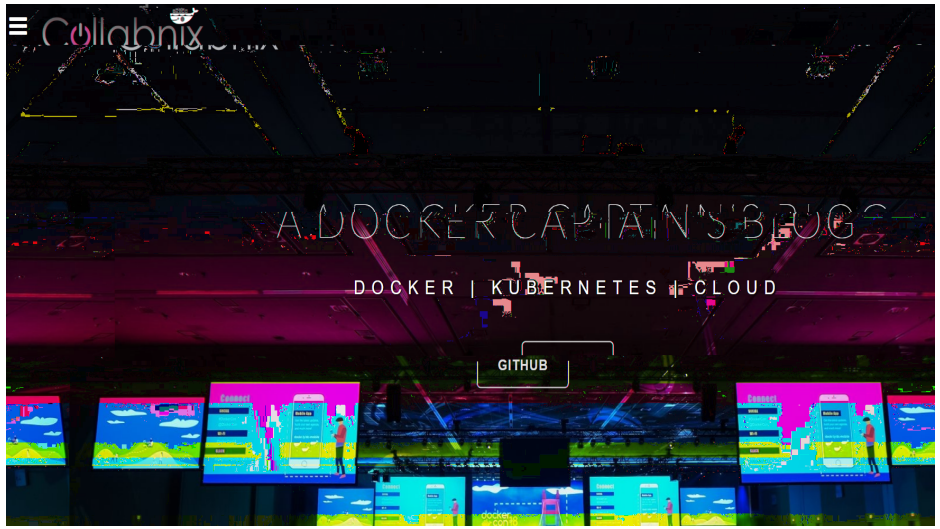
DockerLabs Incubator

Ajeet Singh Raina
Twitter: @ajeetsraina
GitHub: ajeetraina

vmware DELL EMC

$curl www.collabnix.com

- My Personal Blog website
- Built on WordPress, Hosted on One.com
- Recently Completed 4th Year
- 175+ blogs on Docker | Kubernetes | Cloud
- Monthly close to 1 million visitors per month



Agenda

- A Shift from Monolithic to Microservices Architecture
- Traditional Software Development Workflow(with/without Docker)
- What is Docker? What problem does it solve for us?
- Docker Vs Linux Containers
- Docker Vs Virtual Machine
- Docker Underlying Technology
- Running Your First Docker Container - Demo
- Using Docker: Build, Ship and Run Workflow - Demo
- Building Your Docker Image using Dockerfile – Demo
- Introduction to Docker Compose – Demo
- Docker Workshop

A Shift from Monolithic to Microservice Architecture

Applications have changed dramatically

Monolithic

Slow Changing

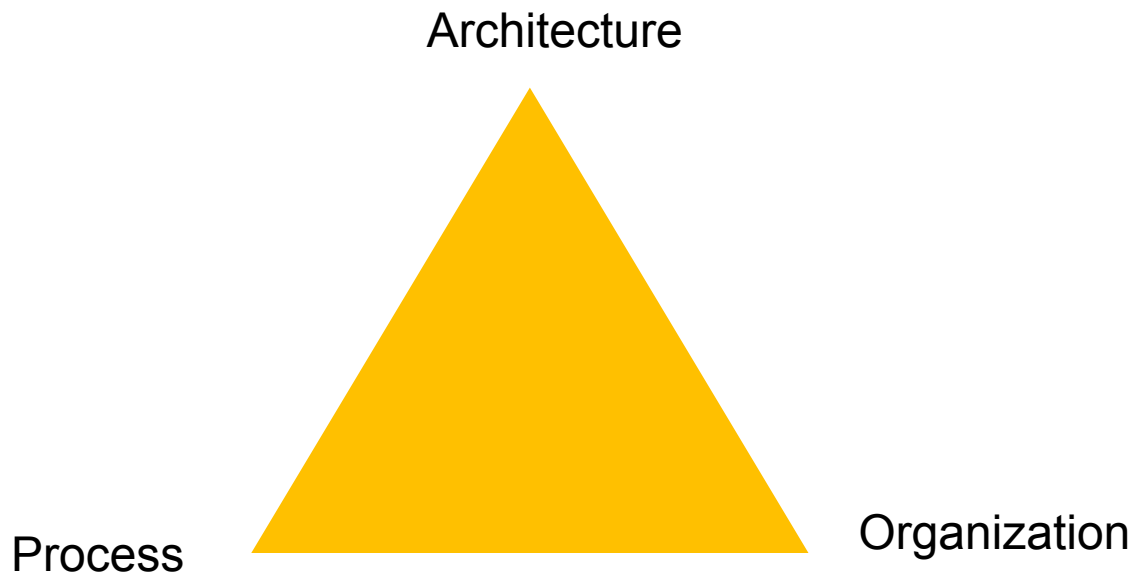
Big Server

A Decade Ago (and still valid)

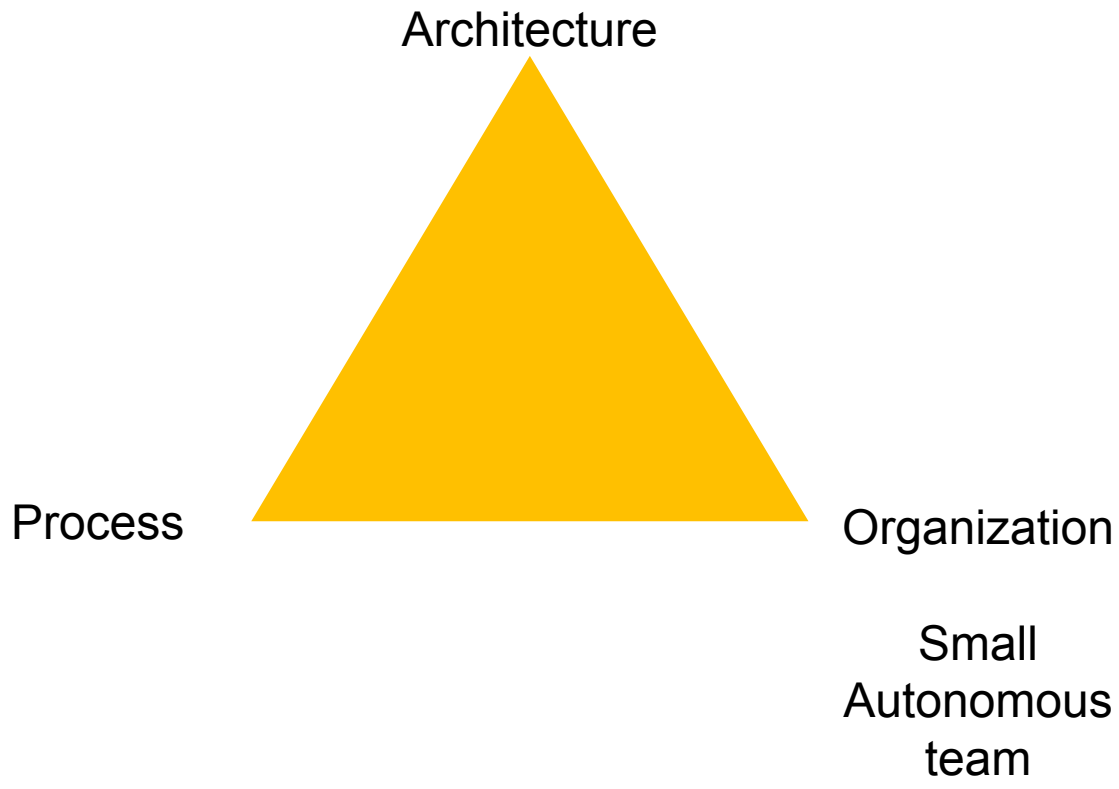
- Apps were monolithic
- Built on a single stack such as .NET or Java
- Long Lived
- Deployed to a single server

Let's imagine you are building a large, complex application, e.g., Online Store

Successful Software Development

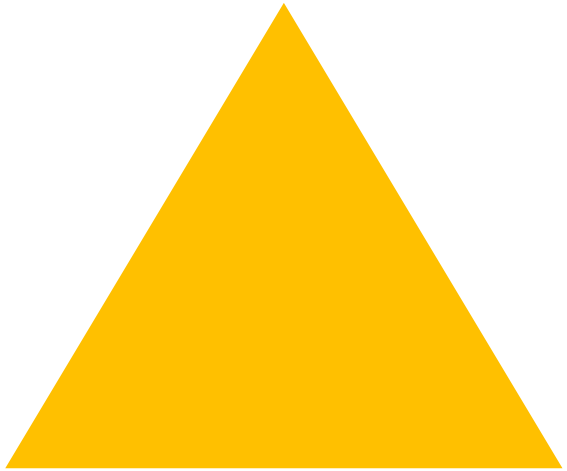


Successful Software Development



Successful Software Development

Architecture



Process

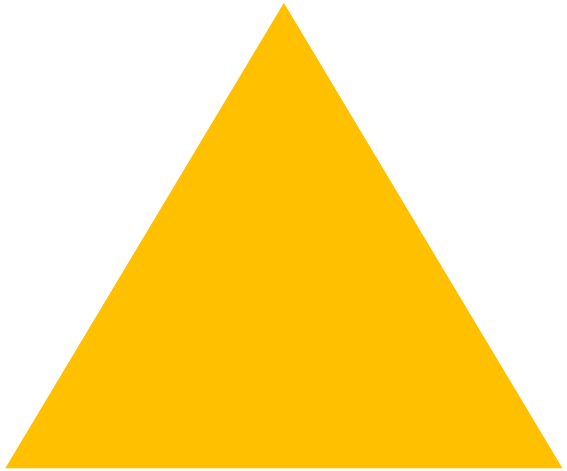
Agile
Continuous Delivery

Organization

Small
Autonomous
team

Successful Software Development

Architecture
Monolthic Vs Microservices



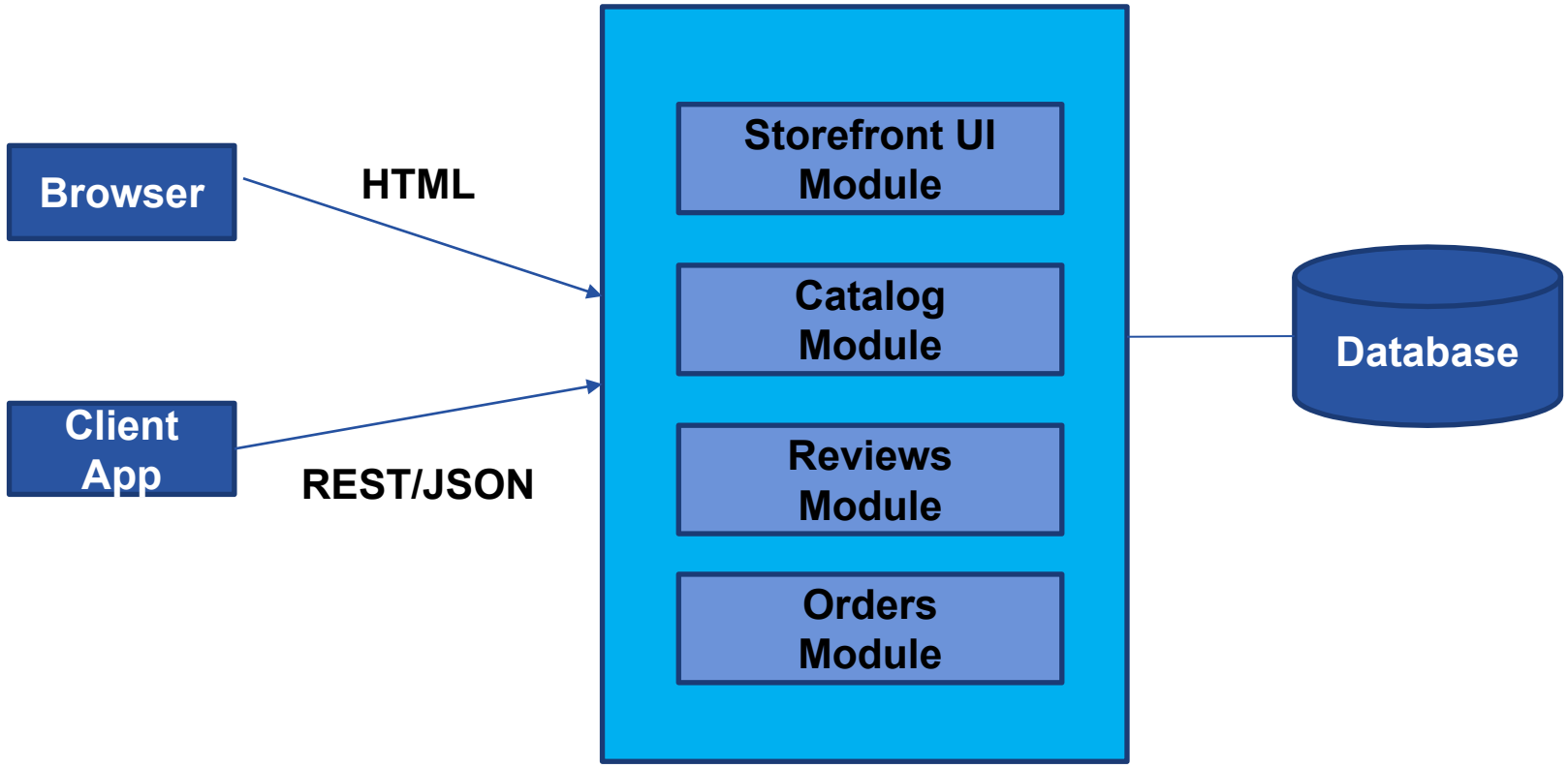
Process

Agile
Continuous Delivery

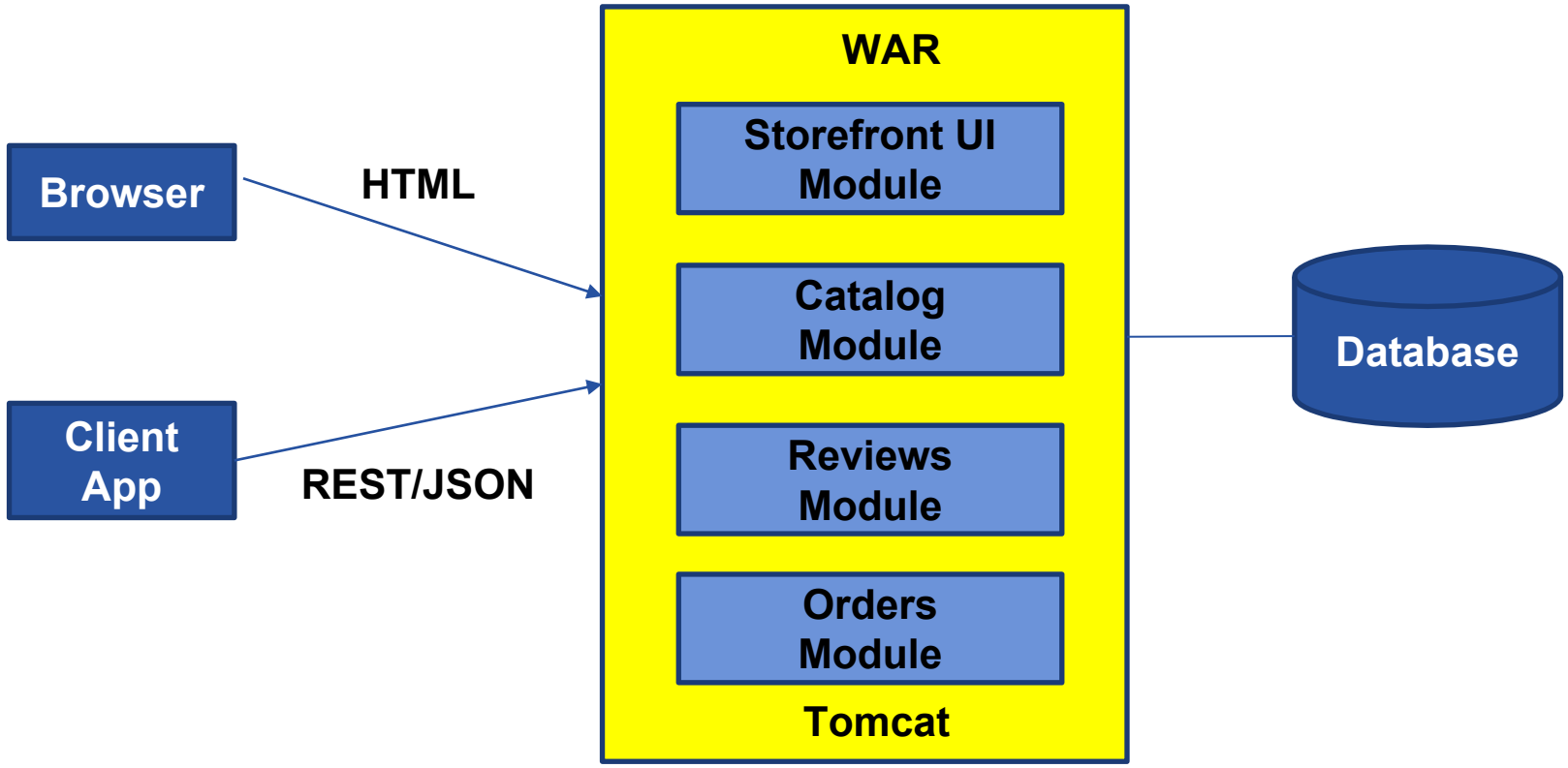
Organization

Small
Autonomous
team

A Close Look at Monolithic



A Close Look at Monolithic



Benefits of Monolith

Simple to Develop, Test, Deploy & Scale

- Simple to develop because of all the tools and IDEs support to that kind of application by default.
- Easy to deploy because all components are packed into one bundle.
- Easy to scale the whole application.

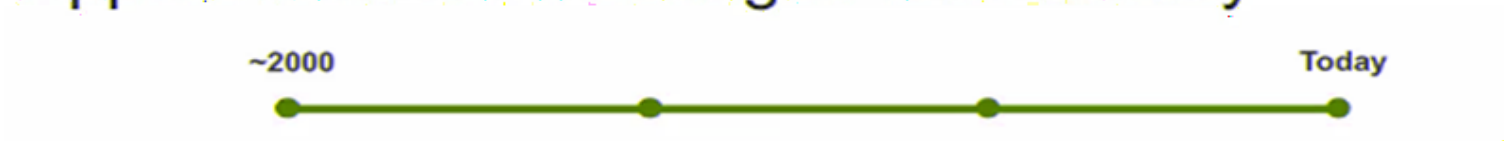
But successful applications keep growing ....

@crichardsor

Disadvantages of Monolith

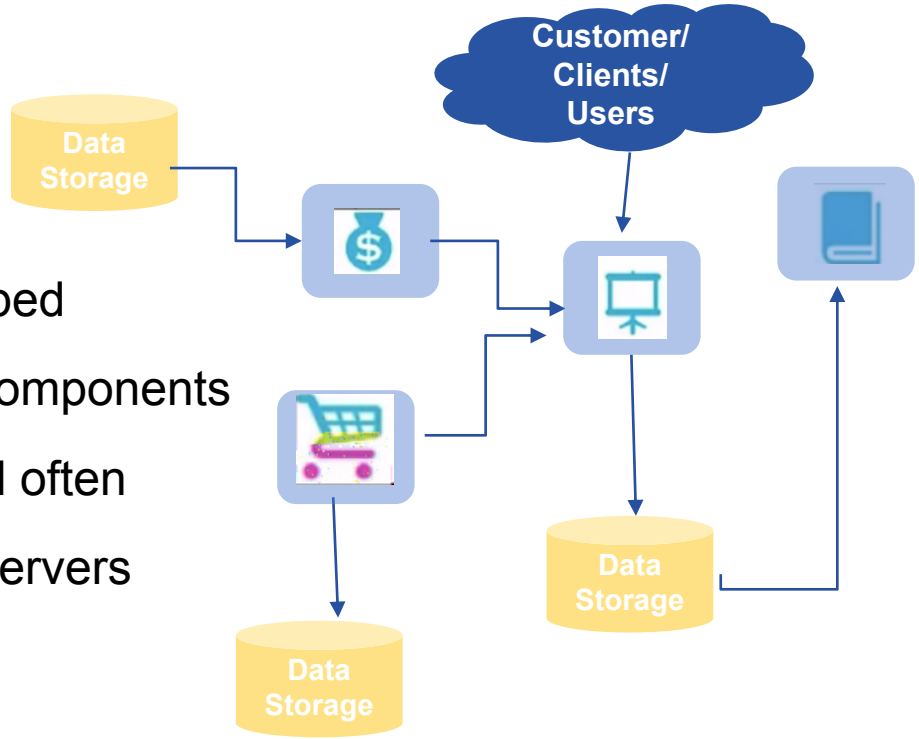
- Very difficult to maintain
- One component failure will cause the whole system to fail.
- Very difficult to understand and create the patches for monolithic applications.
- Adapting to new technology is very challengeable.
- Take a long time to startup because all the components need to get started.

Applications have changed dramatically

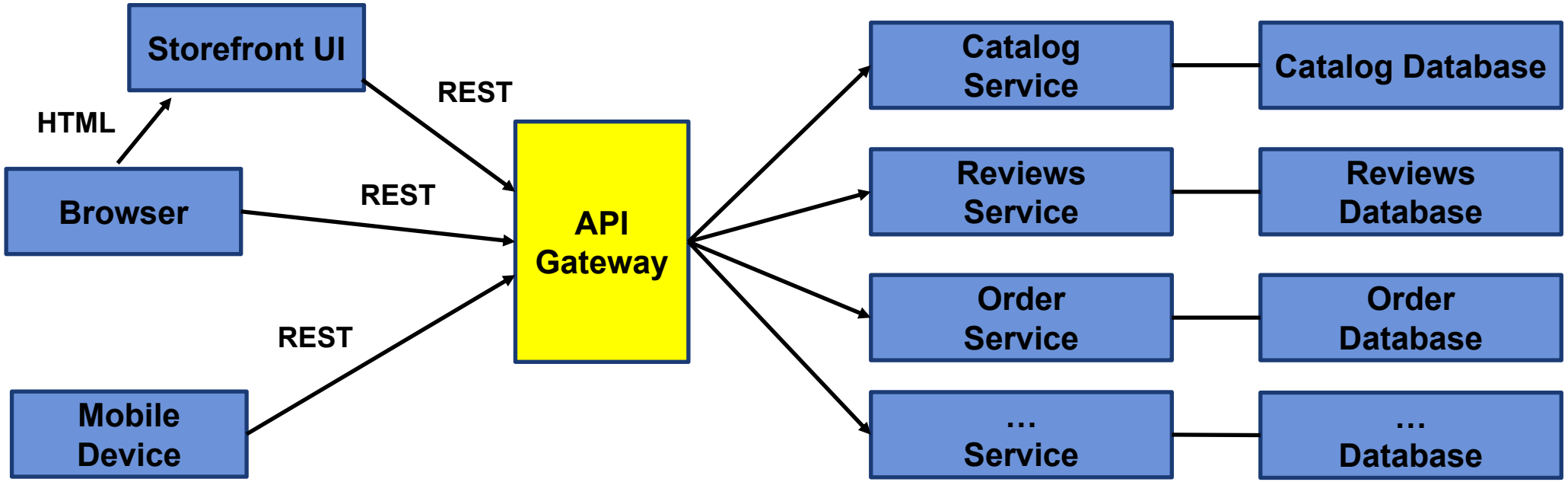


Today

- Apps are constantly developed
- Built from loosely coupled components
- Newer version are deployed often
- Deployed to a multitude of servers



Microservice Architecture

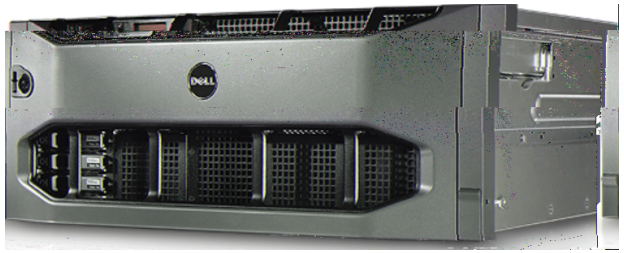


Benefits of Microservices

- Can scale independent microservices separately. No need to scale the whole the system
- Can use the latest technologies to develop the microservices.
- One component failure will not cause entire system downtimes.
- When developing an overall solution we can parallel the microservices development task with the small teams. So it helps to decrease the development time.

Once upon a time...a Software Stack

LAMP



Now much more distributed & complex...

Static website

nginx 1.5 + modsecurity + openssl + bootstrap 2

User DB

postgresql + pgv8 + v8

Analytics DB

hadoop + hive + thrift + OpenJDK

Queue

Redis + redis-sentinel

Background workers

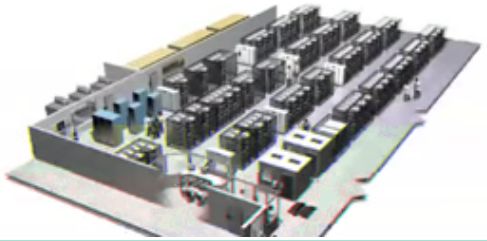
ython 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

Web frontend

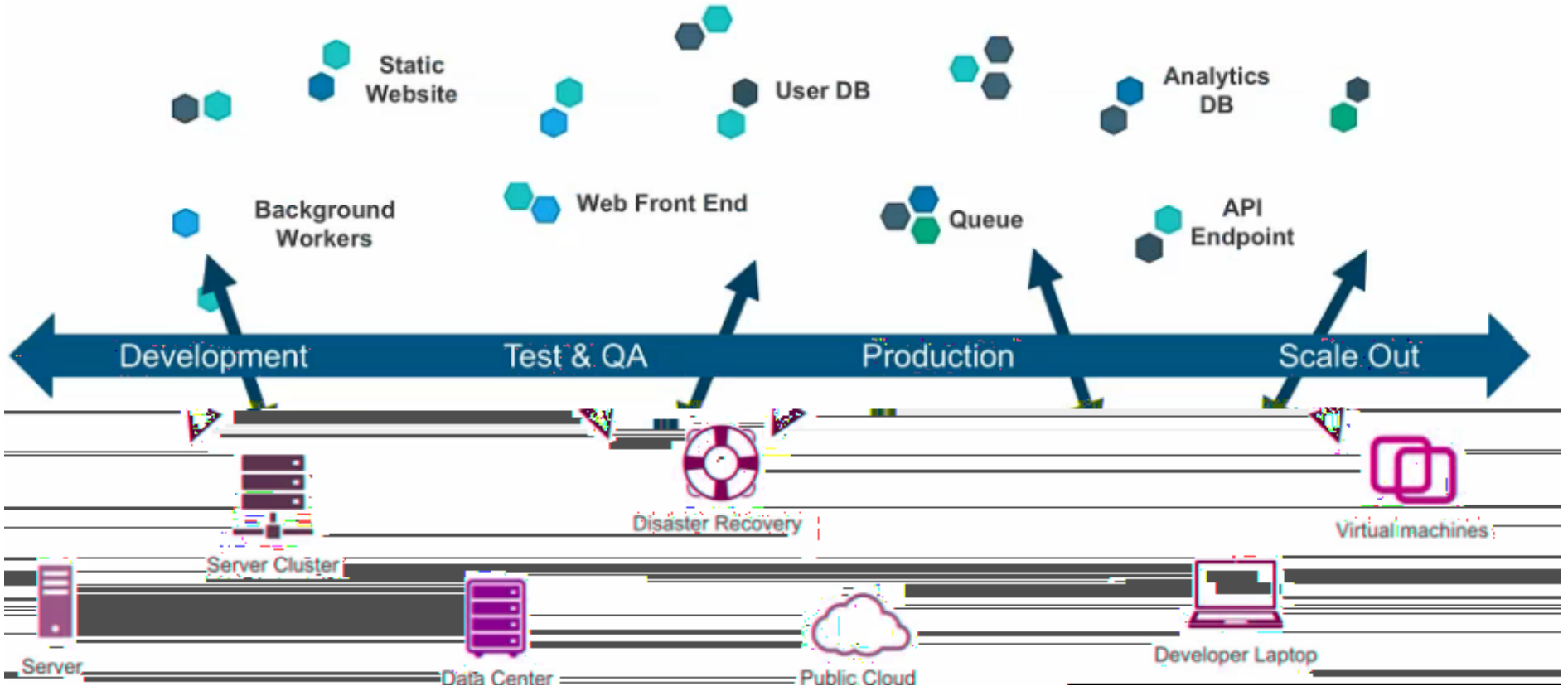
Ruby + Rails + sass + Unicorn

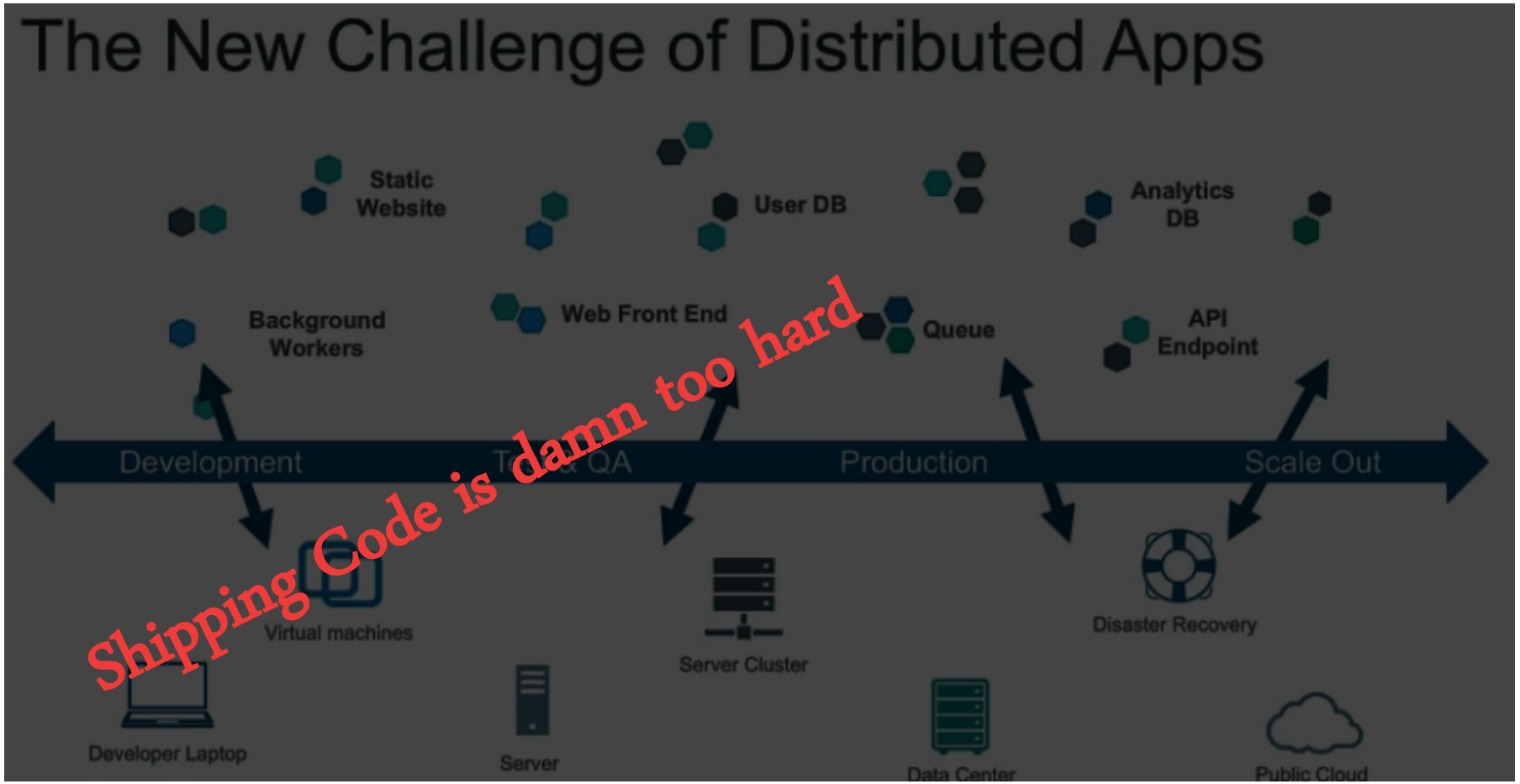
API endpoint

Python 2.7 + Flask + pyredis + celery + psycop + postgresql-client



The New Challenge of Distributed Apps



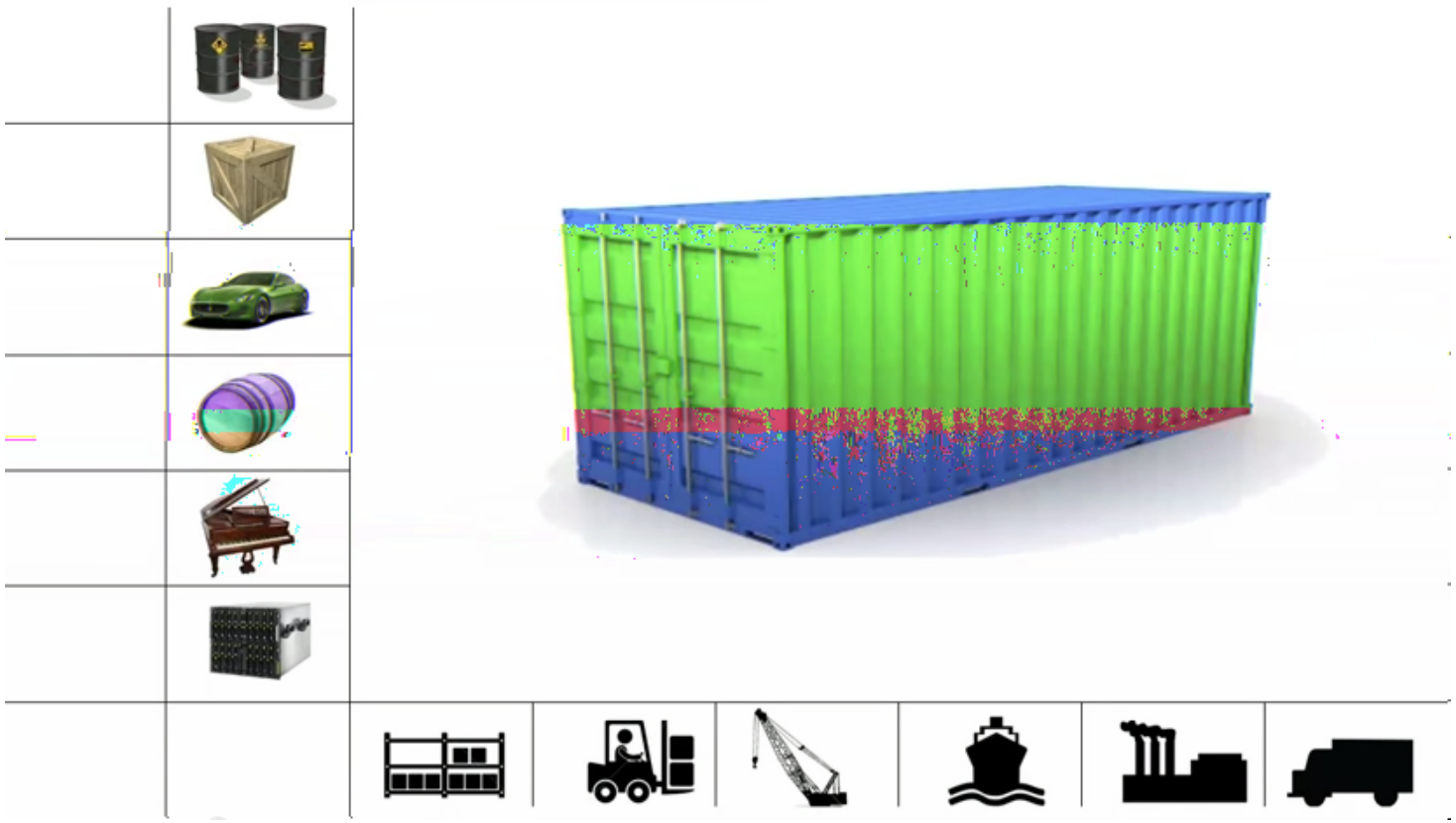

The New Challenge of Distributed Apps
Static Website
User DB
Analytics DB
Background Workers
Web Front End
Queue
API Endpoint
Development
Test & QA
Production
Scale Out
Shipping Code is damn too hard
Virtual machines
Server Cluster
Disaster Recovery
Developer Laptop
Server
Data Center
Public Cloud

An Effort to solve the problem Complexity

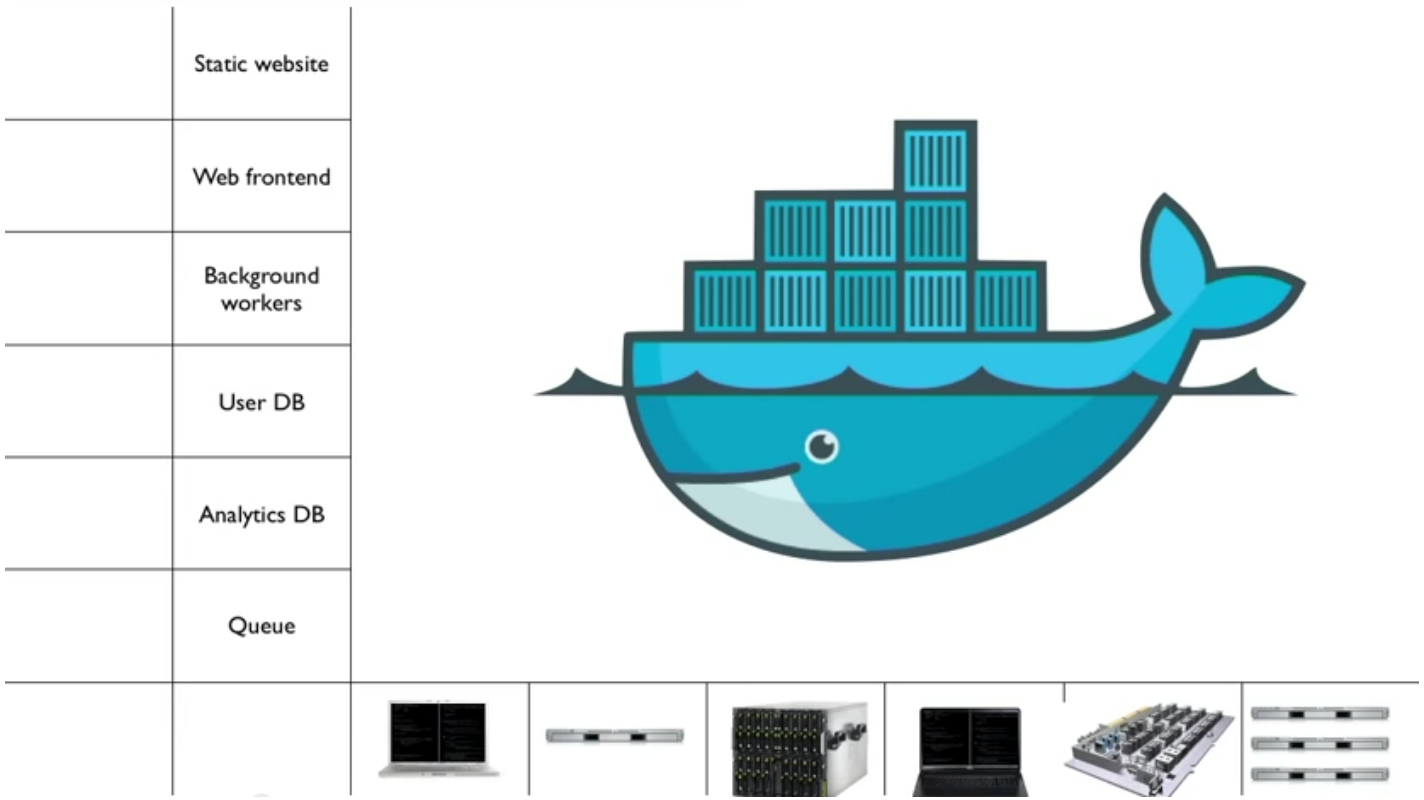
Every possible good to ship X Every Possible way to Ship

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

Enter....Internodal Container



That's what Docker is all about..



What is Docker and what problem does it solve?

What is Docker?

Refers to several things in 2019

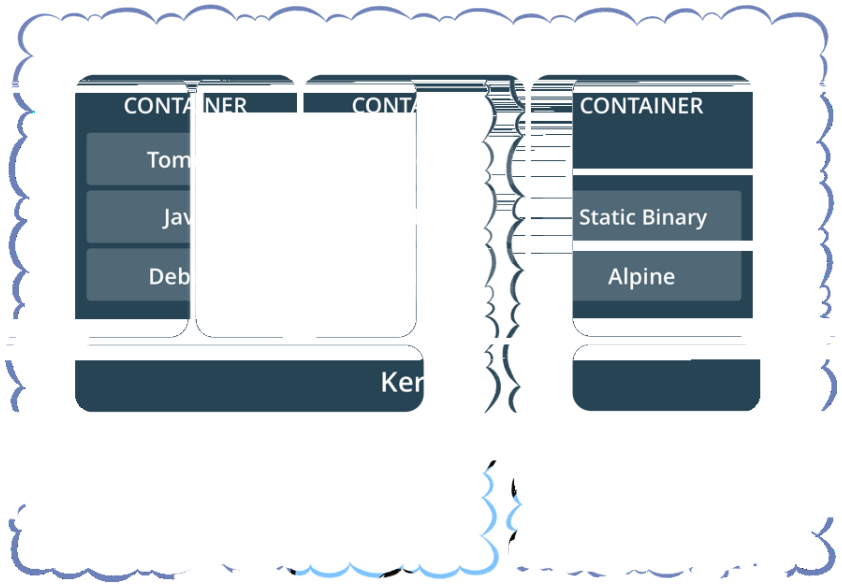
- Docker as a “Company”
- Docker as a “Product”
- Docker as a “Platform”
- Docker as a “CLI Tool”
- Docker as a “Computer Program”

docker

```
PS C:\Users\Ajeet_Raina> docker version
Client: Docker Engine - Community
 Version:           18.09.1
 API version:       1.39
 Go version:        go1.10.6
 Git commit:        4c52b90
 Built:             Wed Jan  9 19:34:26 2019
 OS/Arch:           windows/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.1
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.6
  Git commit:       4c52b90
  Built:            Wed Jan  9 19:41:49 2019
  OS/Arch:          linux/amd64
  Experimental:     true
```

What is Docker?



- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works for all major Linux distributions
- Containers native to Windows Server 2016 & 1809

Docker Product Offerings

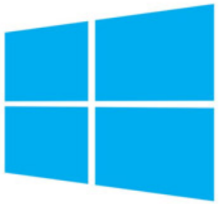
SaaS	On the Desktop	On Servers and In the Cloud		From the Ecosystem
	Docker for Mac	Docker Community Edition	Docker Enterprise Edition (EE) Docker Universal Control Plane Docker Trusted Registry Docker Engine (aka EE Basic)	Docker Certified Certified Containers Certified Plugins Certified Infrastructure
Docker Hub	Docker for Windows			

History of Docker

- A Company as well as Product – 6 Years Old company
- Developed by DotCloud Inc. (Currently Docker Inc.)
- A Framework they built their PaaS upon.
- Released it as open source 3 and 1/2 years back
- Cross-platform nature and friendliness towards System Admin and Developers
- Possible to set up in any OS, be it Windows, OSX, Linux, Solaris - It work the same way
- Guaranteed to run the same way - Your development desktop, a bare-metal server, virtual machine, data center, or cloud

Today Docker runs on

Windows 10



debian

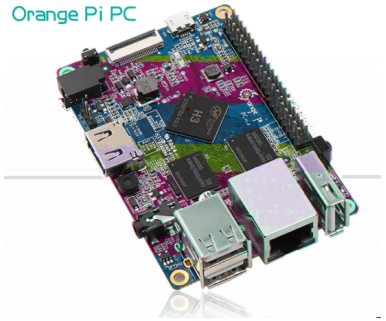
amazon web services

SUSE

ubuntu

orange pi

Orange Pi PC



Microsoft Windows Server 2019

Google Cloud Platform

Microsoft Azure

vmware

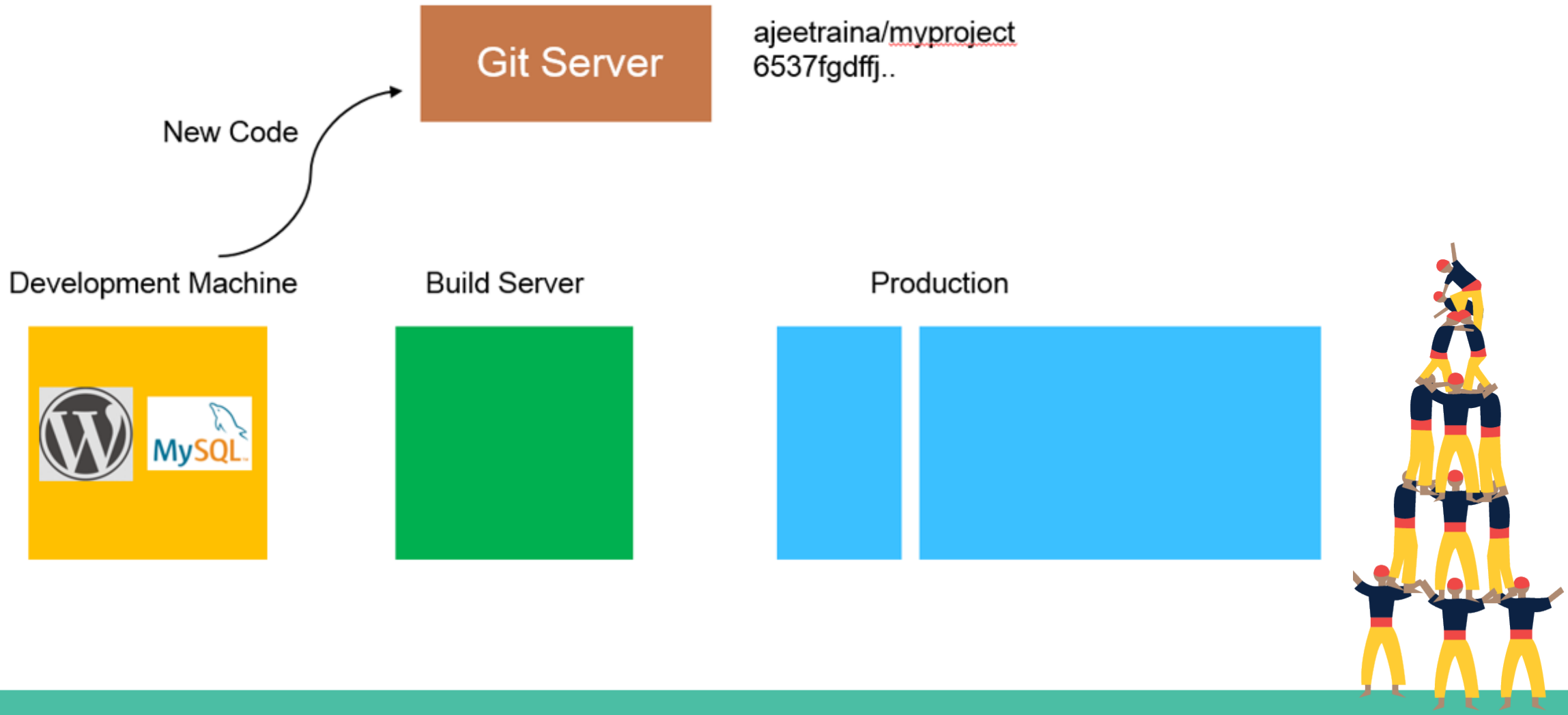
Traditional Software Development WorkFlow (without Docker)

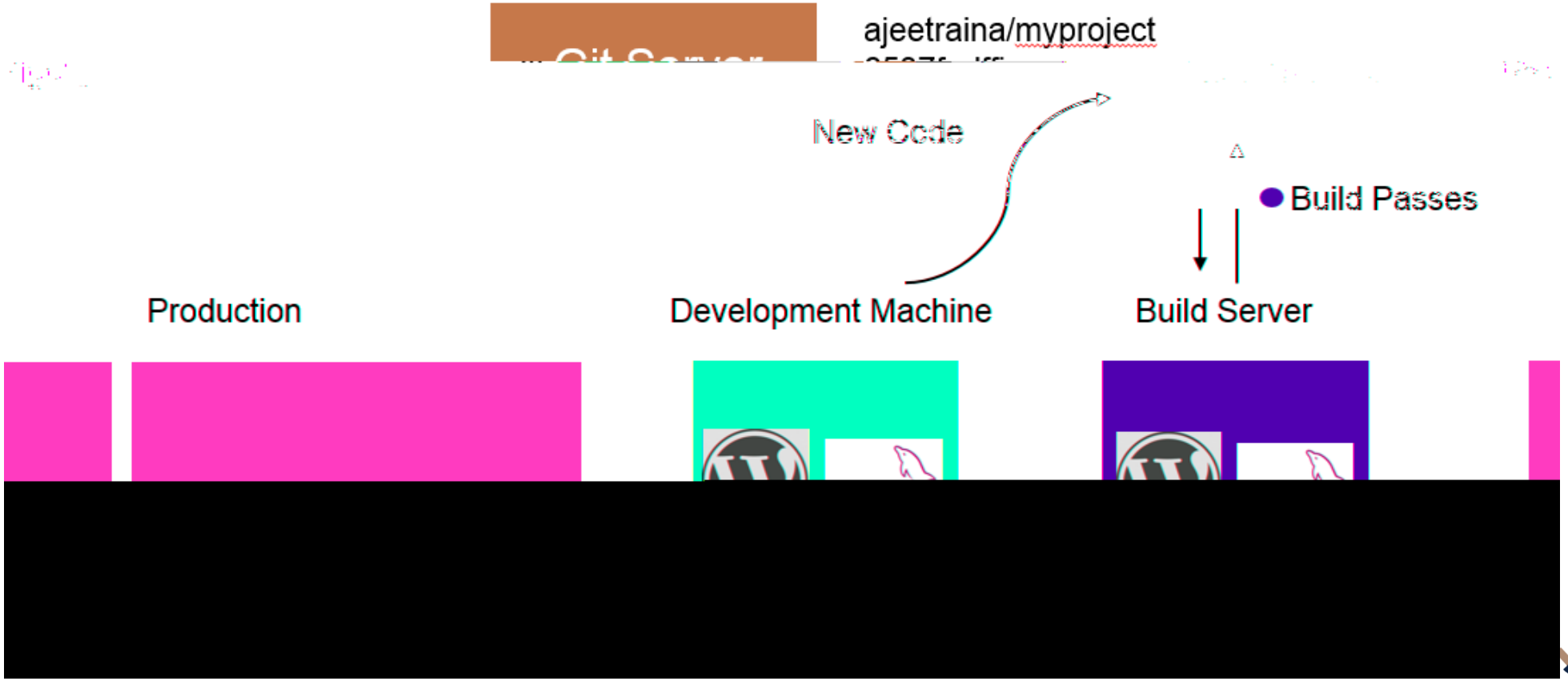
Git Server

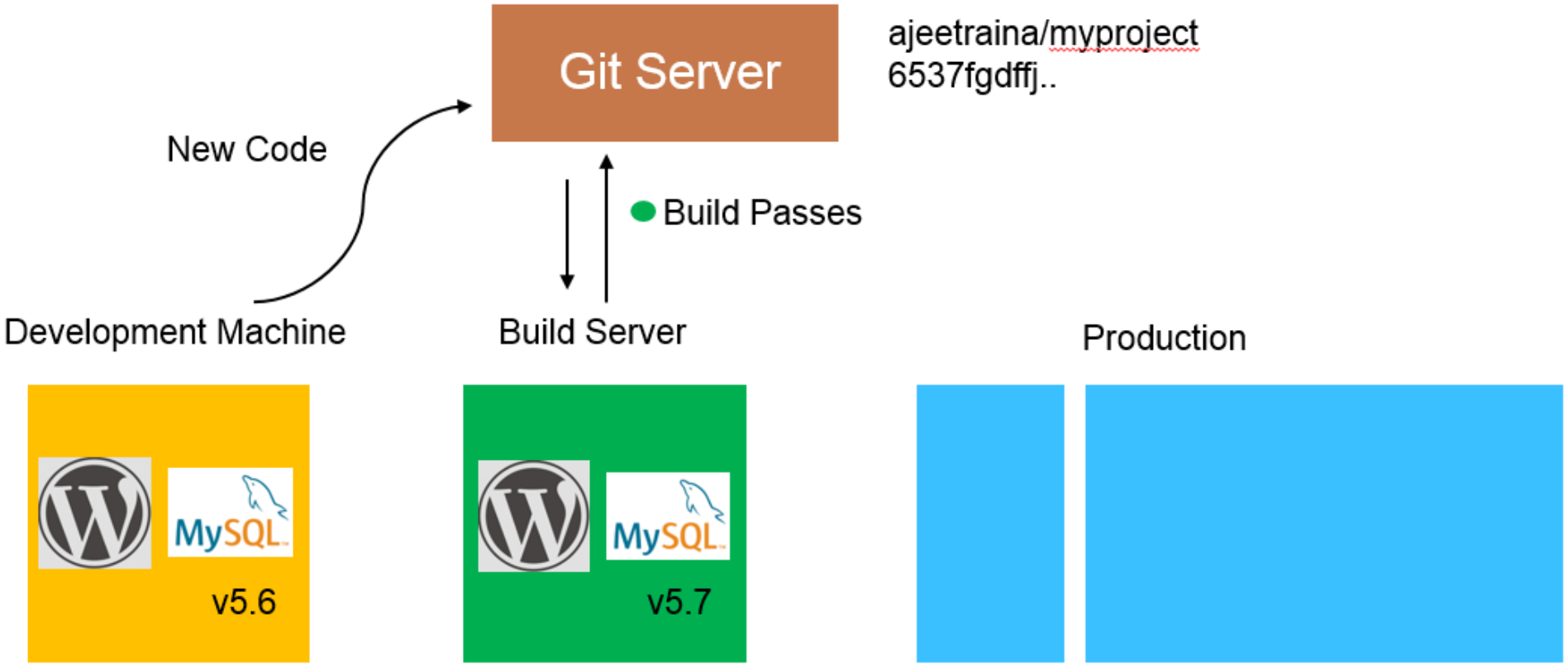
Development Machine

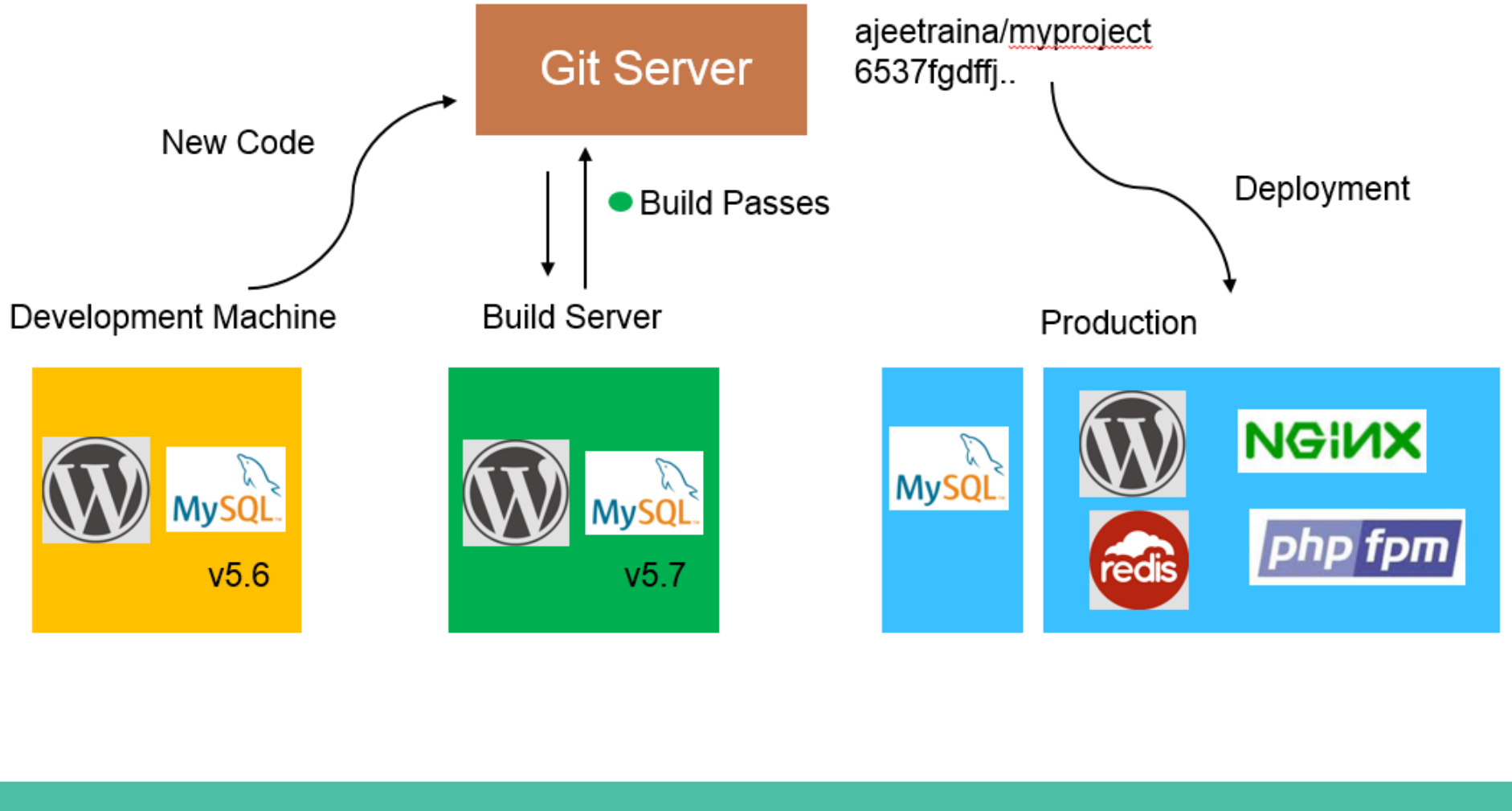
Build Server

Production


Git Server
ajeetraina/myproject
6537fgdffj..
New Code
Development Machine
Build Server
Production
MySQL


ajeetraina/myproject
New Code
Build Passes
Production
Development Machine
Build Server


Git Server
ajeetraina/myproject
6537fgdffj..
New Code
Build Passes
Development Machine
Build Server
Production
MySQL
v5.6
MySQL
v5.7


Git Server
ajeetraina/myproject
6537fgdffj..
New Code
Build Passes
Deployment
Development Machine
Build Server
Production
MySQL
v5.6
MySQL
v5.7
MySQL
NGINX
redis
php fpm

Traditional Software Development WorkFlow (with Docker)

Git Server

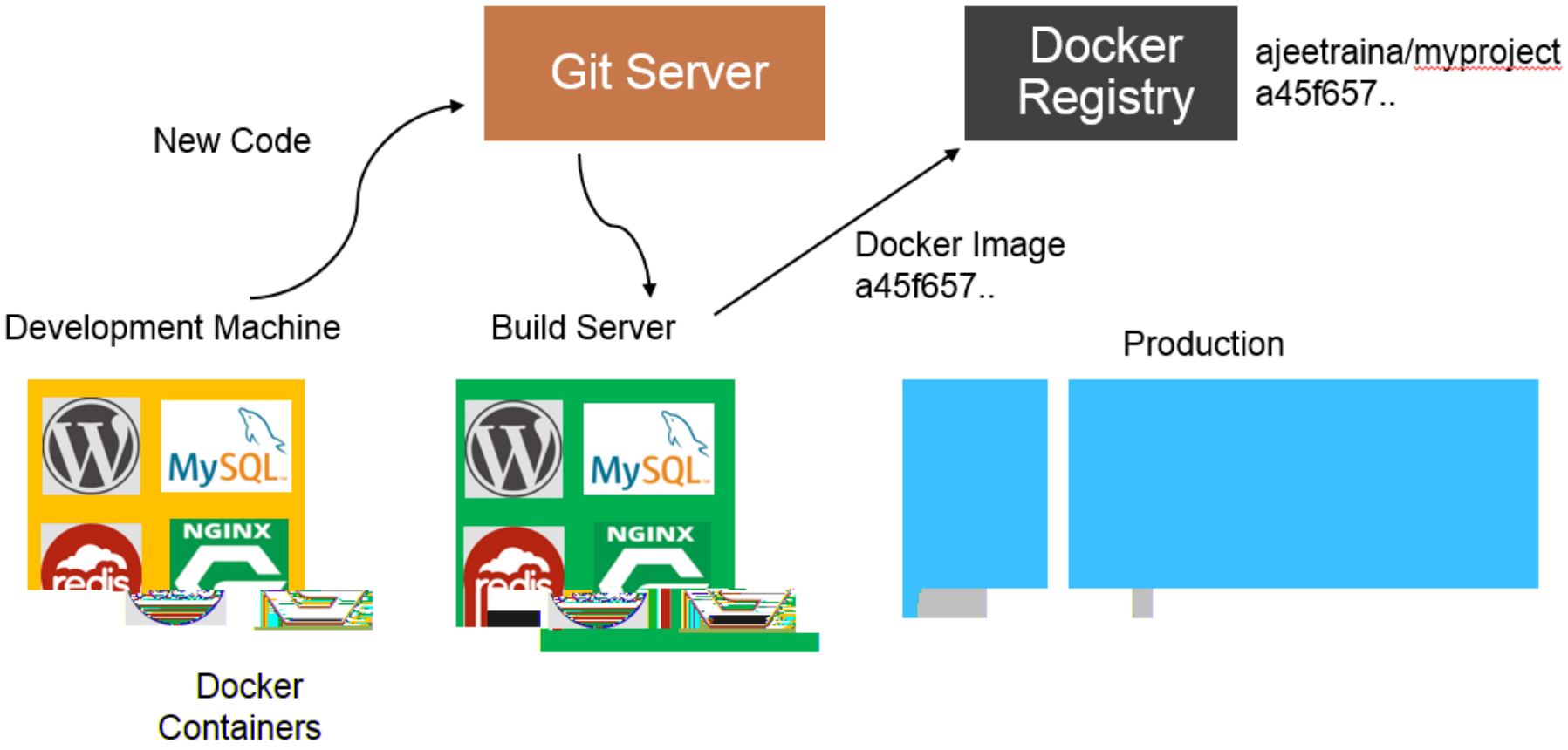
Docker Registry

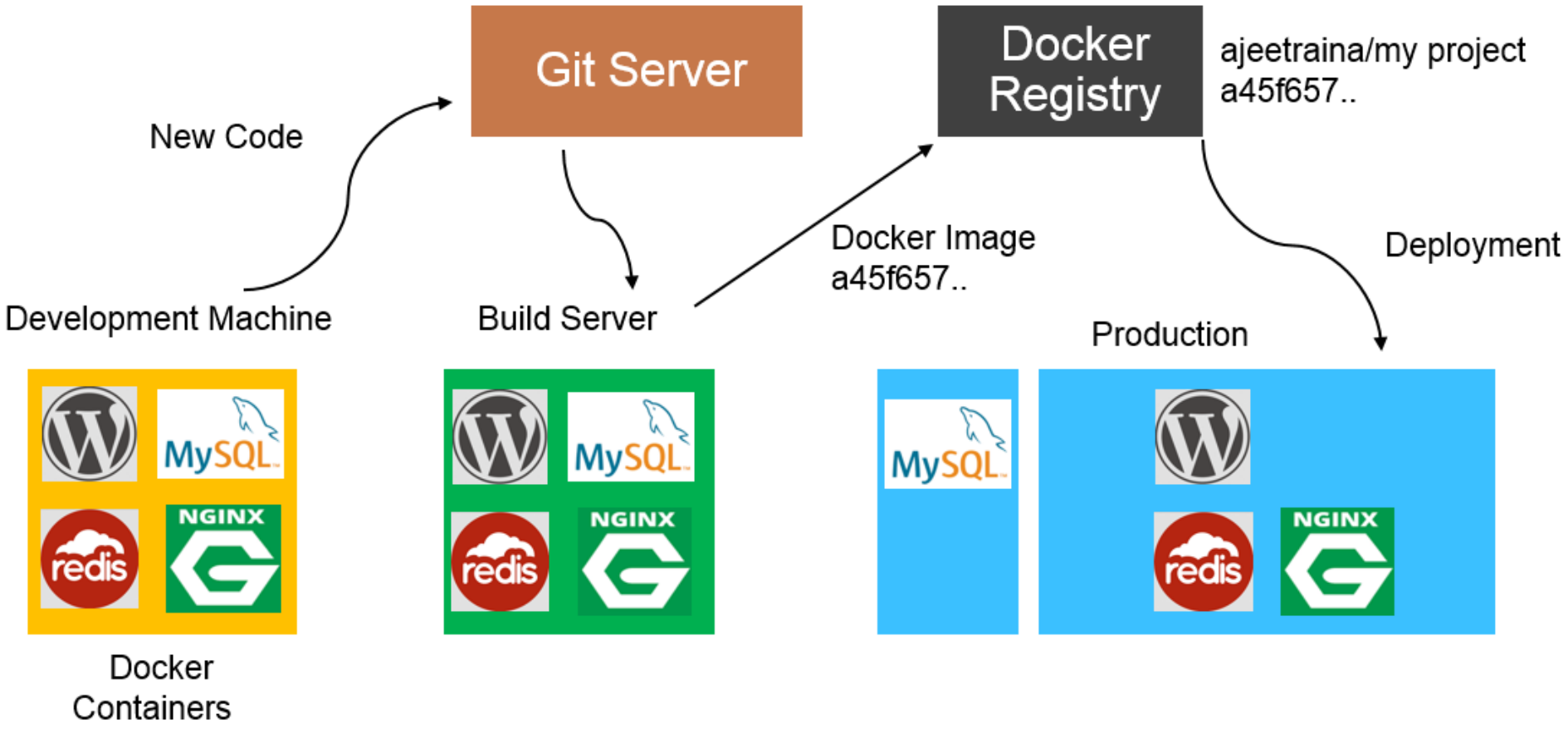
Development Machine

Docker Containers

Build Server

Production


Git Server
Docker Registry
ajeetraina/myproject
a45f657..
New Code
Docker Image
a45f657..
Development Machine
Build Server
Production
MySQL
NGINX
redis
Docker
Containers


Git Server
Docker Registry
ajeetraina/my project
a45f657..
New Code
Development Machine
Build Server
Docker Image
a45f657..
Deployment
Production
MySQL
NGINX
redis
Docker
Containers

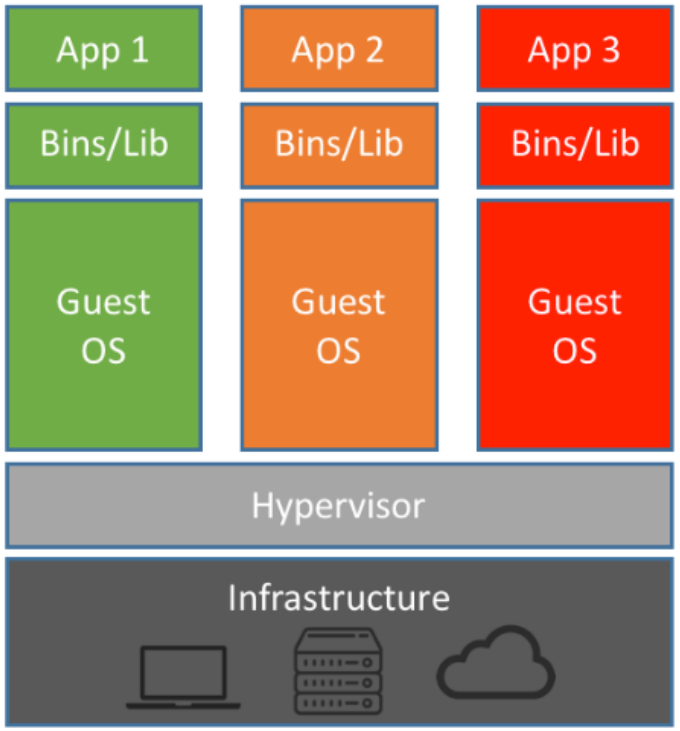
Docker Vs VM

Docker Containers are NOT VMs

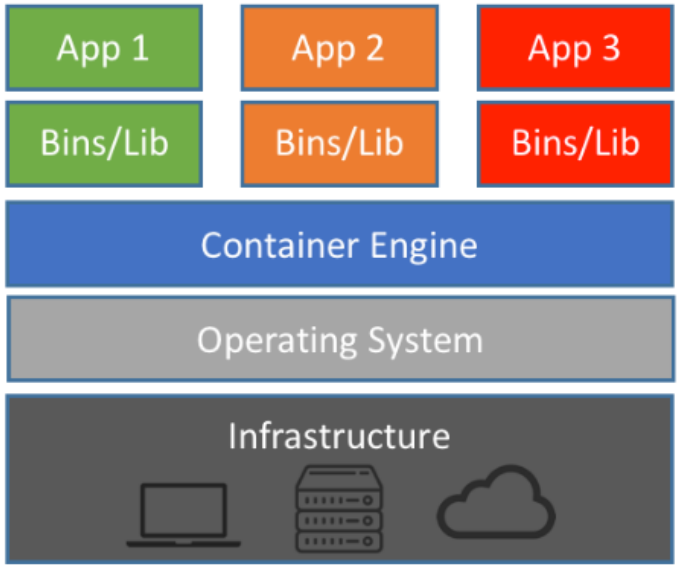
Virtual Machine

Containers

Docker Container Vs VM



Machine Virtualization



Containers

Comparing Docker & VM

Virtual Machines	Docker
Each VM runs its own OS	Container is just a user space of OS
Boot up time is in minutes	Containers instantiate in seconds
VMs snapshots are used sparingly	Images are built incrementally on top of another like layers. Lots of images/snapshots
Not effective diffs. Not version controlled	Images can be diffed and can be version controlled. Docker hub is like GITHUB
Cannot run more than couple of VMs on an average laptop	Can run many Docker containers in a laptop.
Only one VM can be started from one set of VMX and VMDK files	Multiple Docker containers can be started from one Docker image

Docker Vocabulary

Some Docker vocabulary

Containers	Images
How you **run** your application	How you **store** your application

Docker Image

The basis of a Docker container. Represents a full application

Docker Container

The standard unit in which the application service resides and executes

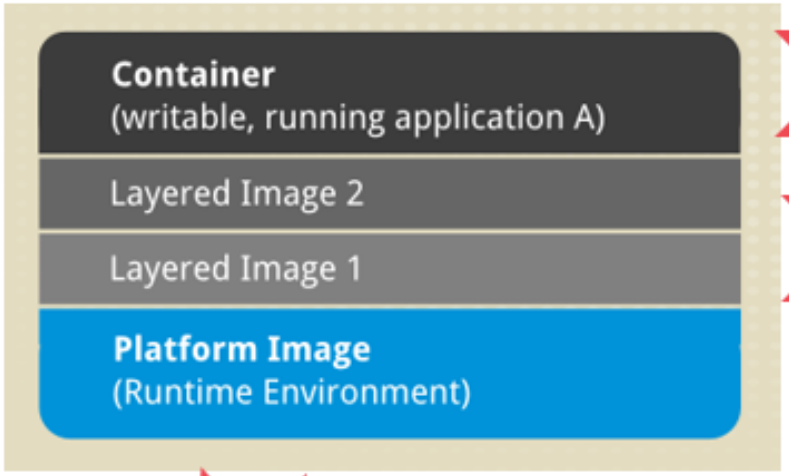
Docker Engine

Creates, ships and runs Docker containers deployable on a physical or virtual, host locally, in a datacenter or cloud service provider

Registry Service (Docker Hub or Docker Trusted Registry)

Cloud or server based storage and distribution service for your images

Image Layering

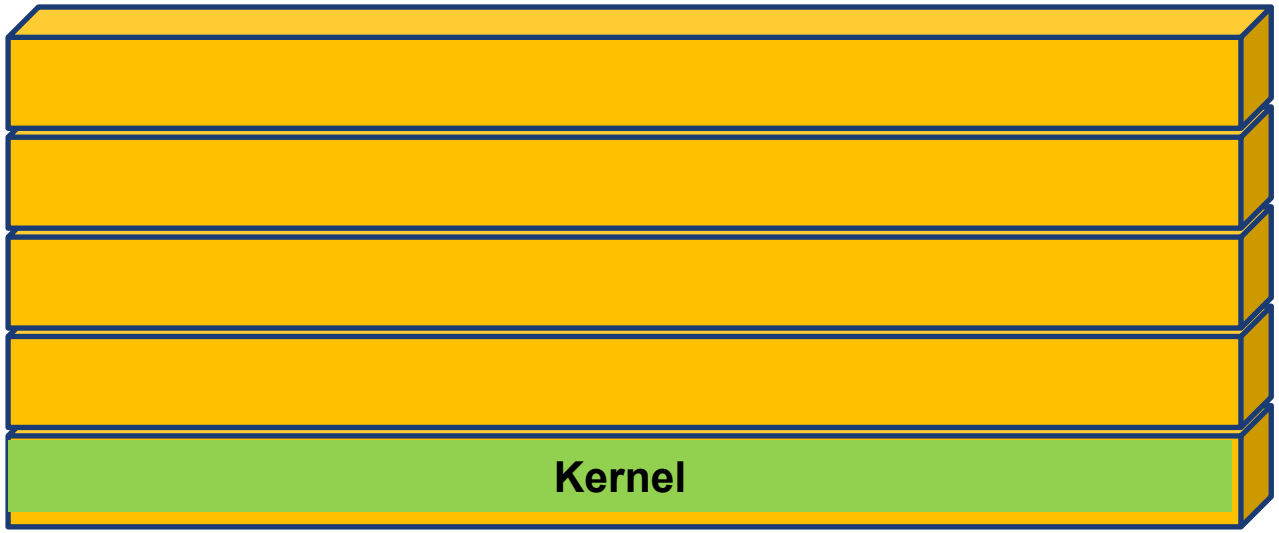


- An application sandbox.
- Each container is based on an image that holds necessary config data.
- When you launch a container from an image, a writable layer is added on top of this image

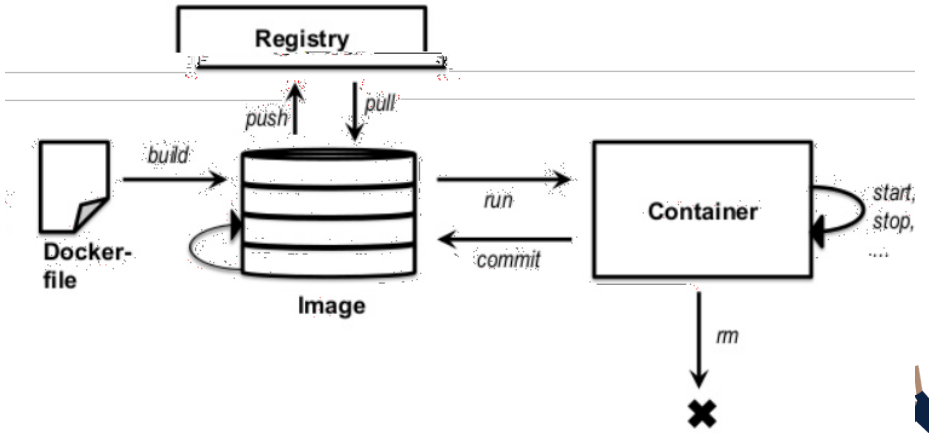
- A static snapshot of the containers' configuration.
- Image is a read-only layer that is never modified, all changes are made in top-most writable layer, and can be saved only by creating a new image.
- Each image depends on one or more parent images

- An image that has no parent.
- Platform images define the runtime environment, packages and utilities necessary for containerized application to run.

Image Layers



Basic Docker CLIs



Pulling Docker Image

$ docker pull ajeetraina/hellowhale

Listing out Docker Images

$ docker image ls

Running Docker Containers

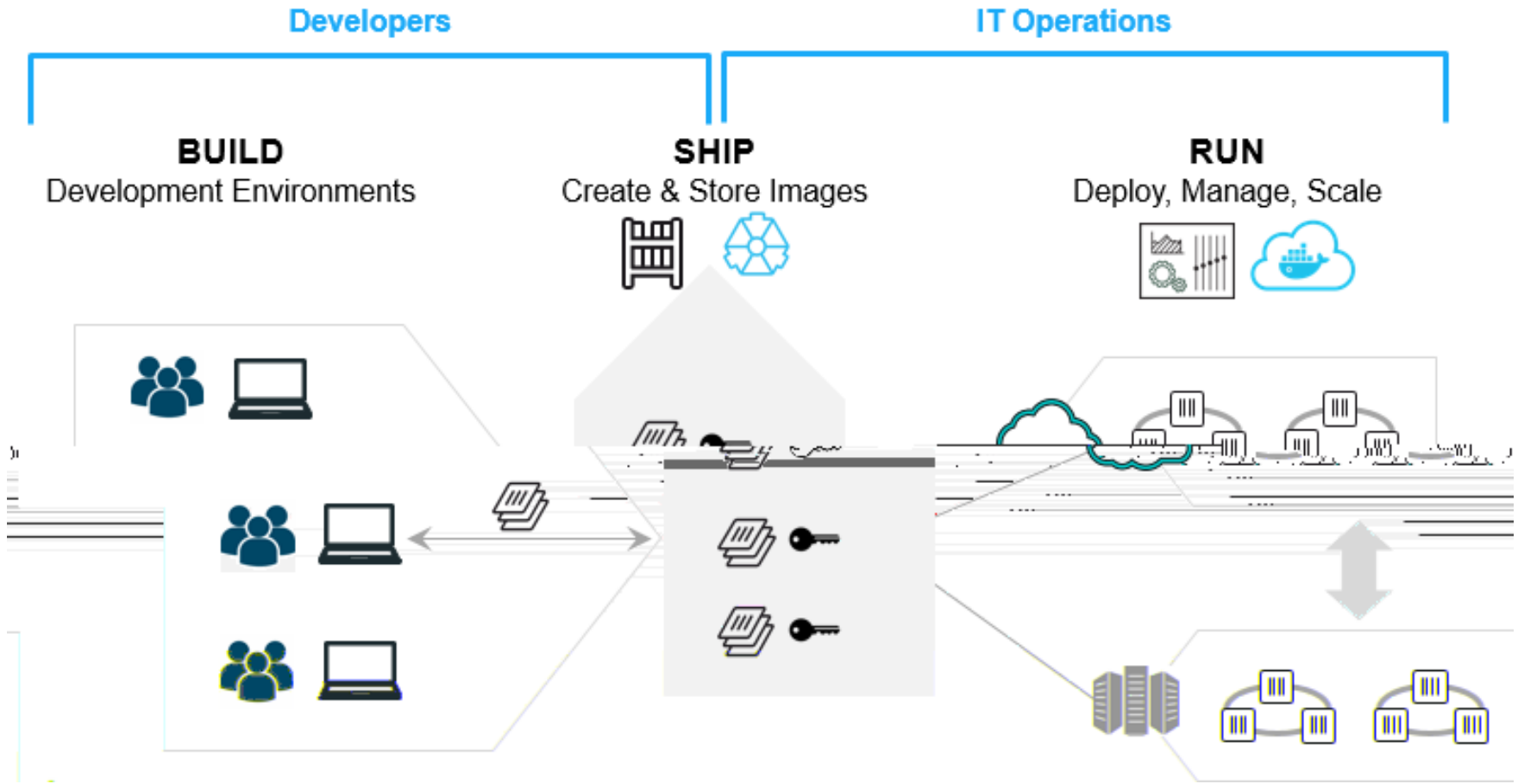
$ docker run –d –p 5000:5000 –-name hellowhale ajeetraina/hellowhale

Stopping the container

$ docker stop hellowhale (or <container id>)

Using Docker – Build, Ship & Run WorkFlow

Build. Ship. Run.



Demo

- Create DockerHub Account(if not completed)
- Open https://play-with-docker.com
- Hello Whale Example
- Build Your First Docker Image & Push it to DockerHub

Building Docker Containers & Microservices

Introducing Dockerfile

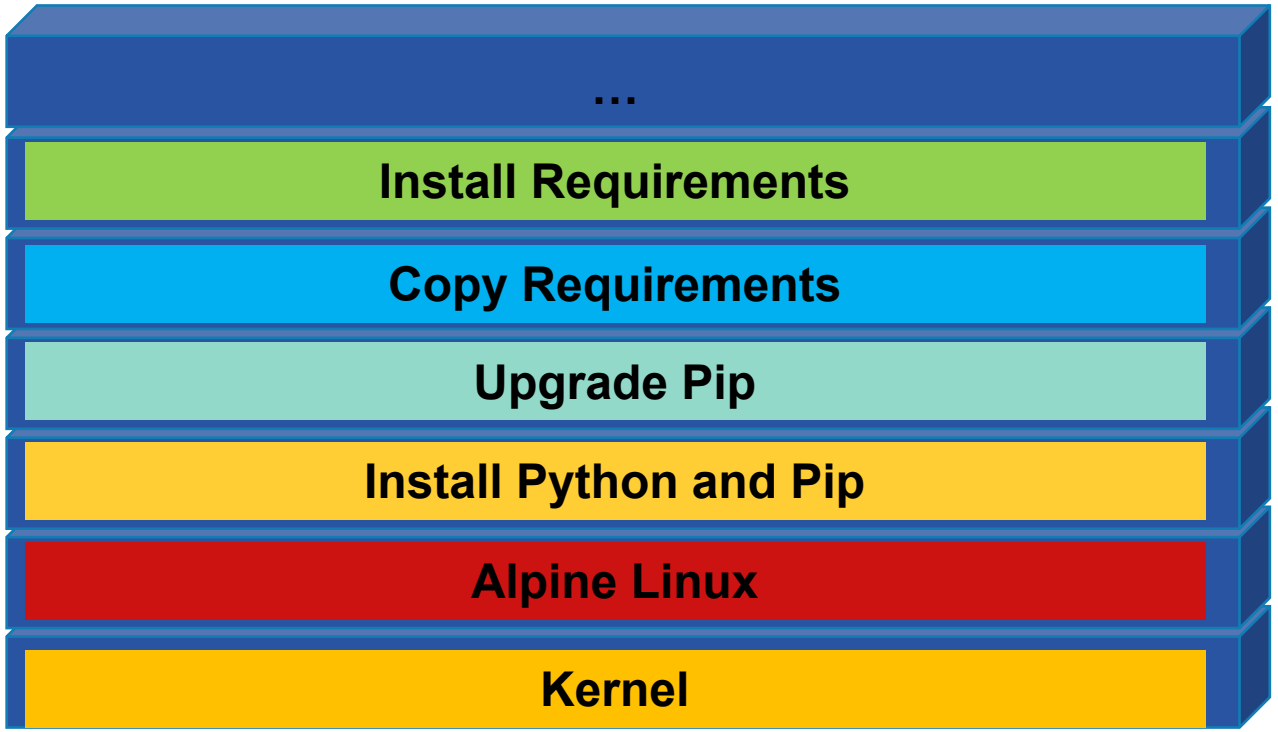
Series of instructions to build Docker Images

```
FROM alpine:3.6
LABEL maintainer ajeetraina@gmail.com
RUN apk update && \
    apk add git
```

Dockerfile – Example

```
1  # our base image
2  FROM alpine:latest
3
4  # Install python and pip
5  RUN apk add --update py-pip
6
7  # upgrade pip
8  RUN pip install --upgrade pip
9
10 # install Python modules needed by the Python app
11 COPY requirements.txt /usr/src/app/
12 RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt
13
14 # copy files required for the app to run
15 COPY app.py /usr/src/app/
16 COPY templates/index.html /usr/src/app/templates/
17
18 # tell the port number the container should expose
19 EXPOSE 5000
20
21 # run the application
22 CMD ["python", "/usr/src/app/app.py"]
```

Each Dockerfile creates a Layer



Docker Compose

Compose is a tool for defining and running multi-container Docker applications

```
version: '3'
services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress
   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:
```

Backend Service

Specify Volumes/Network

Environmental variables

Frontend Service

Specify Volumes/Network

Environmental variables

docker

docker con 18 EUROPE

Demo

- Running WordPress Application using Docker Compose

Workshop

https://collabnix.github.io/dockerlabs/

References

- https://github.com/collabnix/dockerlabs
- https://docs.docker.com
- https://collabnix.com


Thank You
dockercon18
EUROPE