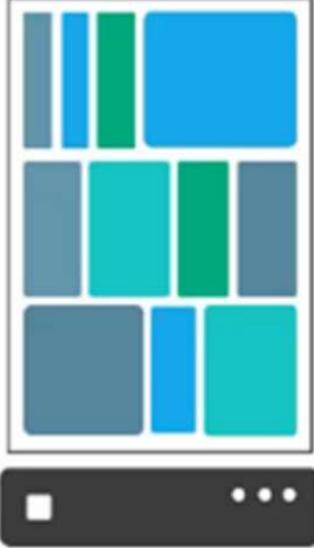




# A Shift from Monolithic to Microservice Architecture

# Applications have changed dramatically

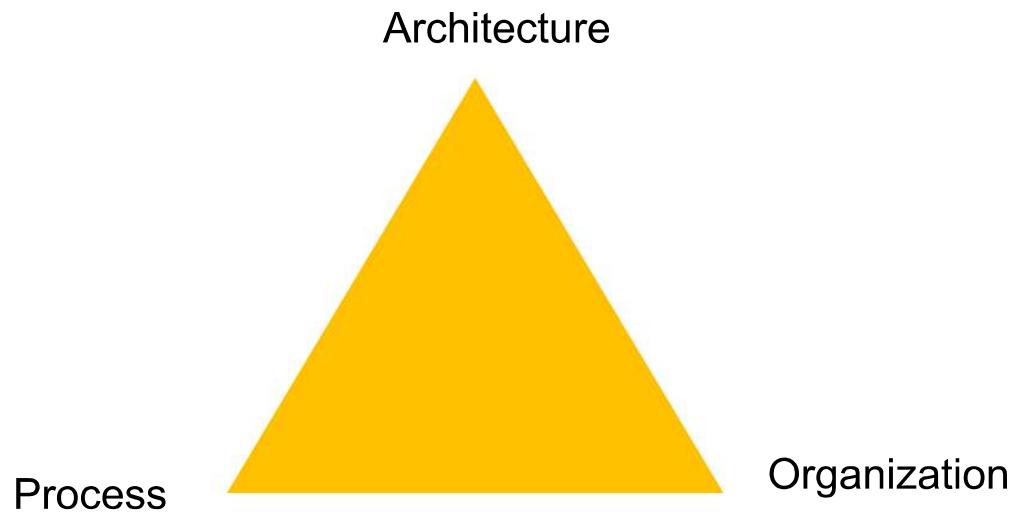
		~2000	Today
Monolithic			A Decade Ago (and still valid)
Slow Changing			<ul style="list-style-type: none"><li>- Apps were monolithic</li><li>- Built on a single stack such as .NET or Java</li><li>- Long Lived</li><li>- Deployed to a single server</li></ul>
Big Server			



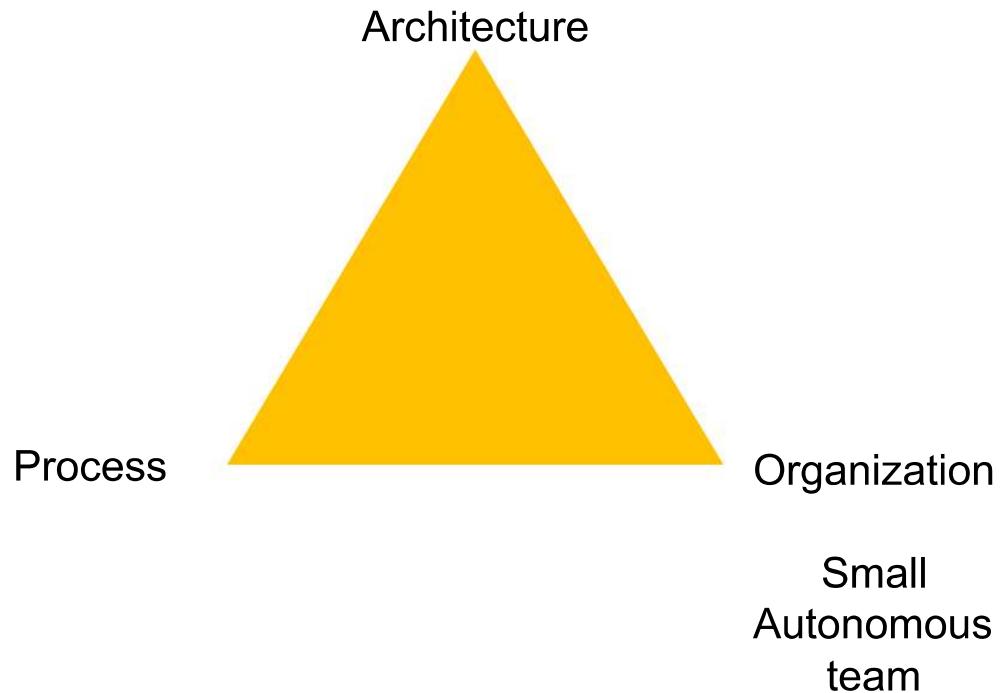
**Let's imagine you are building  
a large, complex application,  
e.g., Online Store**



# Successful Software Development

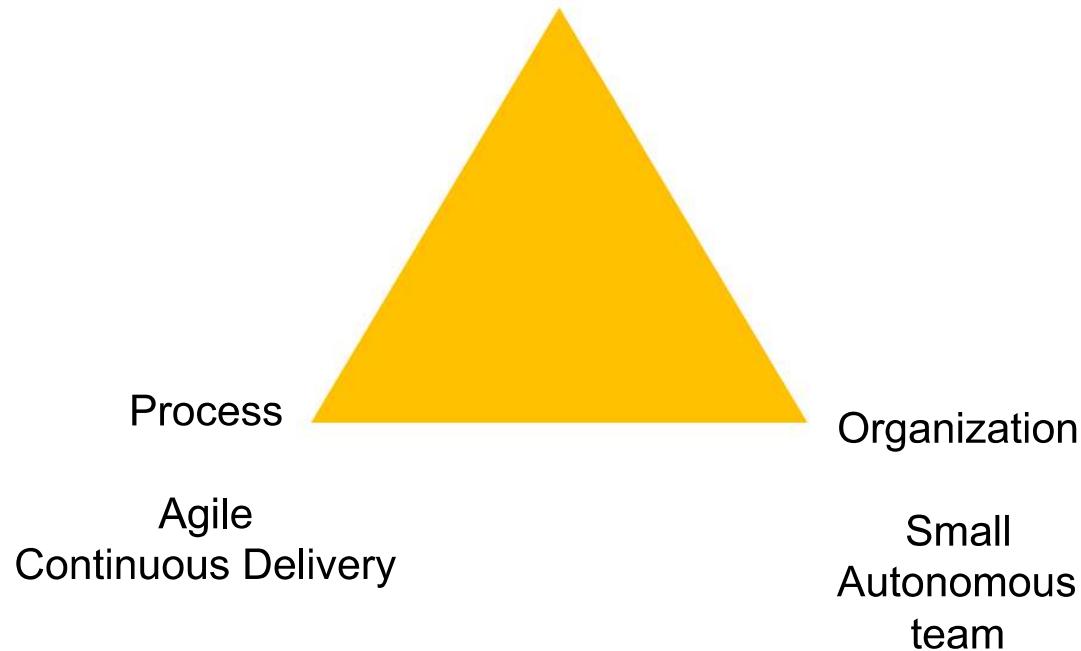


# Successful Software Development



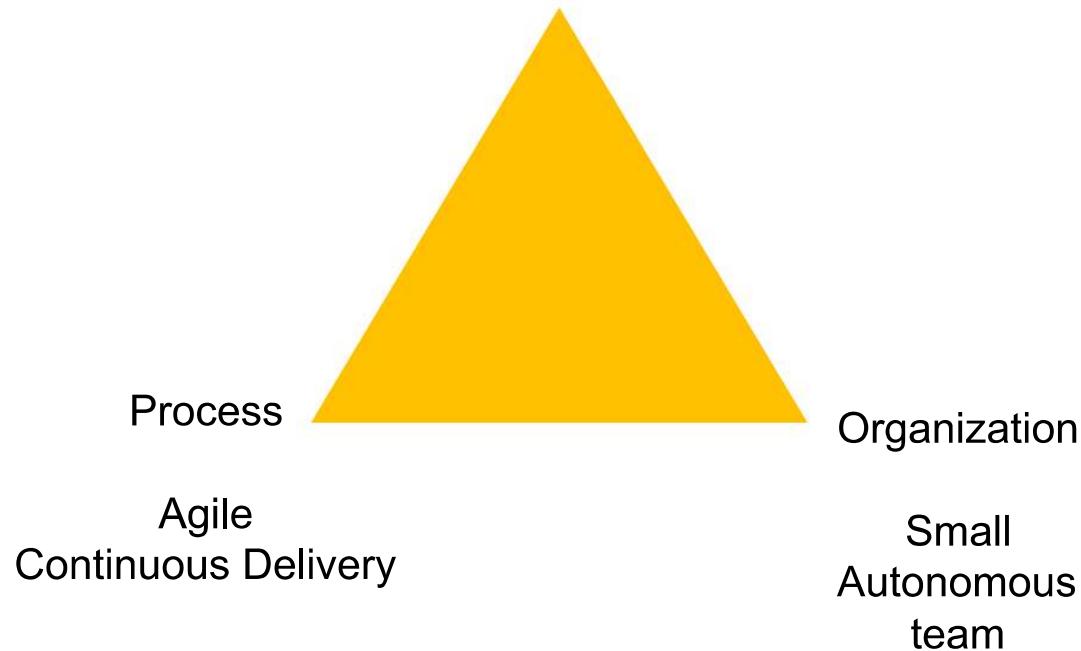
# Successful Software Development

Architecture

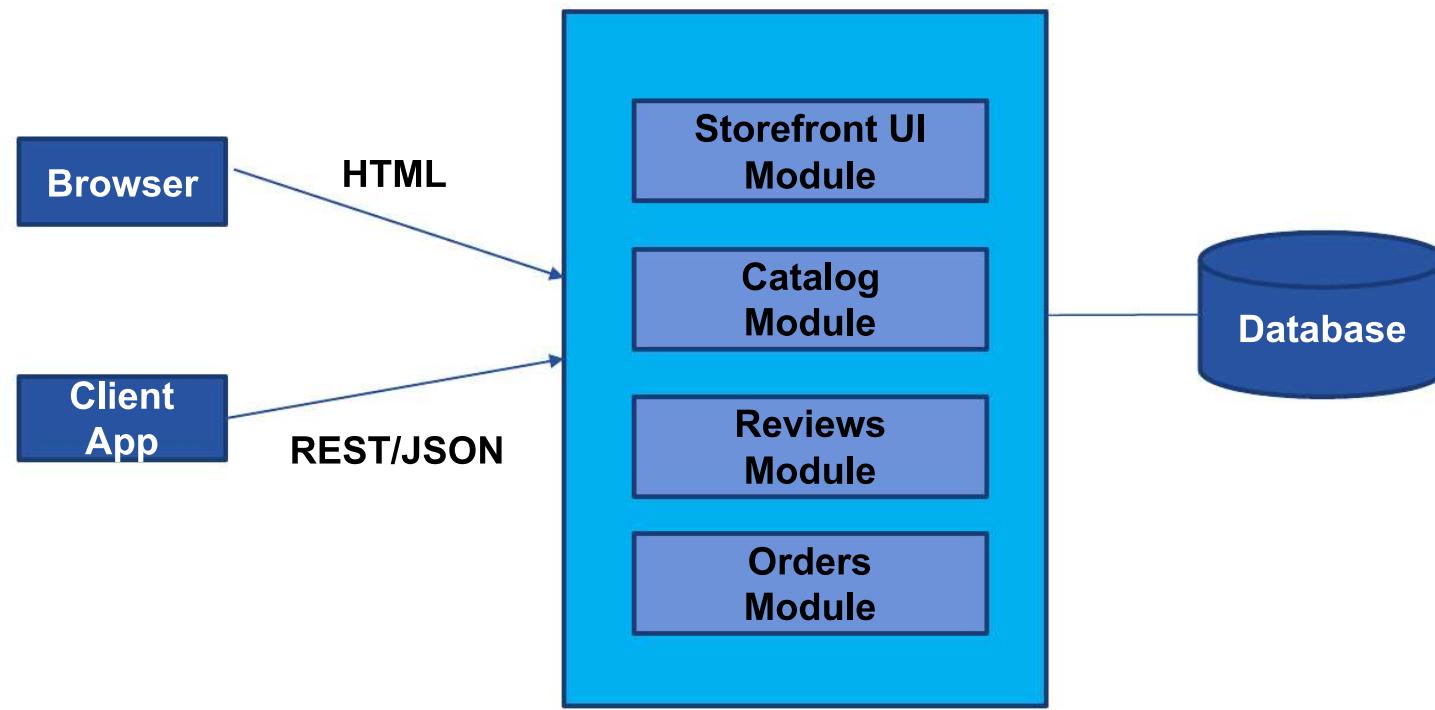


# Successful Software Development

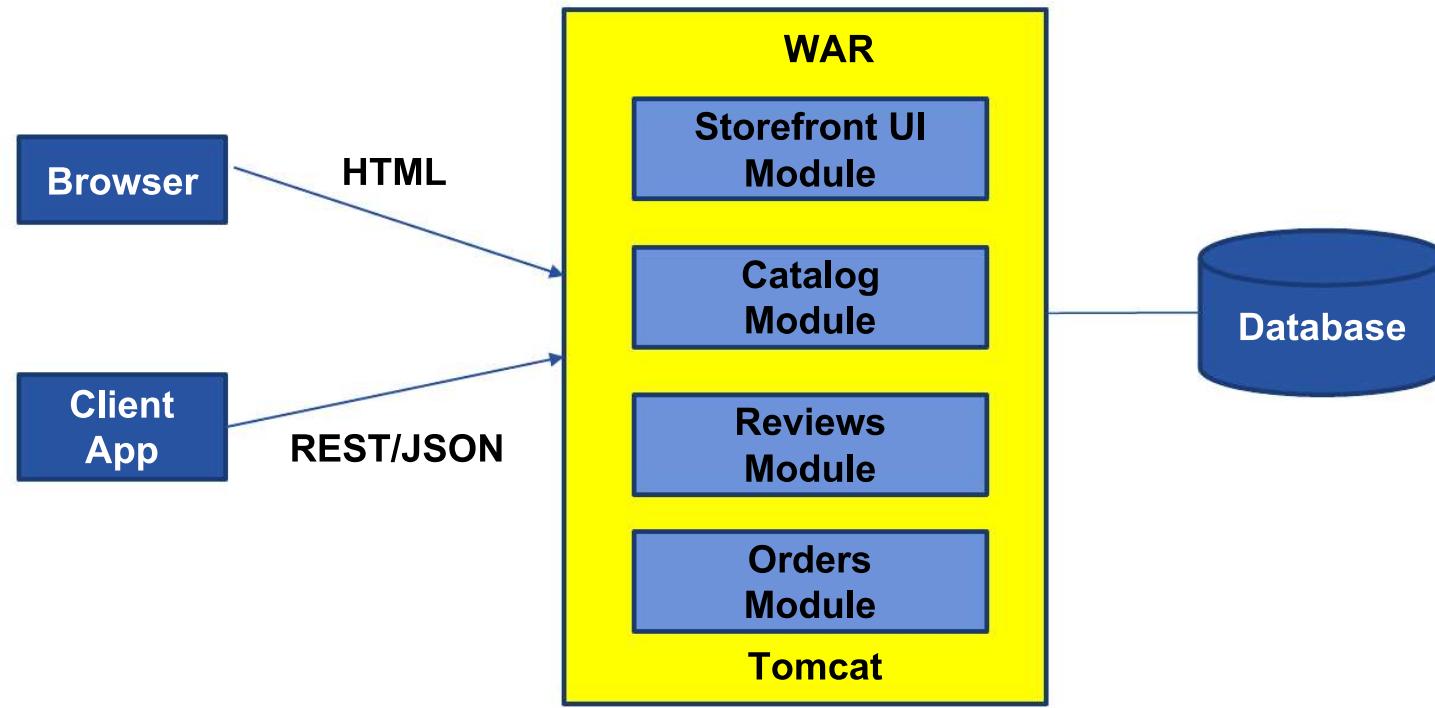
Architecture  
Monolithic Vs Microservices



# 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

....

@crichtardsor

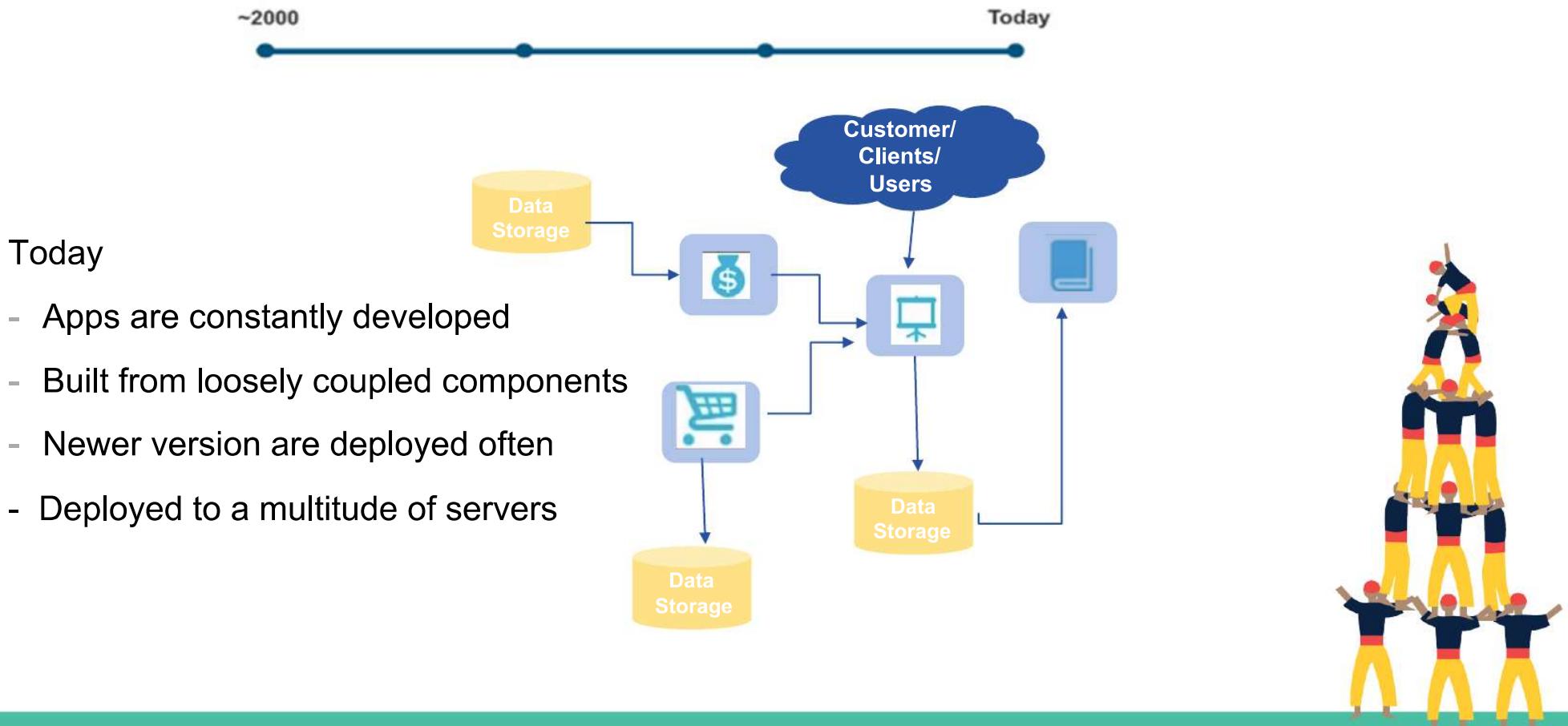


# Disadvantages of Monolith

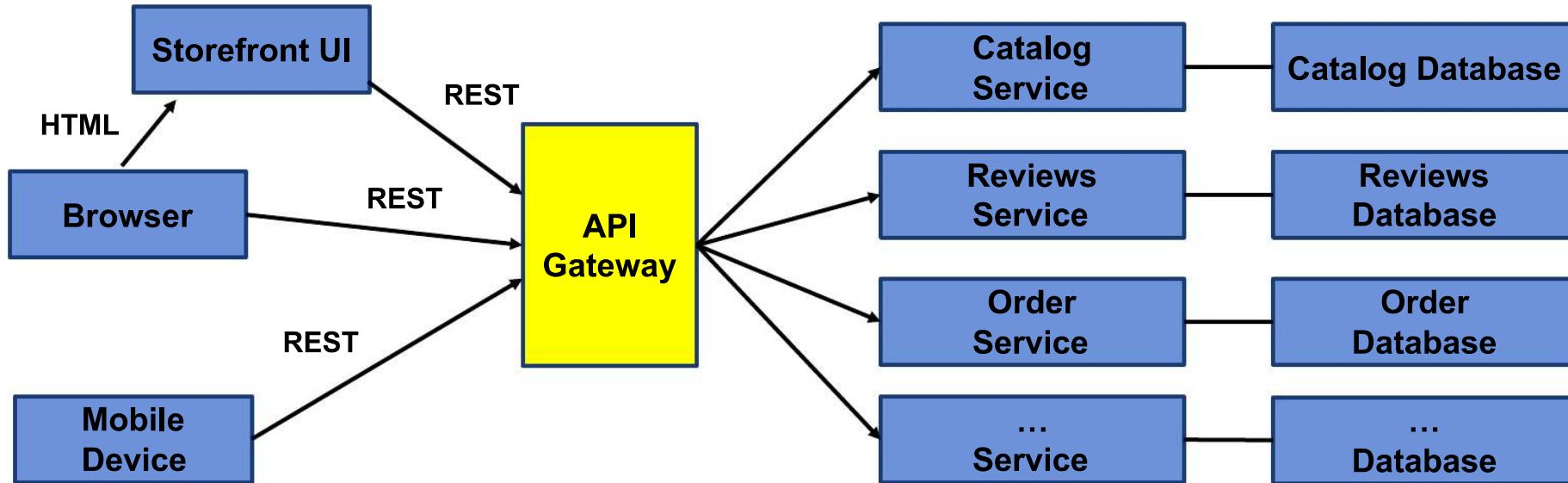
- Very difficult to maintain
- One component failure will cause the whole system to fail.
- Very difficult to understand and create the patches for monolithic applications.
- Adapting to new technology is very challengeable.
- Take a long time to startup because all the components need to get started.



# Applications have changed dramatically



# 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

## Background workers

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

## Web frontend

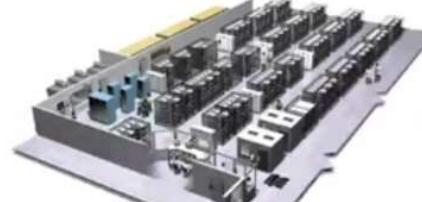
Ruby + Rails + sass + Unicorn

## Queue

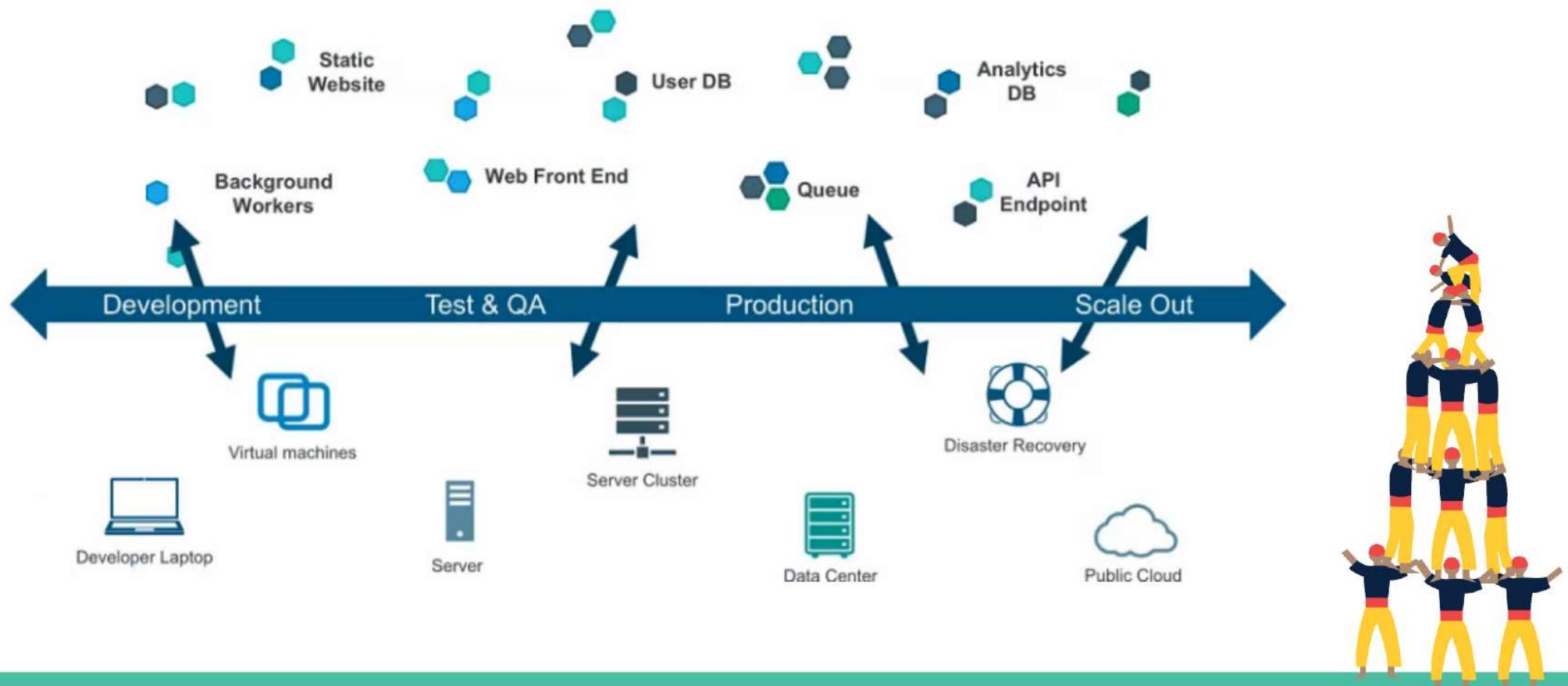
Redis + redis-sentinel

## API endpoint

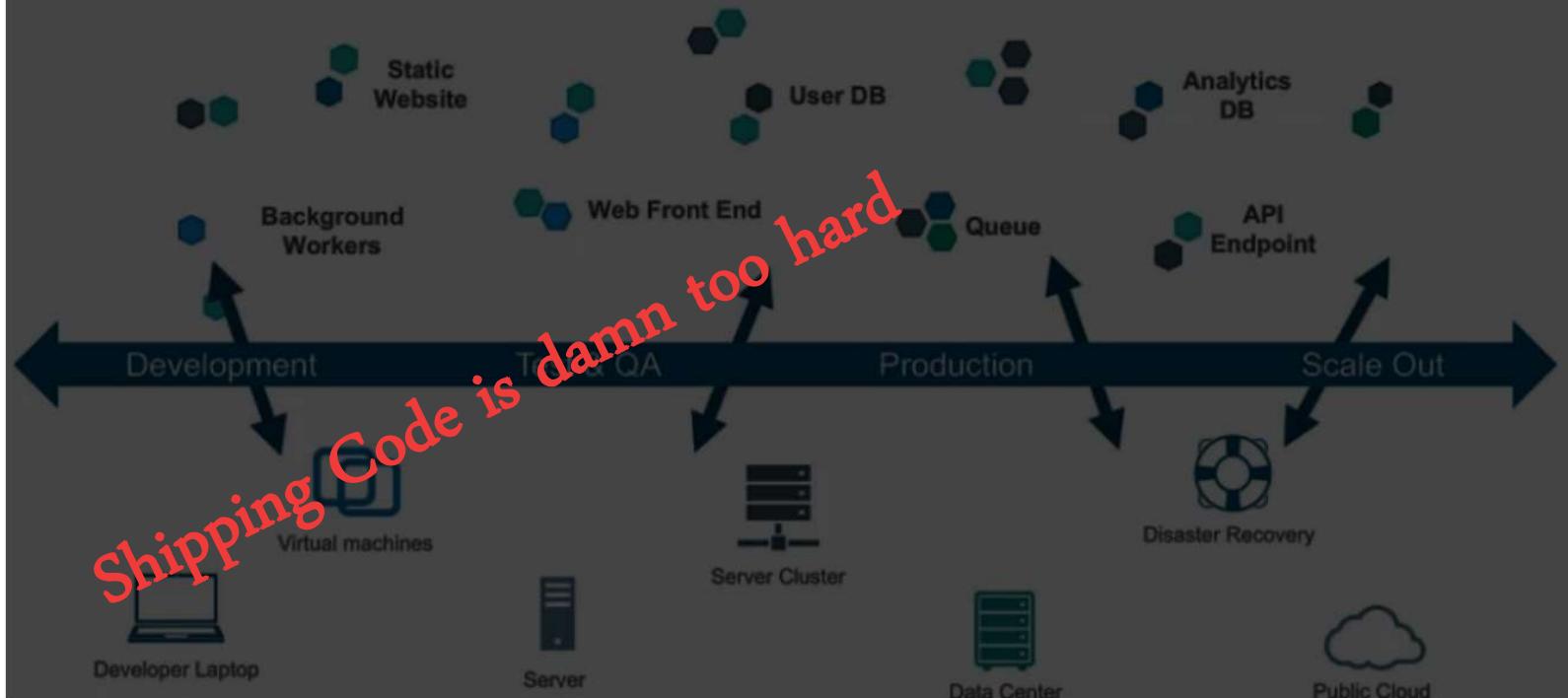
Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client



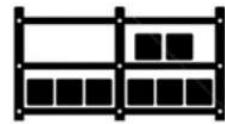
# The New Challenge of Distributed Apps



# The New Challenge of Distributed Apps



# An Effort to solve the problem Complexity



# Every possible good to ship X Every Possible way to Ship

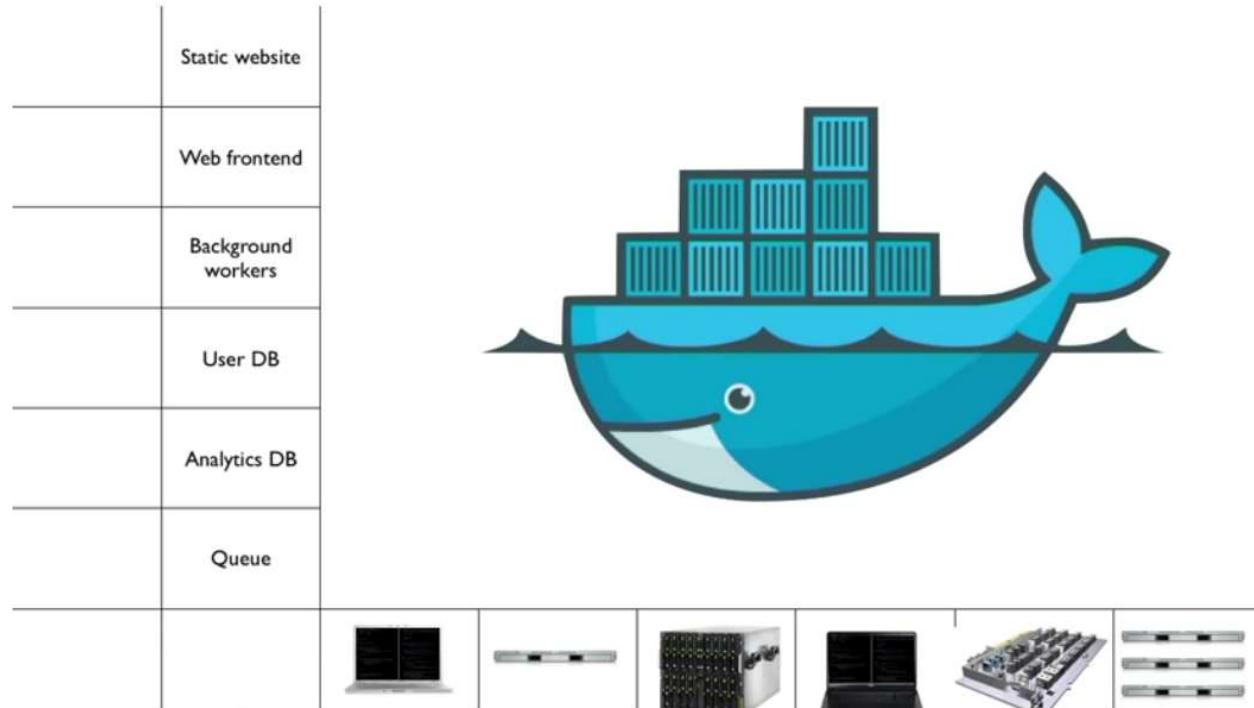
	?	?	?	?	?	?
	?	?	?	?	?	?
	?	?	?	?	?	?
	?	?	?	?	?	?
	?	?	?	?	?	?
	?	?	?	?	?	?



# Enter....Internodal Container



# That's what Docker is all about..



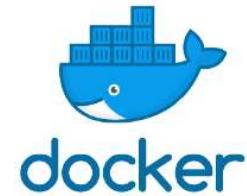


# **What is Docker and what problem does it solve?**

# 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”

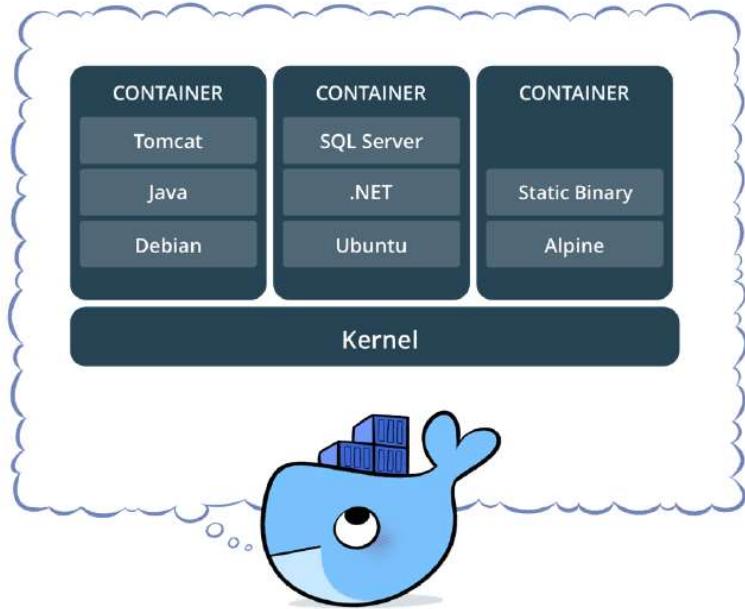


```
PS C:\Users\Ajeet_Raina> docker version
Client: Docker Engine - Community
 Version:           18.09.1
 API version:      1.39
 Go version:       go1.10.6
 Git commit:        4c52b90
 Built:             Wed Jan  9 19:34:26 2019
 OS/Arch:           windows/amd64
 Experimental:     false

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
```



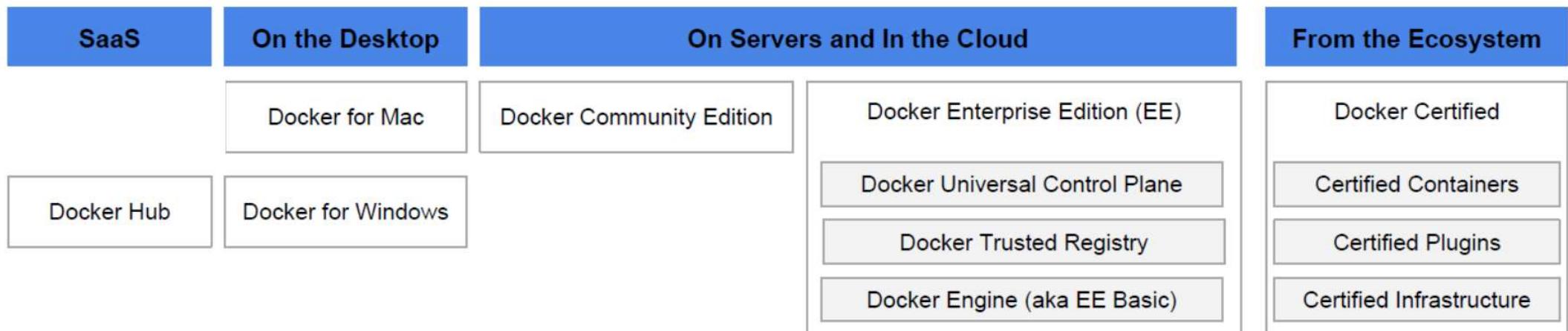
# What is Docker?

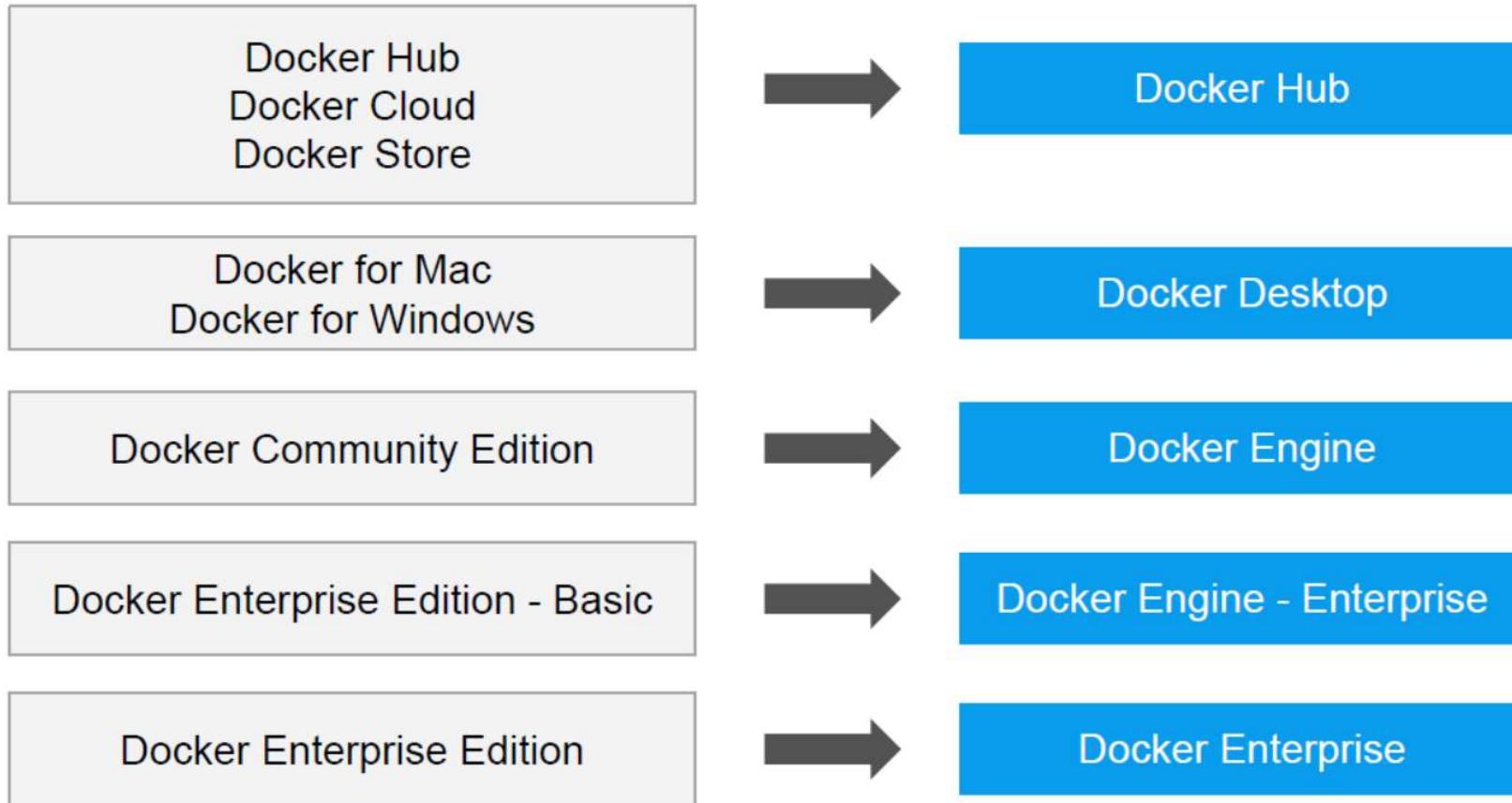


- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works for all major Linux distributions
- Containers native to Windows Server 2016 & 1809



# Docker Product Offerings





# Today Docker runs on





# **Traditional Software Development WorkFlow (without Docker)**

Git Server

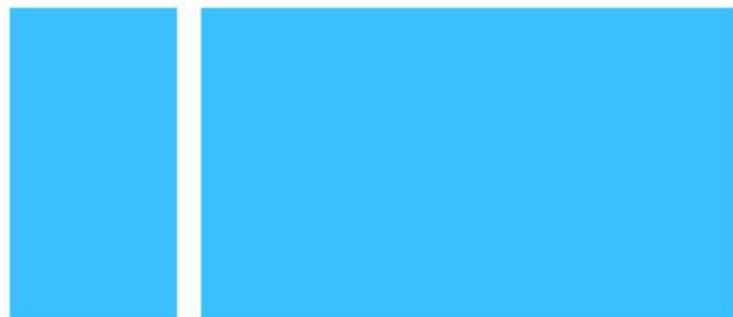
Development Machine

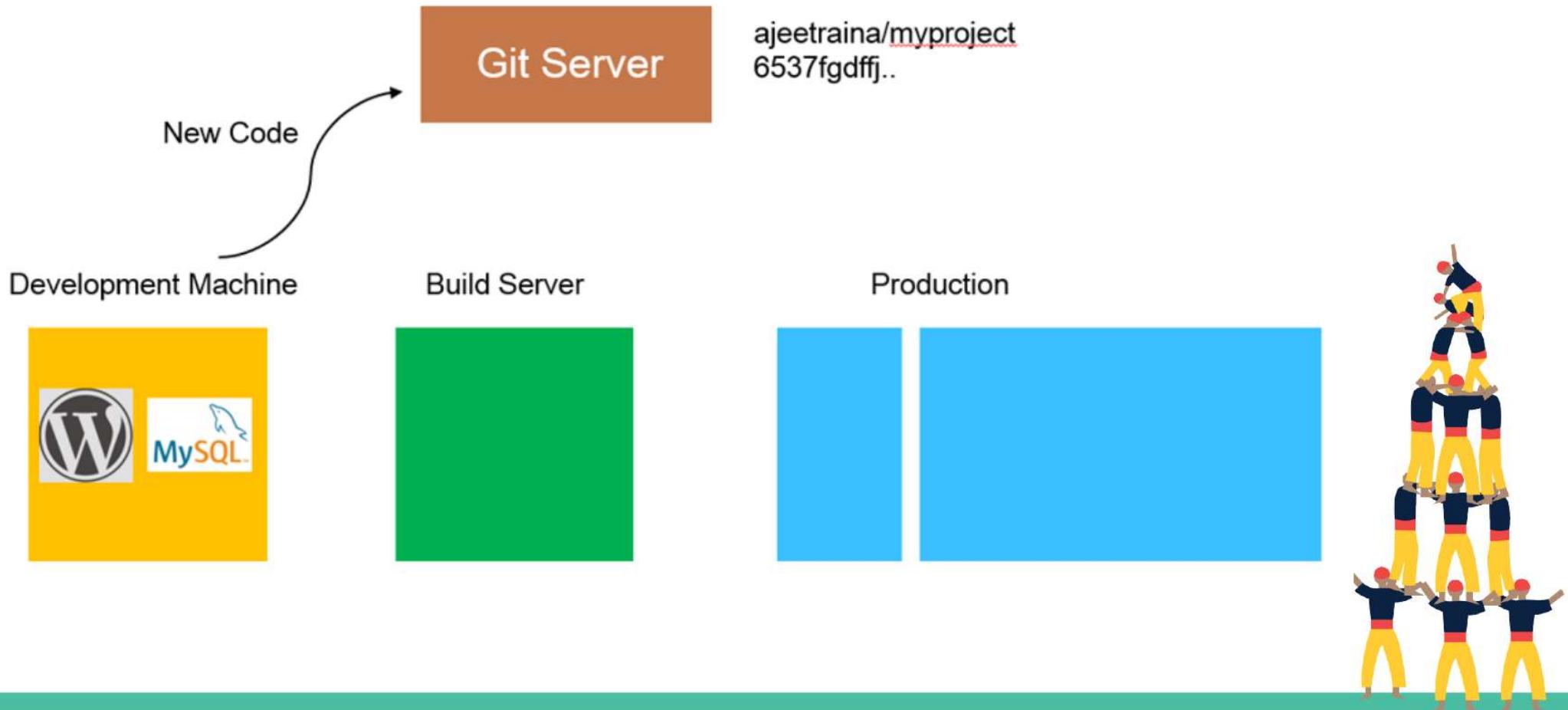


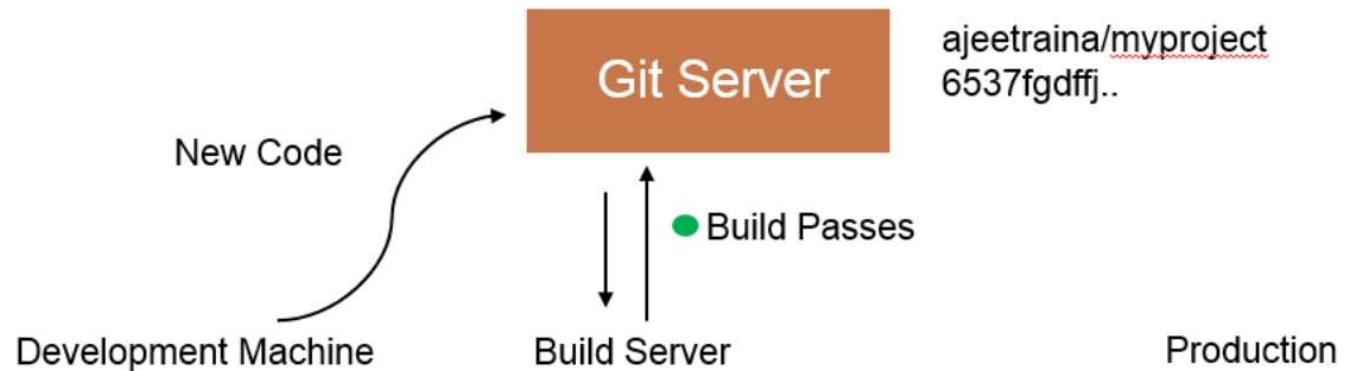
Build Server



Production

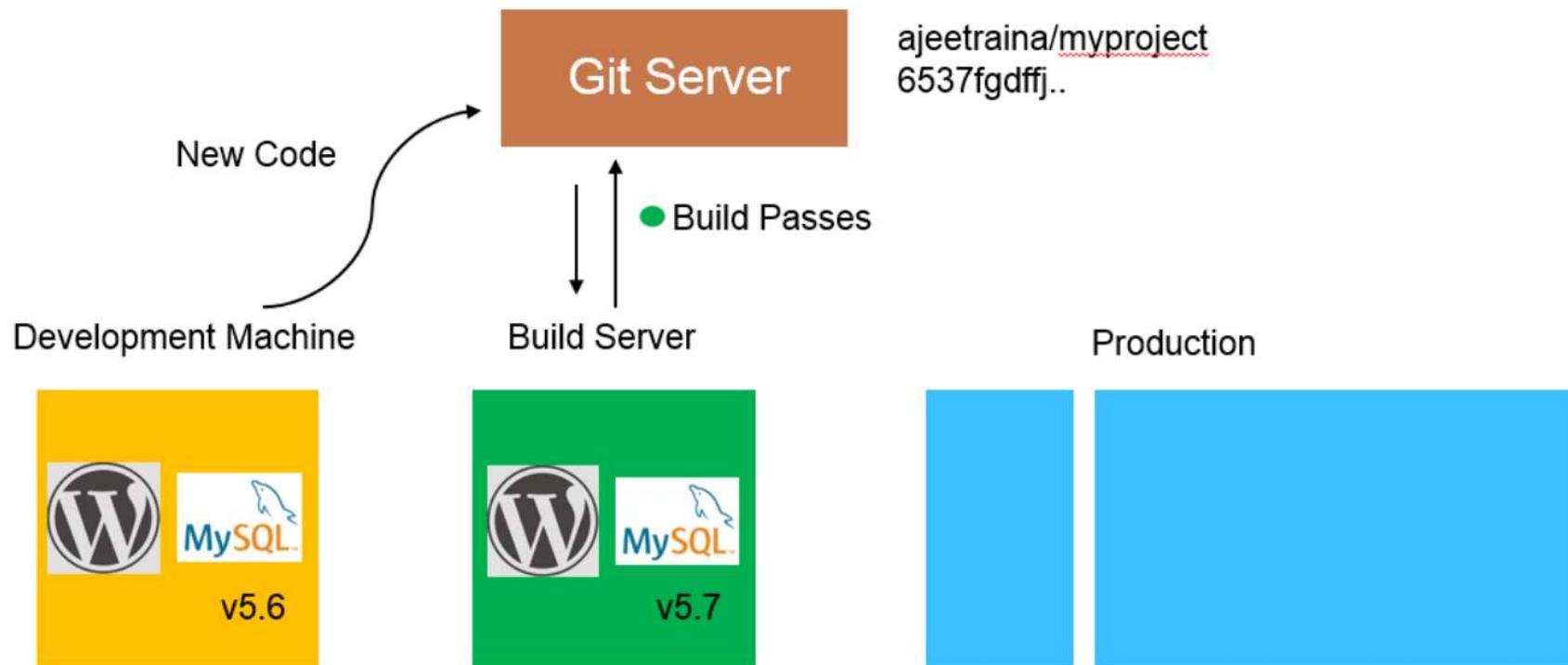


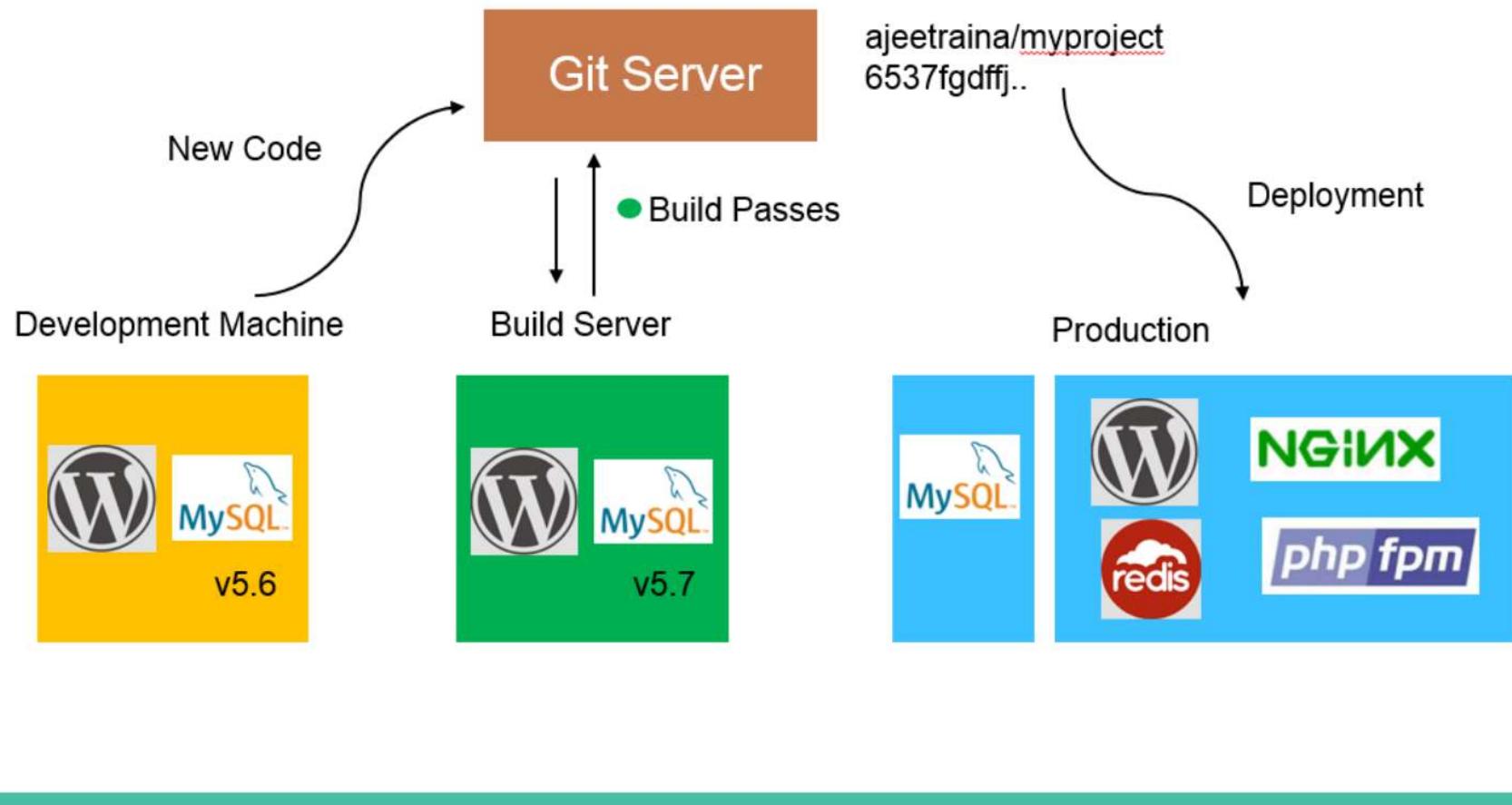




ajeetraina/myproject  
6537fgdffj..









# **Traditional Software Development WorkFlow (with Docker)**

Git Server

Docker Registry

Development Machine



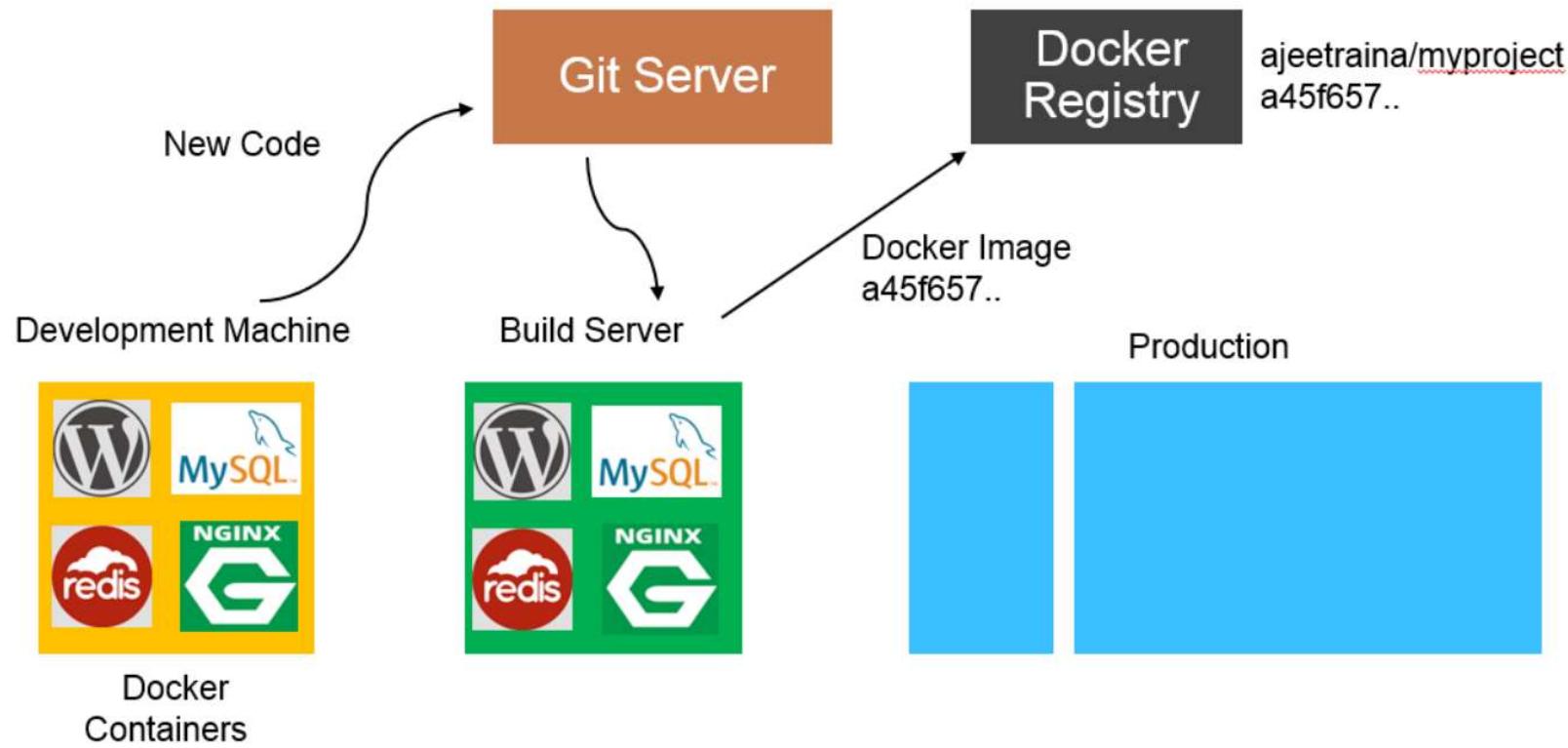
Docker  
Containers

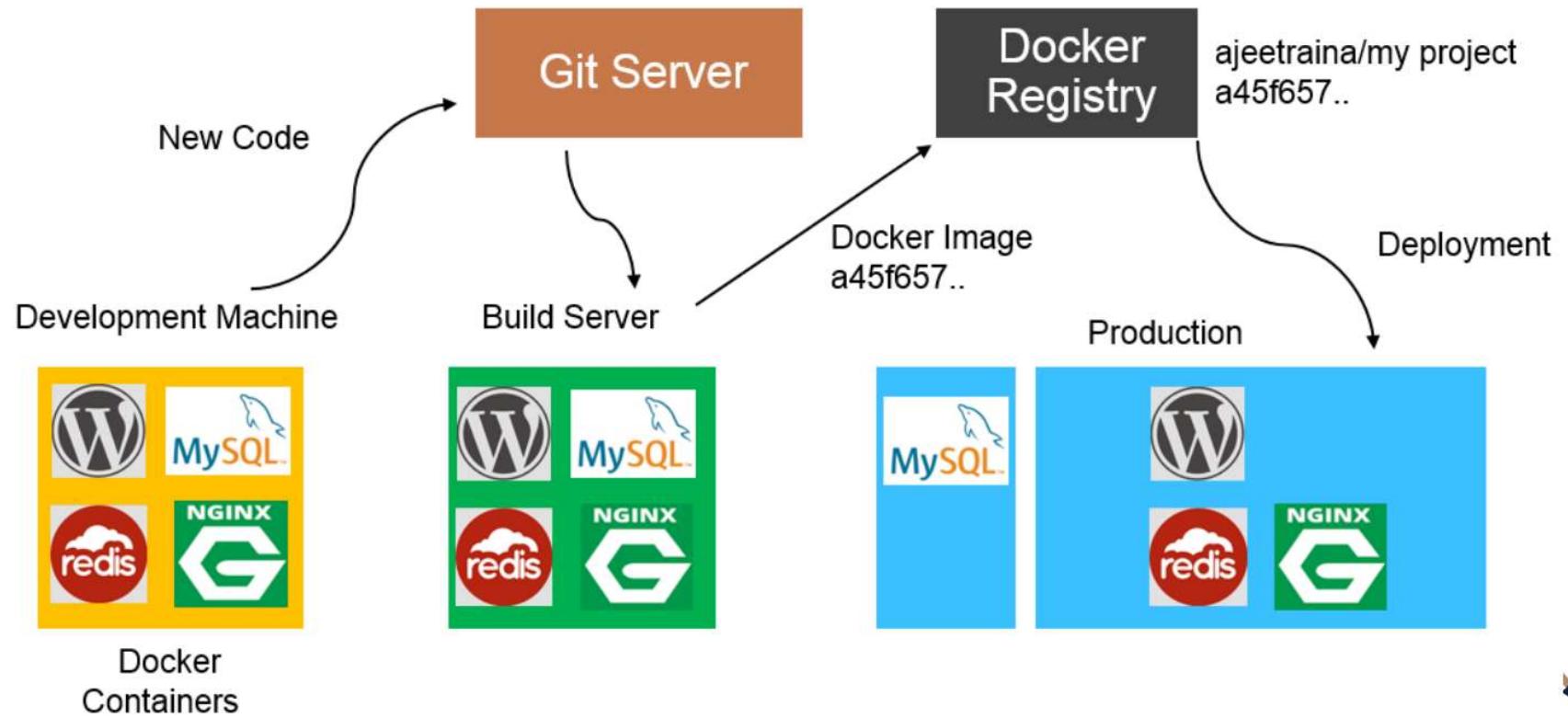
Build Server



Production









# Docker Vs VM

# Docker Containers are NOT VMs



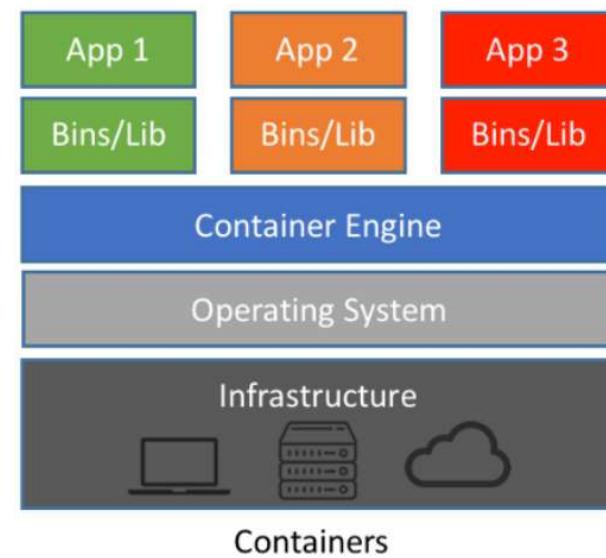
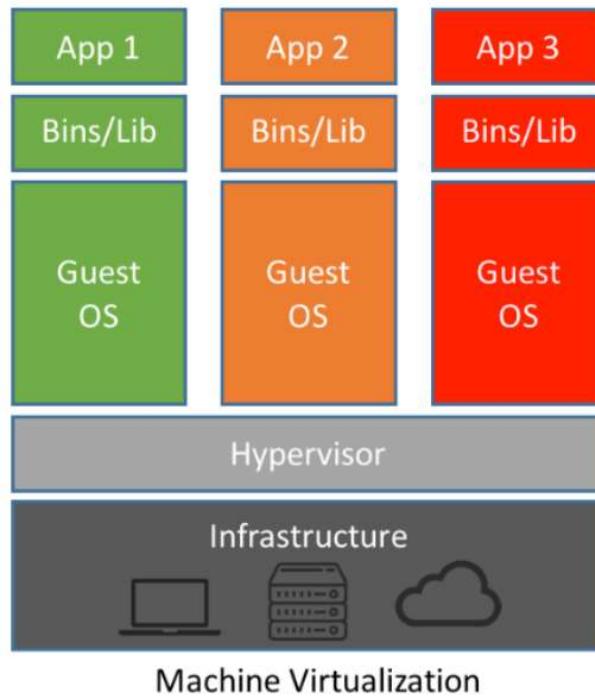
# Virtual Machine



# Containers



# Docker Container Vs VM



# 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

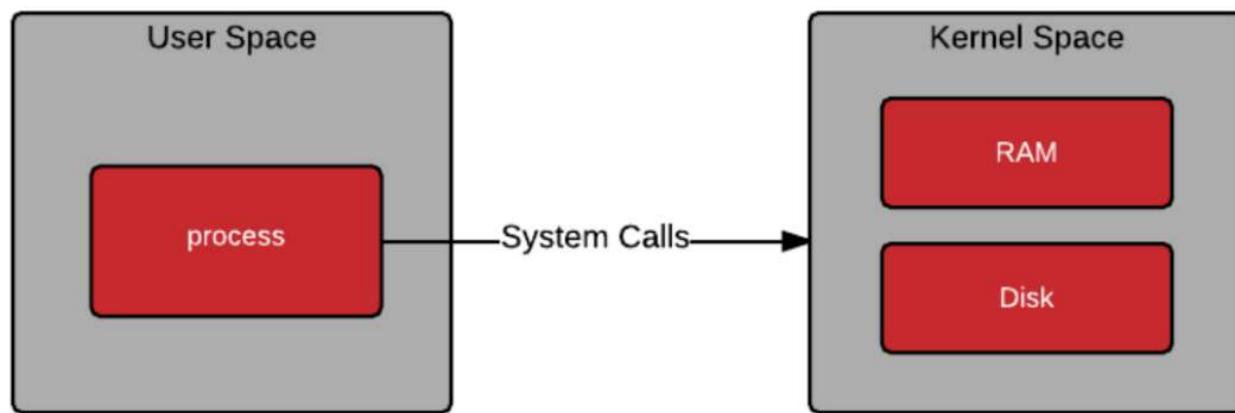




# What makes Containers so small?

# **Container = User Space of OS**

User space refers to all of the code in an operating system that lives outside of the kernel.



## **Process Virtualization**





# Demonstrating Process Virtualization

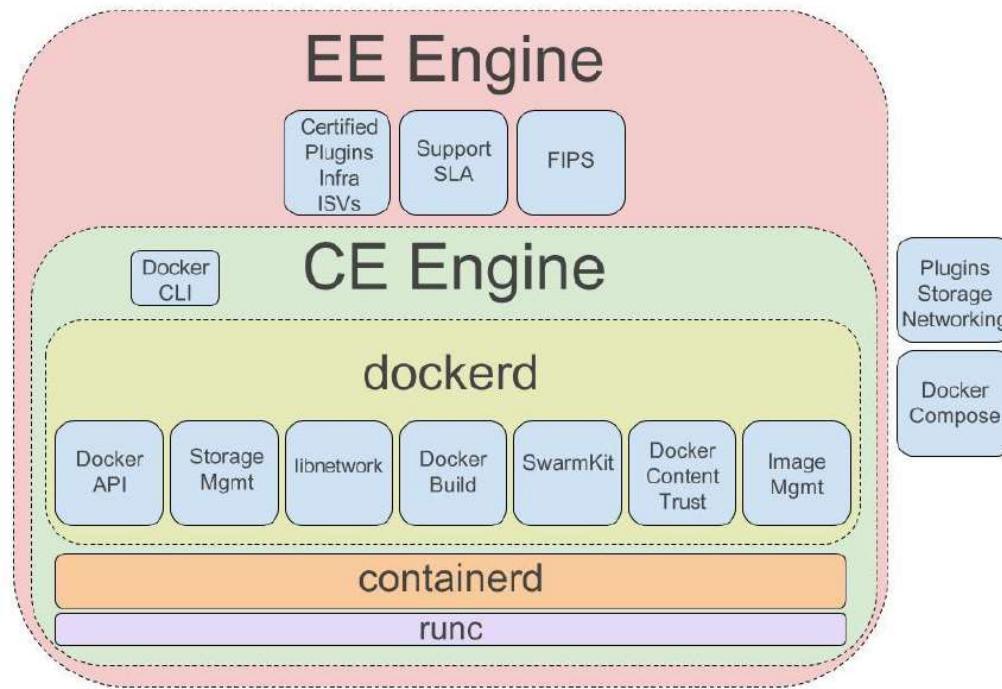
# Demonstration Process Virtualization

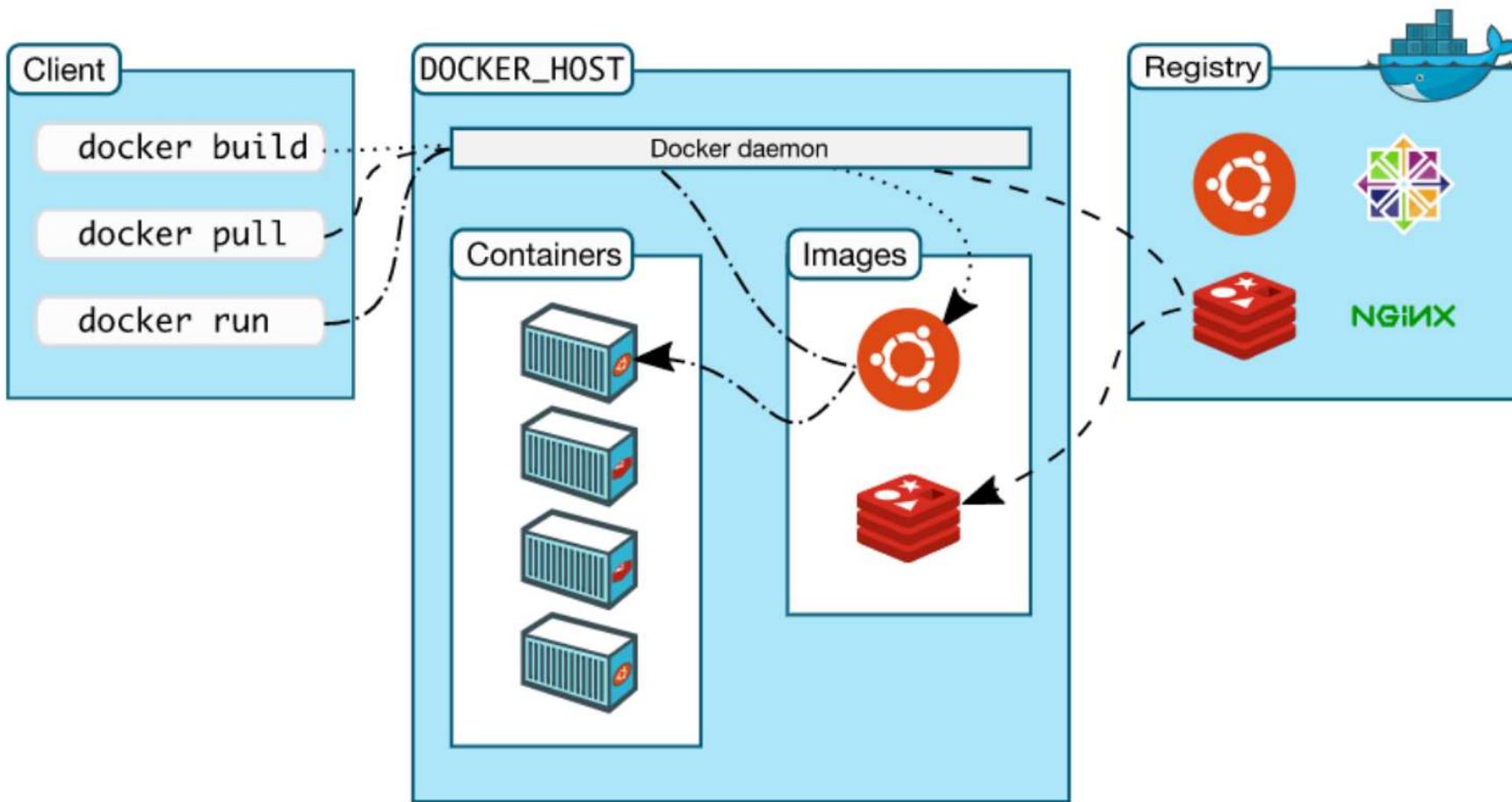




# Docker Engine Architecture

# Docker Engine Architecture







# Docker Vocabulary

# Some Docker vocabulary

## Containers

How you **run**  
your application

## Images

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

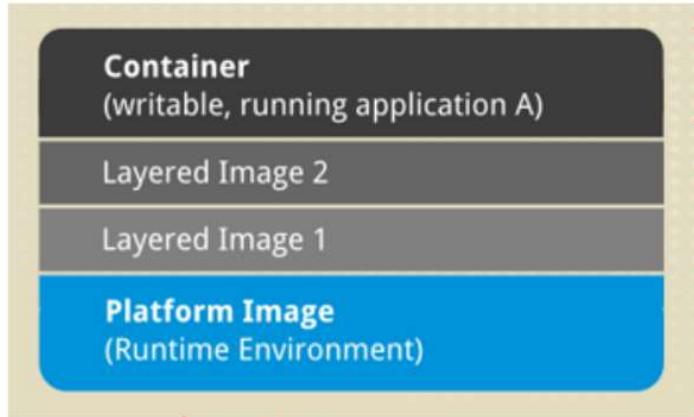


### Registry Service (Docker Hub or Docker Trusted Registry)

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.

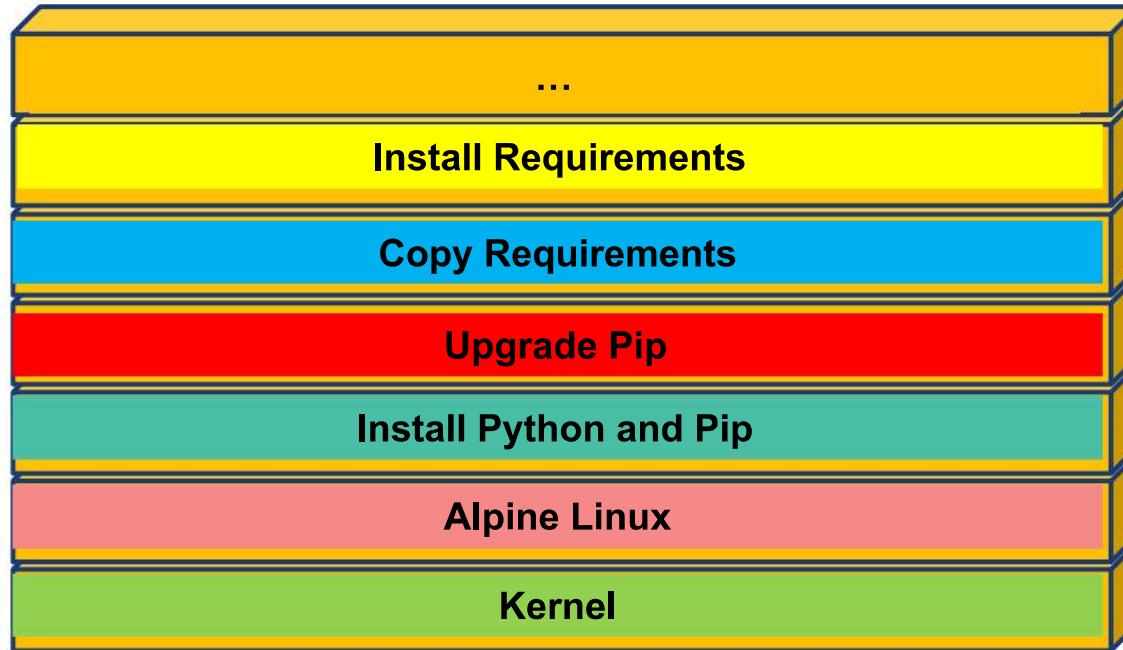
- Image is a read-only layer that is never modified, all changes are made in top-most writable layer, and can be saved only by creating a new image.
- Each image depends on one or more parent images

- An image that has no parent.

- Platform images define the runtime environment, packages and utilities necessary for containerized application to run.



# Image Layers



# A Look at “Dive” Tool

A tool for exploring each layer in a Docker image

[• Layers]			[Current Layer Contents]			
Cmp	Size	Command	Permission	UID:GID	Size	File
█	5.6 MB	FROM sha256:03901b4a	drwxr-xr-x	0:0	833 kB	
	1.4 MB	apk update	-rwxrwxrwx	0:0	0 B	
	16 MB	apk add git	-rwxrwxrwx	0:0	0 B	
[Layer Details]			-rwxrwxrwx	0:0	0 B	
Digest:	sha256:03901b4a2ea88eeaad62dbe59			-rwxrwxrwx	0:0	0 B
b072b28b6efa00491962b8741081c5df50c65e0			-rwxrwxrwx	0:0	0 B	
Command:	-rwxrwxrwx			-rwxrwxrwx	0:0	0 B
#(nop) ADD file:fe64057fbb83dccb960efabb			-rwxrwxrwx	0:0	0 B	
f1cd8777920ef279a7fa8dbca0a8801c651bdf7c			-rwxrwxrwx	0:0	0 B	
in /			-rwxrwxrwx	0:0	0 B	
[Image Details]			-rwxrwxrwx	0:0	0 B	
Total Image size:	23 MB		-rwxrwxrwx	0:0	0 B	
Potential wasted space:	70 kB		-rwxrwxrwx	0:0	0 B	
Image efficiency score:	99 %		-rwxrwxrwx	0:0	0 B	
█^C Quit █Tab Switch view █^F Filter █^L Show layer changes █^A Show aggregated						



<https://github.com/wagoodman/dive>

# Basic Docker CLIs

## Pulling Docker Image

```
$ docker pull ajeettraina/hellowhale
```

## Listing out Docker Images

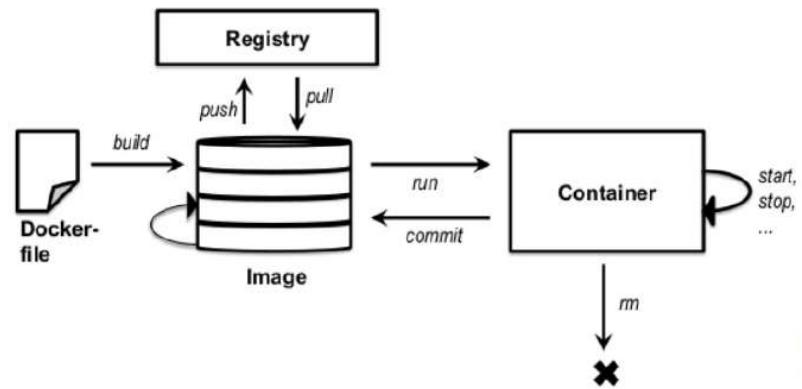
```
$ docker image ls
```

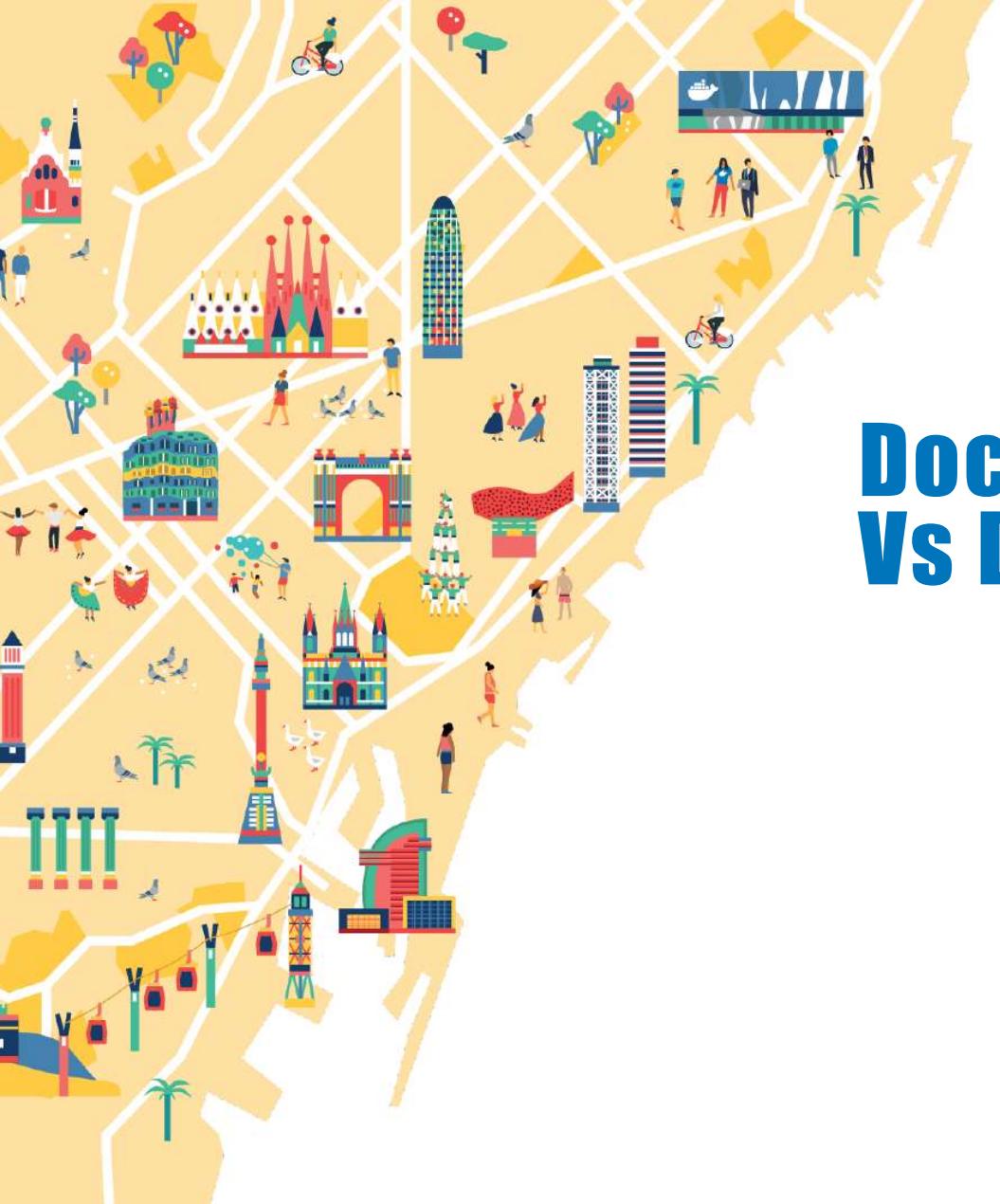
## Running Docker Containers

```
$ docker run -d -p 5000:5000 --name hellowhale ajeettraina/hellowhale
```

## Stopping the container

```
$ docker stop hellowhale (or <container id>)
```





# Docker Desktop Community Vs Docker Enterprise

# Docker Community Edition

All in one development for Swarm and Kubernetes

Develop with Docker  
Community Edition  
on your workstation



Test locally on Swarm  
and Kubernetes



Deploy to production  
in Swarm

Deploy to production  
in Kubernetes



# Docker Desktop Community Vs Enterprise

	Docker Desktop Community	Docker Desktop Enterprise
--	-----------------------------	------------------------------

Simplest Path to Container-based Development

Latest Docker Engine Based on Containerd	●	●
Certified Kubernetes	●	●
Available for Windows 10 & MacOS	●	●
Develop in any Language & Framework, even multiple versions simultaneously	●	●



# Docker Desktop Community Vs Enterprise

	Docker Desktop Community	Docker Desktop Enterprise
--	-----------------------------	------------------------------

## Production-Ready Environment

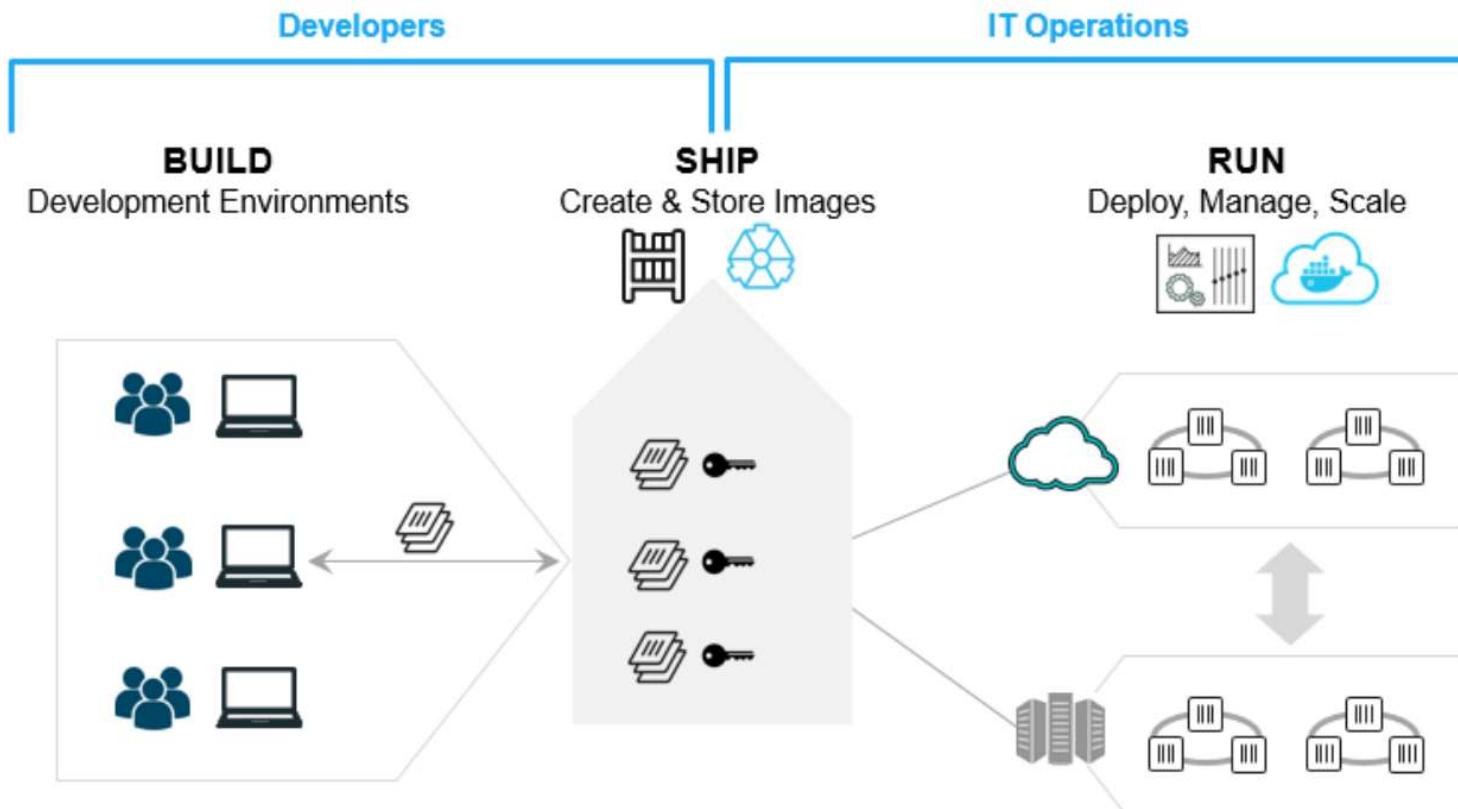
Same API & Commands shared by Developers in Production	●	●
Application Designer interface to simplify creating & developing Docker applications		●
Swappable Docker Engine and Kubernetes versions to match Docker Enterprise production environments		●





# 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](#)





# Docker New CLI Plugin

# Docker New CLI Plugin

Plugins	Delivery Vehicle	Availability
app	19.03-ce, 19.03-ee, Desktop CE, Desktop EE	Available now via Engine Community and Desktop Community (Mac   Windows) Docker Enterprise 3.0
assemble	19.03-ee, Desktop Enterprise	Available with Docker Enterprise 3.0
template	19.03-ee, Desktop Enterprise	Available with Docker Enterprise 3.0
cluster	19.03-ee, Desktop Enterprise	Available with Docker Enterprise 3.0
gmsa	19.03-ee, Desktop Enterprise	Available with Docker Enterprise 3.0
registry	19.03-ee, Desktop Enterprise	Available with Docker Enterprise 3.0
buildx	desktop-edge	Available in Desktop CE Edge (Mac   Windows), download the plugin
jump	<a href="#">Sign up for more information</a>	<a href="#">Sign up for more information</a>
pipeline	<a href="#">Sign up for more information</a>	<a href="#">Sign up for more information</a>





# **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
```

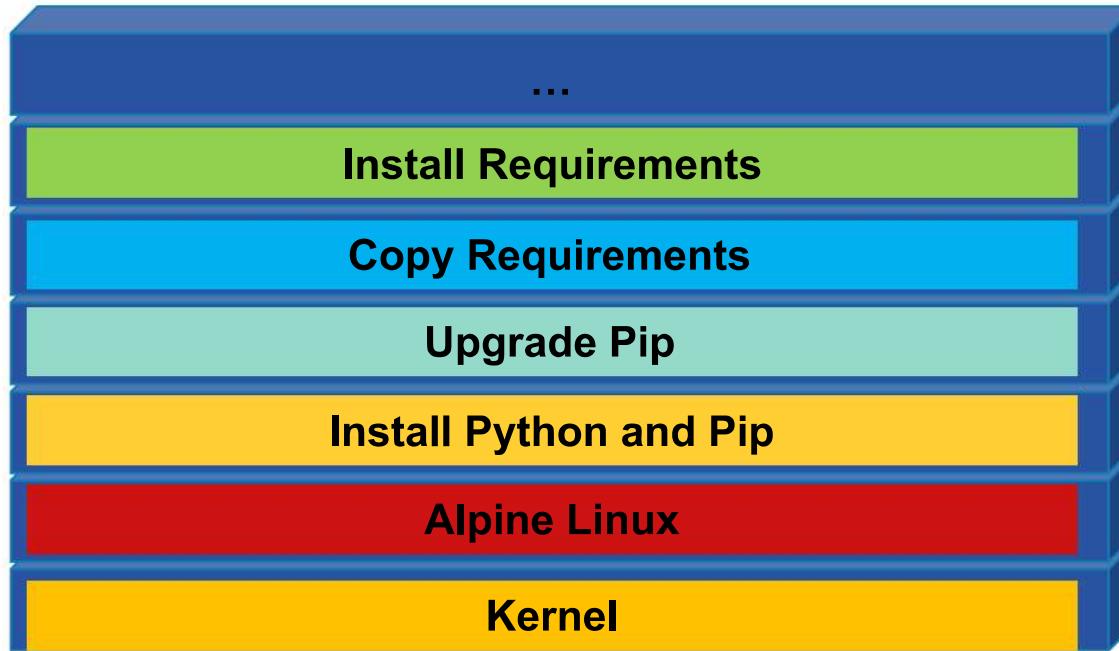


# Dockerfile – Example

```
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

