Java on Kubernetes 101 @DevNexus

Marek Sadowski

IBM San Francisco

ibm.biz/2021-devnexus-k8s

V1 2021 02 18

IBM **Developer**

@blumareks

IBM

Java – Swift – Mobile – Containers – Serverless – AI (Watson) with Visual Recognition, Voice UI, NLU, ...
Last year I co-authored and published a book on Serverless Swift

Rescue Scuba Diver – Snowboarder – Martial Arts - Startups

@blumareks

Lab instructions - http://ibm.biz/2021-devnexus-k8s

Signup to IBM Cloud http://ibm.biz/2021-devnexus-k8s-cloud
Claim your cluster http://ibm.biz/2021-devnexus-k8s-claim
(key: ikslab)
lab 1 - ibm.biz/2021-devnexus-lab1
lab 2 - ibm.biz/2021-devnexus-lab2
lab 3 - ibm.biz/2021-devnexus-lab3
lab 4 - ibm.biz/2021-devnexus-lab4
lab 5 - ibm.biz/2021-devnexus-lab5

Discord: our 1:1s
DM me on Twitter: https://twitter.com/blumareks

Agenda:

1. prerequisites
2. containers to host microservices
3. Docker to build-ship-run containers
4. orchestration of many containers
5. Kubernetes
6. Helm
7. Istio
8. OKD
9. Red Hat OpenShift vs Kubernetes

Lab 1 Prerequisites

A Browser based lab only – 3 tabs:
- lab instructions http://ibm.biz/2021-devnexus-k8s
- IBM Cloud – [our cloud administration]
- Skill Network Lab – [our development space]

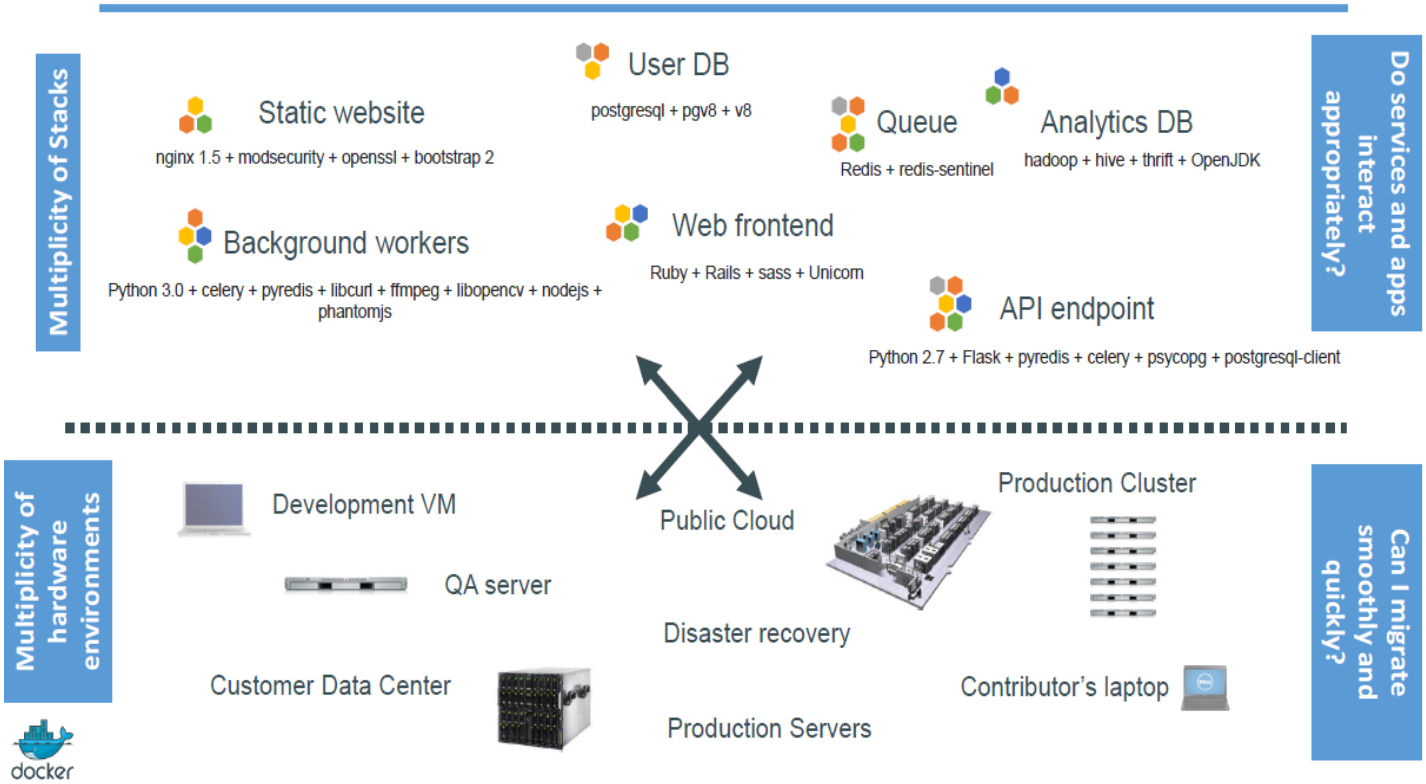
and Discord – just in case…

Why Containers?

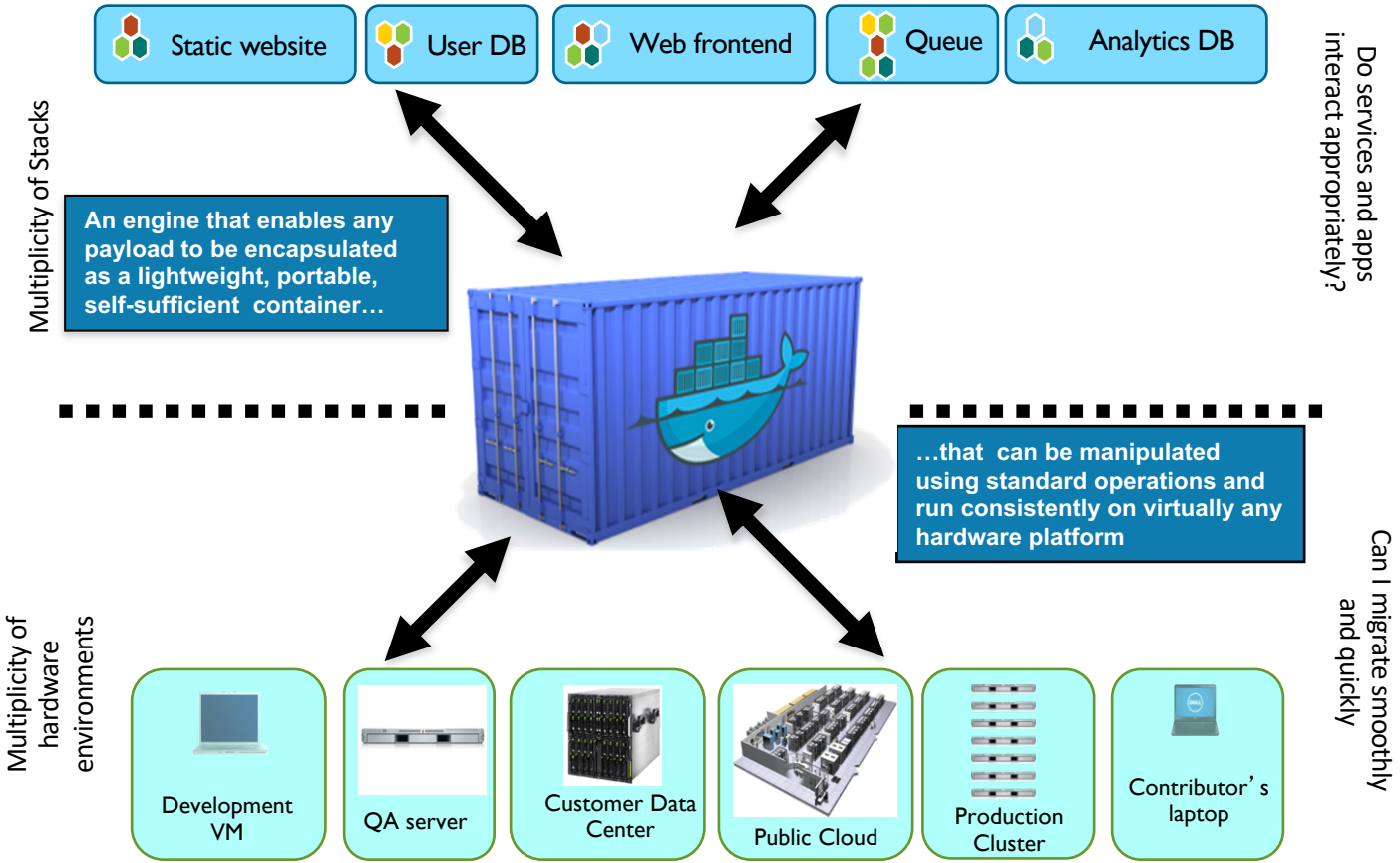
Everyone Loves Containers

- A standard way to package an application and all its dependencies so that it can be moved between environments and run without changes
- Containers work by isolating the differences between applications inside the container so that everything outside the container can be standardized

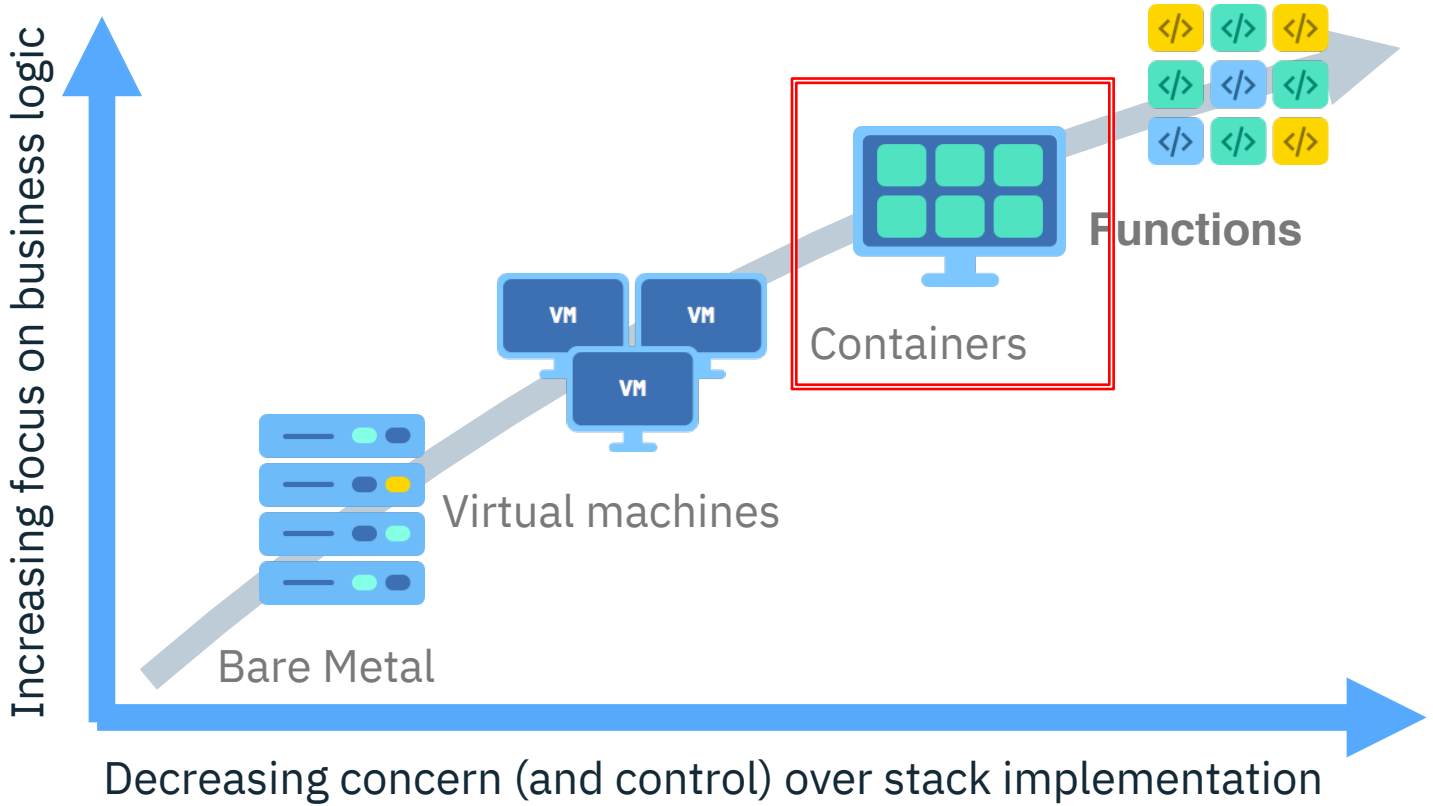
The Challenge



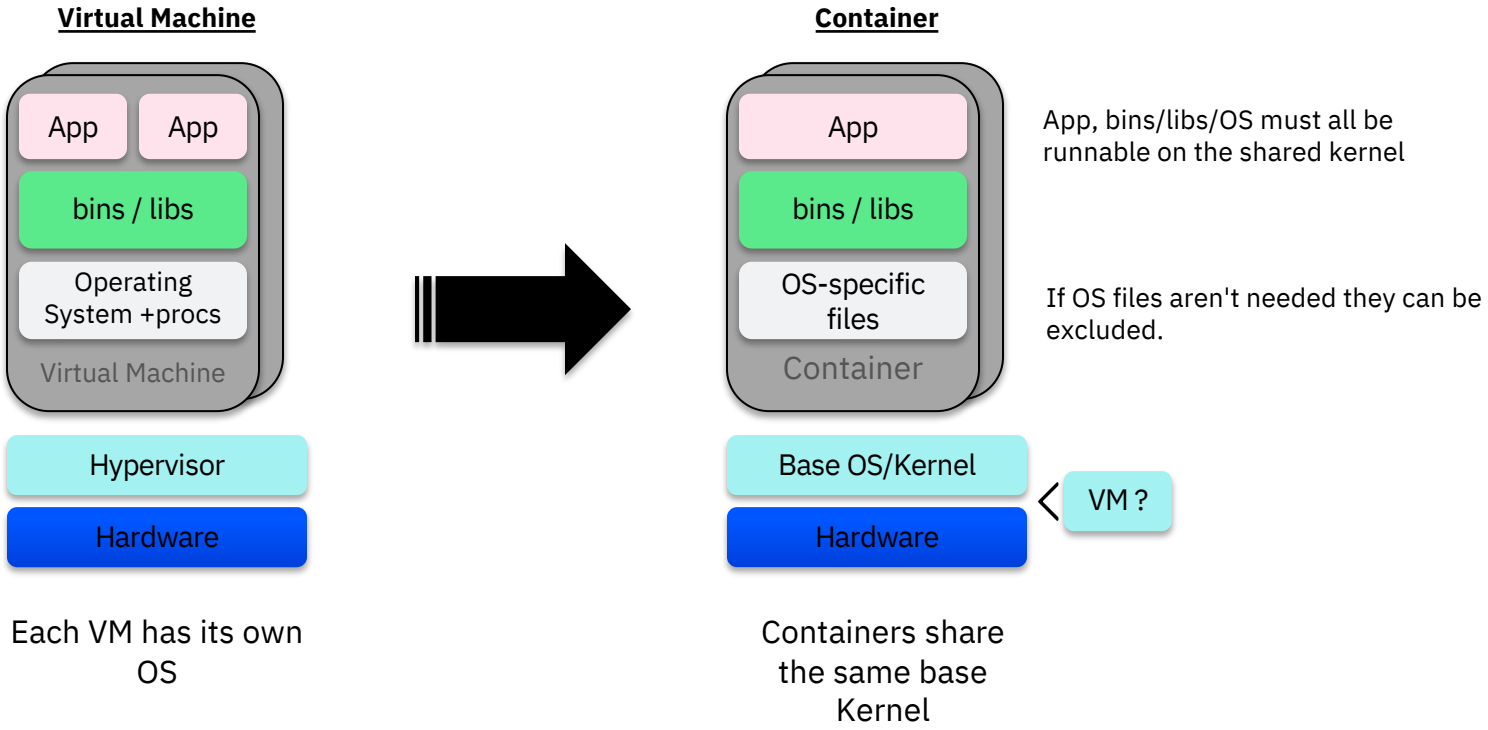
The Solution - A Shipping Container for Code



How we got here

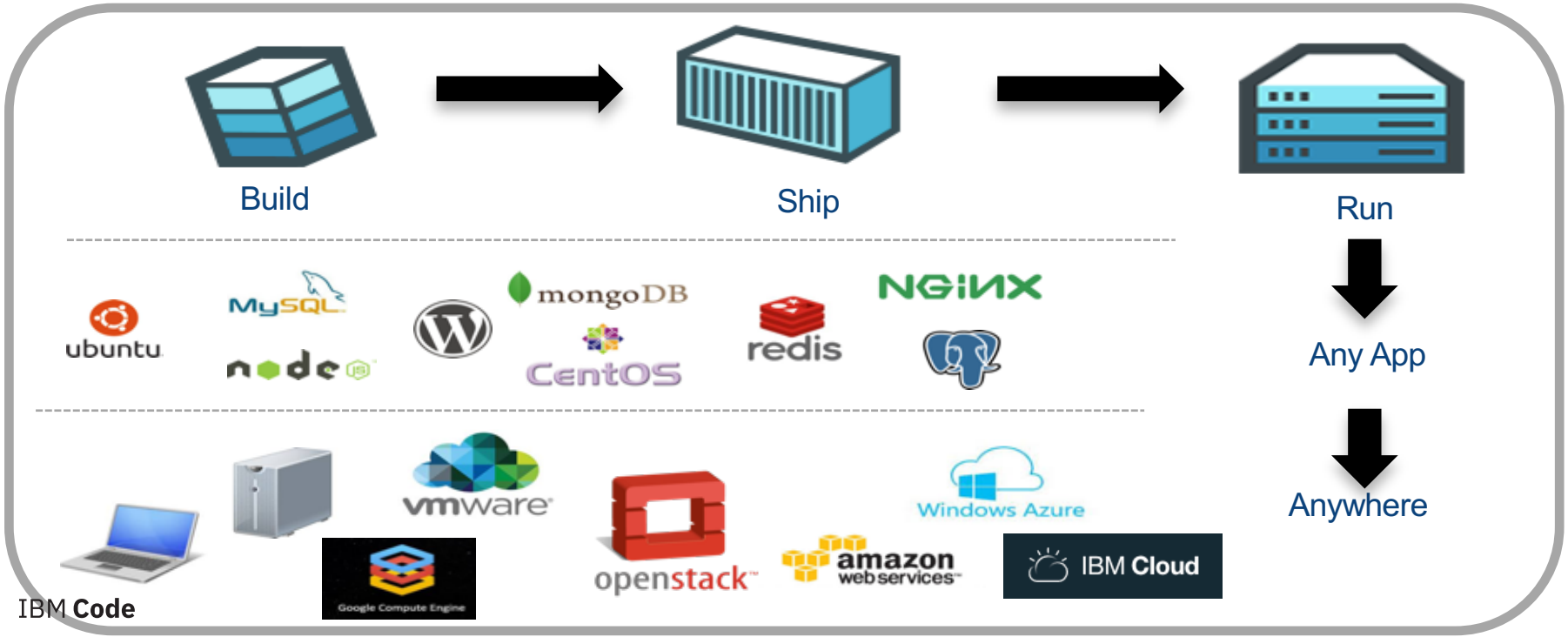


VM vs Container

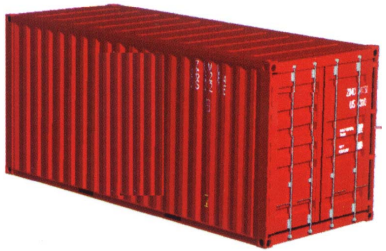


Docker Mission

Docker is an **open platform** for building distributed applications for **developers** and **system administrators**



Dev vs. Ops

Dev	Ops
	
• **Code** • **Libraries** • **Configuration** • **Server runtime** • **OS**	• **Logging** • **Remote access** • **Network configuration** • **Monitoring**

Separation of concerns

A container separates and bridges the Dev and Ops in DevOps

- Dev focuses on the application environment
- Ops focuses on the deployment environment

Docker Component Overview

Docker Engine

- Manages containers on a host
- Accepts requests from clients
  - REST API
- Maps container ports to host ports
  - E.g. 80 → 3582

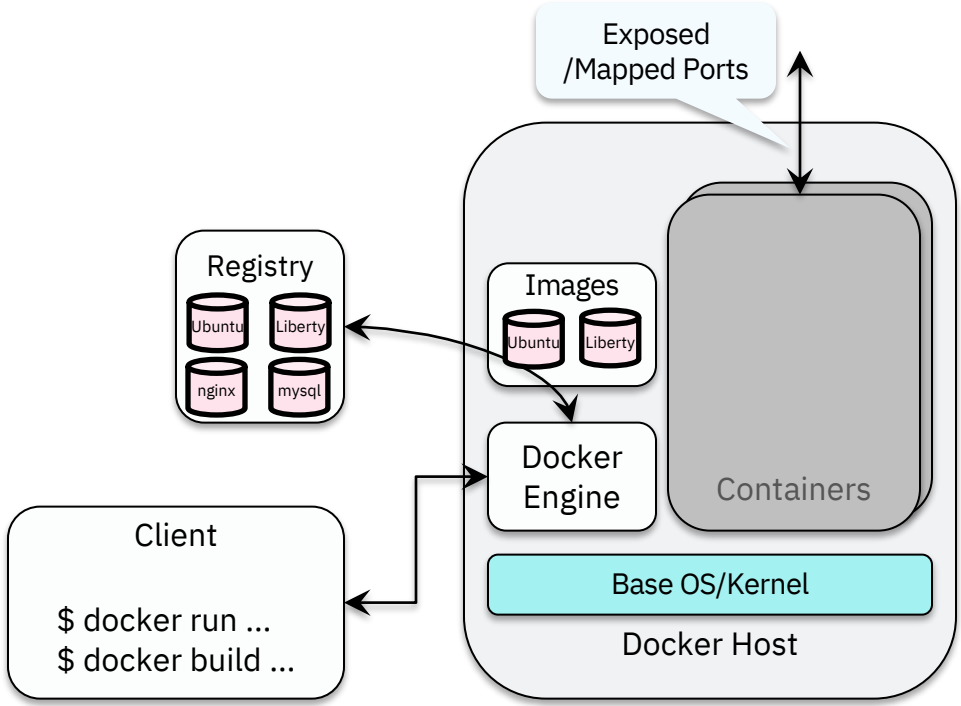
Images

Docker Client

- Drives engine
- Drives "builder" of Images

Docker Registry

- Image DB



Shared / Layered / Union Filesystems

Docker uses a copy-on-write (union) filesystem

New files(& edits) are only visible to current/above layers

Layers allow for reuse

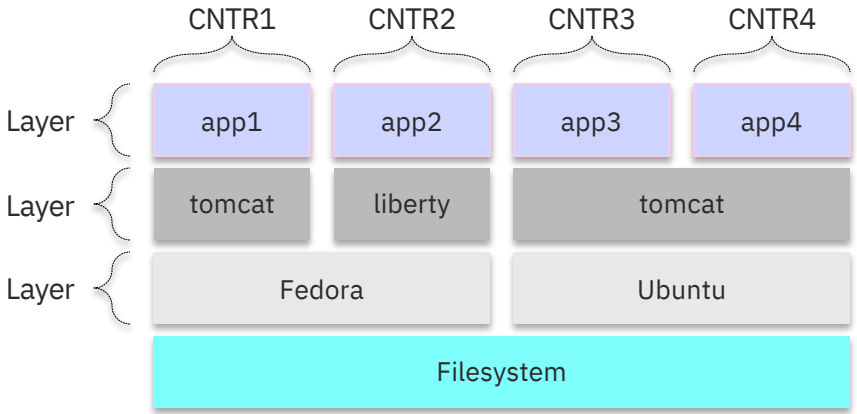
– More containers per host

– Faster start-up/download time

Images

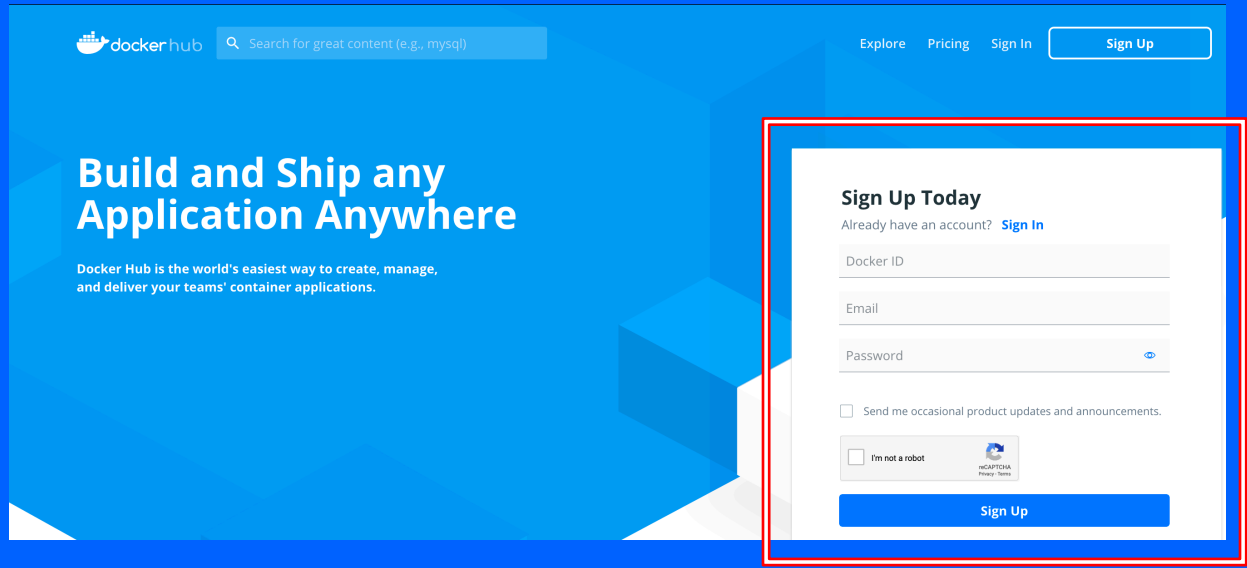
– Tarball of layers

Think: Transparencies on projector



Hands-on

Download FREE Docker Desktop and use your terminal on your computer
or just sit back and watch



Our First Container

```
$ docker run ubuntu echo Hello World
Hello World
```

What happened?

- Docker created a directory with a "ubuntu" filesystem (image)
- Docker created a new set of namespaces
- Ran a new process: `echo Hello World`
  - Using those namespaces to isolate it from other processes
  - Using that new directory as the "root" of the filesystem (`chroot`)
- That's it!
  - Notice as a user I never installed "ubuntu"
- Run it again - notice how quickly it ran

```
Mareks-MBP:~ mareksadowski$ docker run ubuntu echo Hello World
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
6b98dfc16071: Pull complete
4001a1209541: Pull complete
6319fc68c576: Pull complete
b24603670dc3: Pull complete
97f170c87c6f: Pull complete
Digest: sha256:5f4bdc3467537cbbe563e80db2c3ec95d548a9145d64453b06
Status: Downloaded newer image for ubuntu:latest
Hello World
Mareks-MBP:~ mareksadowski$ docker run ubuntu echo Hello World
Hello World
Mareks-MBP:~ mareksadowski$
```

A look under the covers

```
$ docker run ubuntu ps -ef
UID          PID    PPID  C
STIME TTY          TIME CMD
root           1       0  0
14:33 ?        00:00:00 ps -ef
```

Things to notice with these examples

- Each container only sees its own process(es)
- By default running as "root”,
- And running as PID 1
- The good security practice is to run services as the user other than root, and avoid the privileged mode
- Limit the resources used by container – to avoid noisy neighbor effect

Second part –
the orchestration
of successful API deployment

Containers

Everyone’s container journey starts with one container....

Containers

At first the growth is easy to handle....

Containers

But soon it is overwhelming... chaos reigns

Container Orchestration

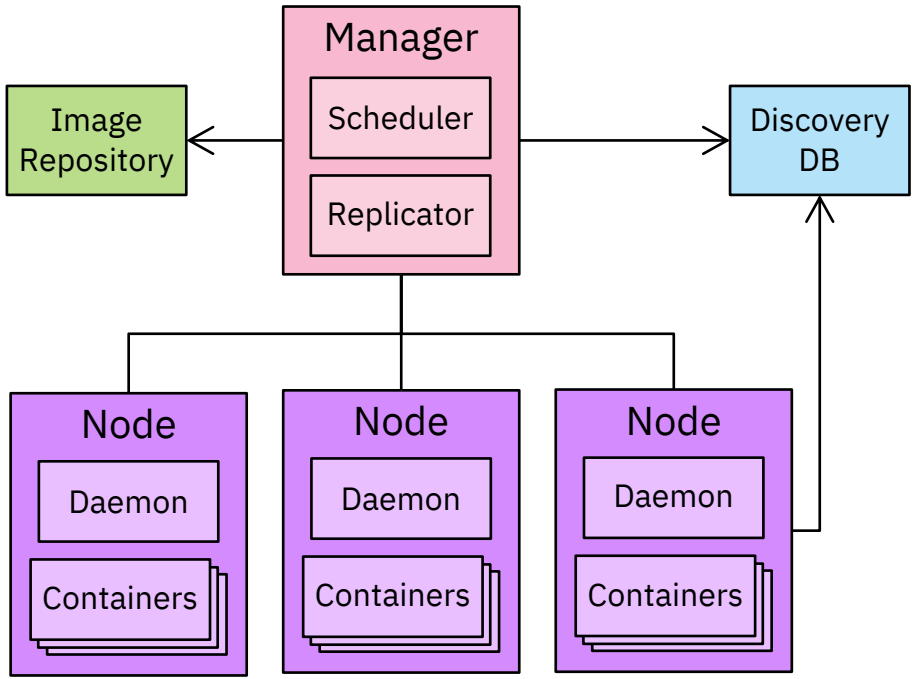
Allows users to define how to coordinate the containers in the cloud when the multi-container packaged application is deployed.

- Scheduling
- Cluster management
- Service discovery
- Provisioning
- Monitoring
- Configuration management

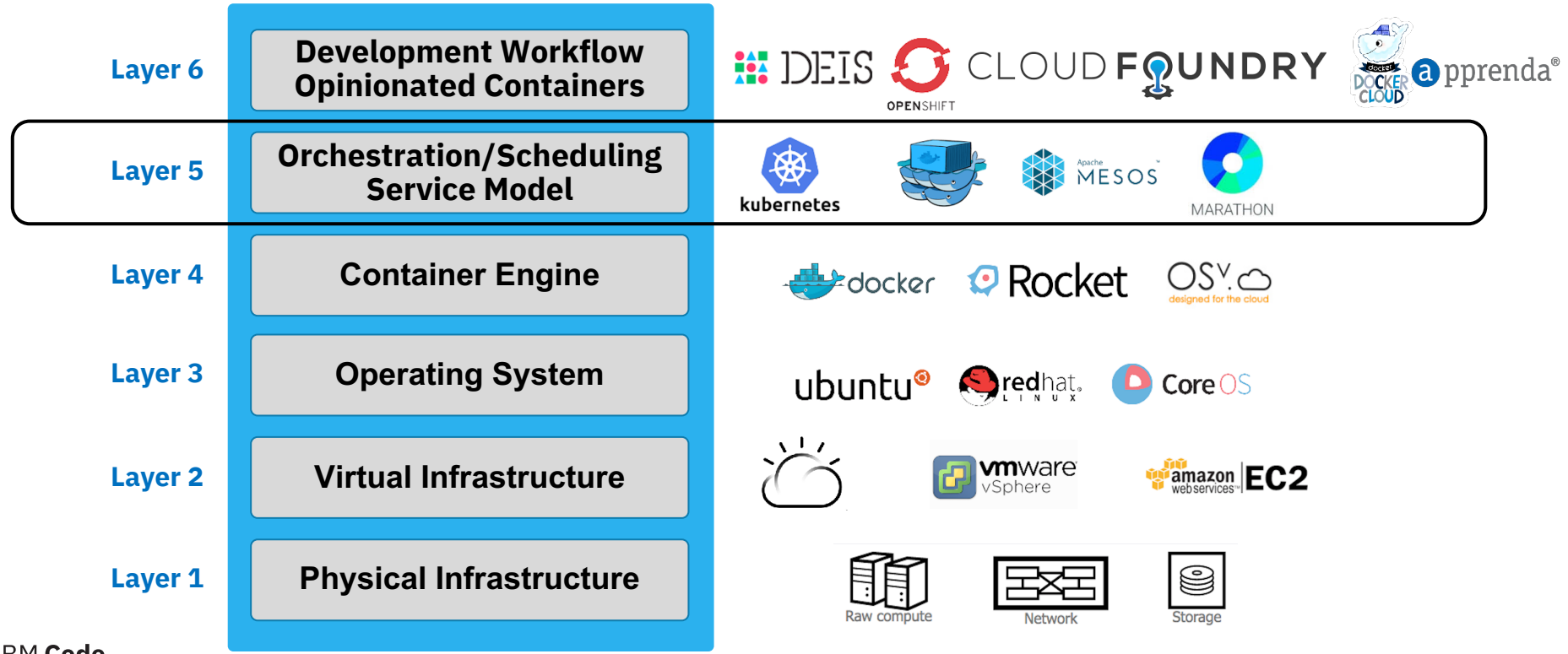
What is Container Orchestration?

- Container orchestration
  - Cluster management
  - Scheduling
  - Service discovery
  - Replication
  - Health management

Container Orchestrator



Container Ecosystem Layers

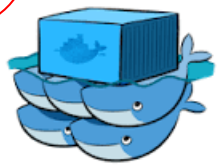


Kubernetes – Mesos – Marathon – Docker Swarm

Kubernetes is a cluster manager for containers (large deployments)

MESOS is a distributed system kernel that will make your cluster look like one giant computer system to all supported frameworks and apps that are build to be run on mesos.

(Kubernetes can be run on Mesos)



Swarm has a YAML-based deployment model, auto-healing of clusters, overlay networks with DNS, high-availability through the use of multiple masters, and network security using TLS with a Certificate Authority (1-10 containers)

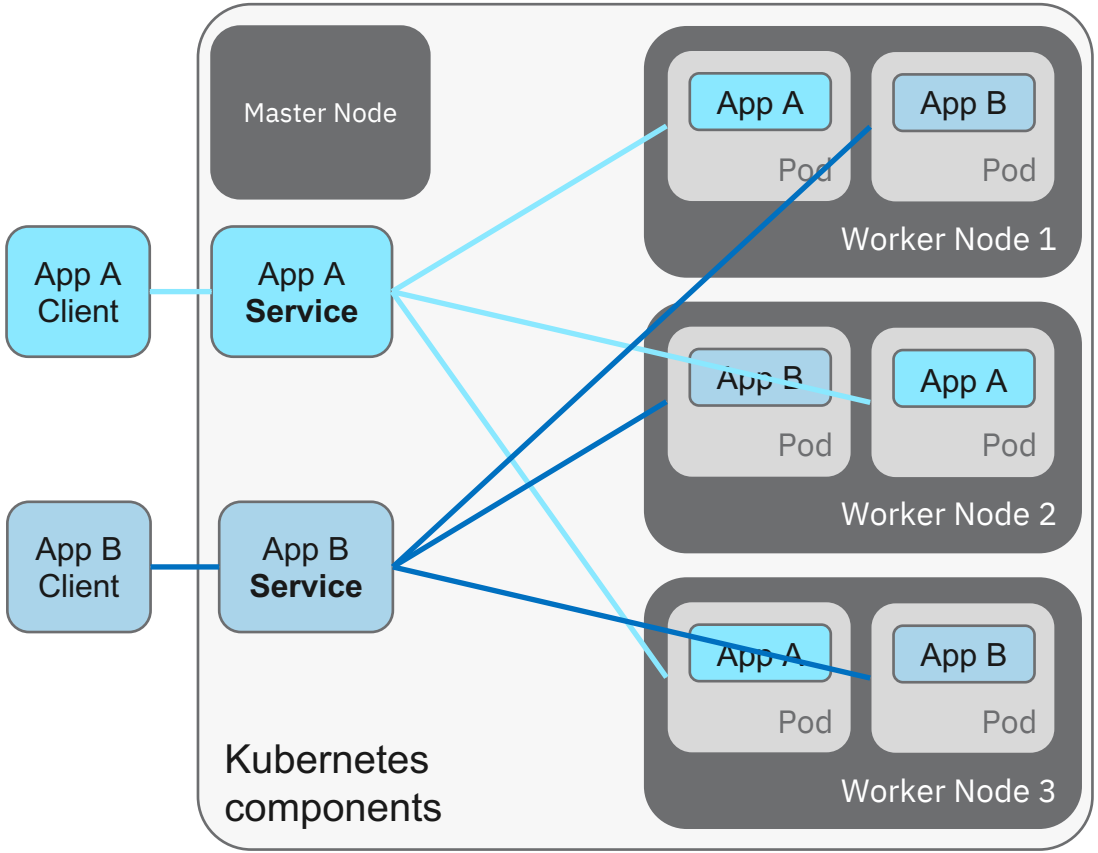
Marathon is a cluster-wide init and control system for running Linux services in cgroups and Docker containers, and it is run on top of Mesos

What is Kubernetes?

- Container orchestrator
- Manage applications, **not machines**
- Designed for extensibility
- Open source project managed by the Linux Foundation

Kubernetes Architecture: Workloads

- Container
  - Packaging of an app
- Pod
  - Unit of deployment
- Service
  - Fixed endpoint for 1+ pods



Kubernetes

Intelligent Scheduling

Self-healing

Horizontal scaling

Service discovery & load balancing

Automated rollouts and rollbacks

Secret and configuration management

Helm & Helm Chart

- Helm
  - Package manager for Kubernetes
  - Used to manage Kubernetes applications
- Helm Chart
  - Used to define, install, and upgrade complex Kubernetes applications
  - Easy to create, version, share and publish
  - Expressed in “Yet Another Markup Language” (YAML) files

Challenges with Microservices

Security

Canary deployments

A/B testing

Retries and Circuit breaking

Rate limiting

Fault injection

Policy management

Telemetry

What is a ‘Service Mesh’ ?

A network for services, not bytes

- Observability
- Resiliency
- Traffic Control
- Security
- Policy Enforcement

Istio

An open platform to connect, manage, and secure microservices

http://istio.io

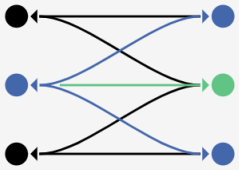
IBM Code

Istio community partners

- RedHat
- Pivotal
- WeaveWorks
- Tigera
- Datawire
- Scytale (SPIFFE)
- Microsoft
- Uber (Jaeger)

Istio

Connect, secure, control, and observe services.

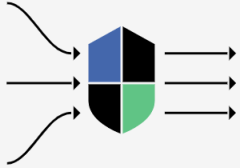


Connect

Intelligently control the flow of traffic and API calls between services, conduct a range of tests, and upgrade gradually with red/black deployments.

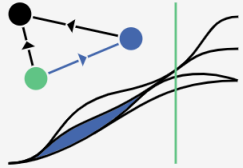
Secure

Automatically secure your services through managed authentication, authorization, and encryption of communication between services.



Control

Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.



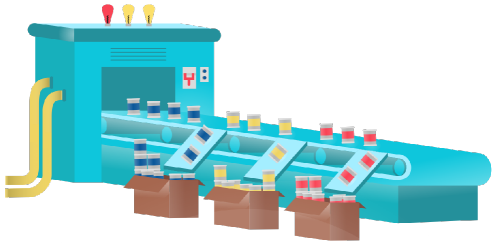
Observe

See what's happening with rich automatic tracing, monitoring, and logging of all your services.

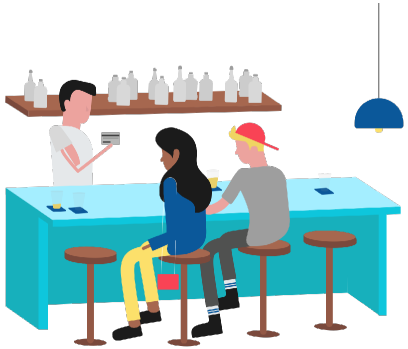
Istio

WHERE ISTIO FINDS ITS PLACE

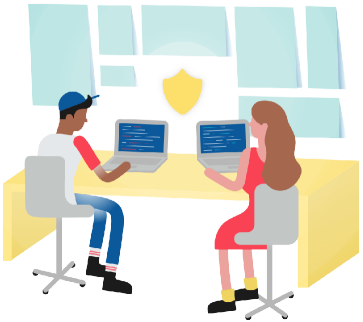
Intelligent Routing and Load Balancing



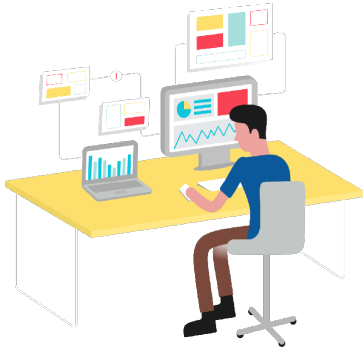
Fleet Wide Policy Enforcement



Resiliency across Languages and Platforms

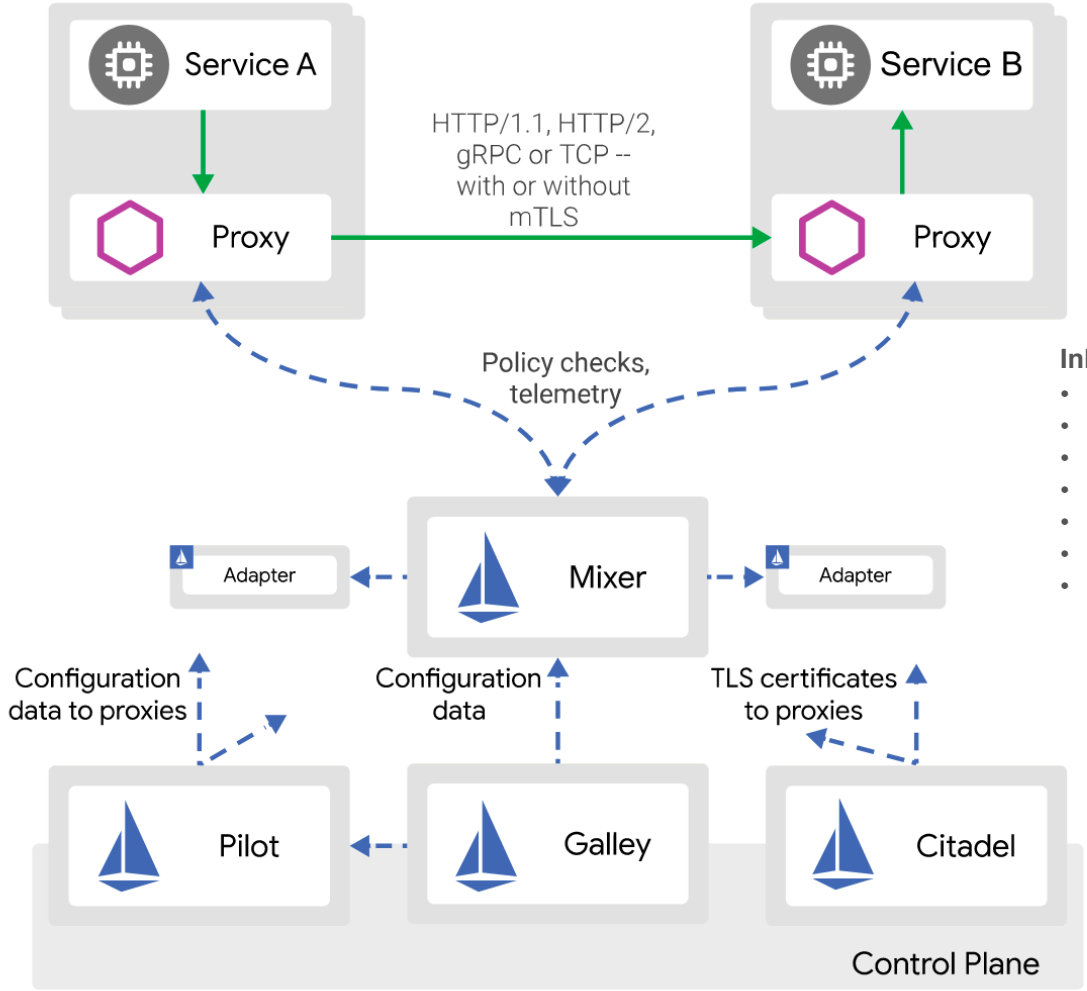


In-Depth Telemetry and Reporting



Outbound features:

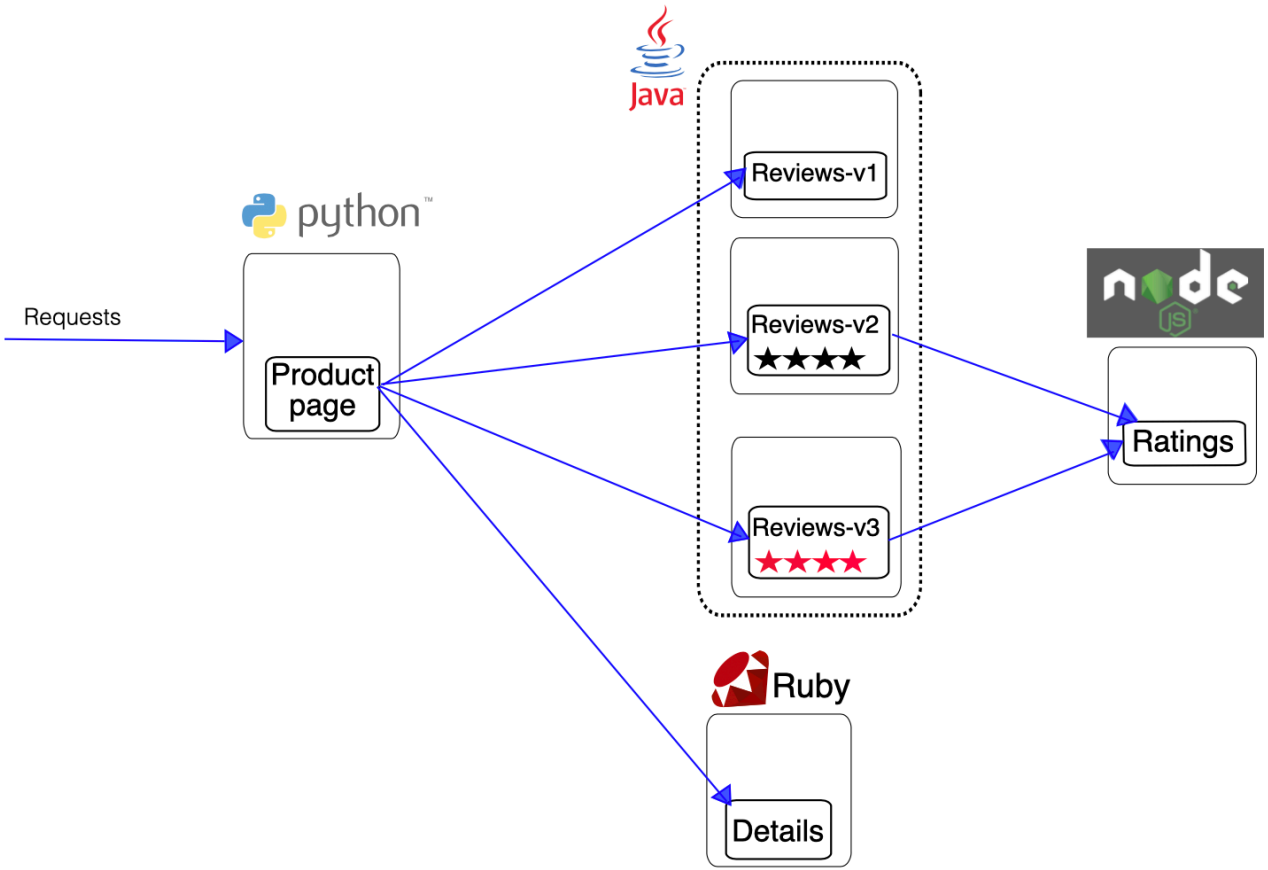
- Service authentication
- Load balancing
- Retry and circuit breaker
- Fine-grained routing
- Telemetry
- Request Tracing
- Fault Injection



Inbound features:

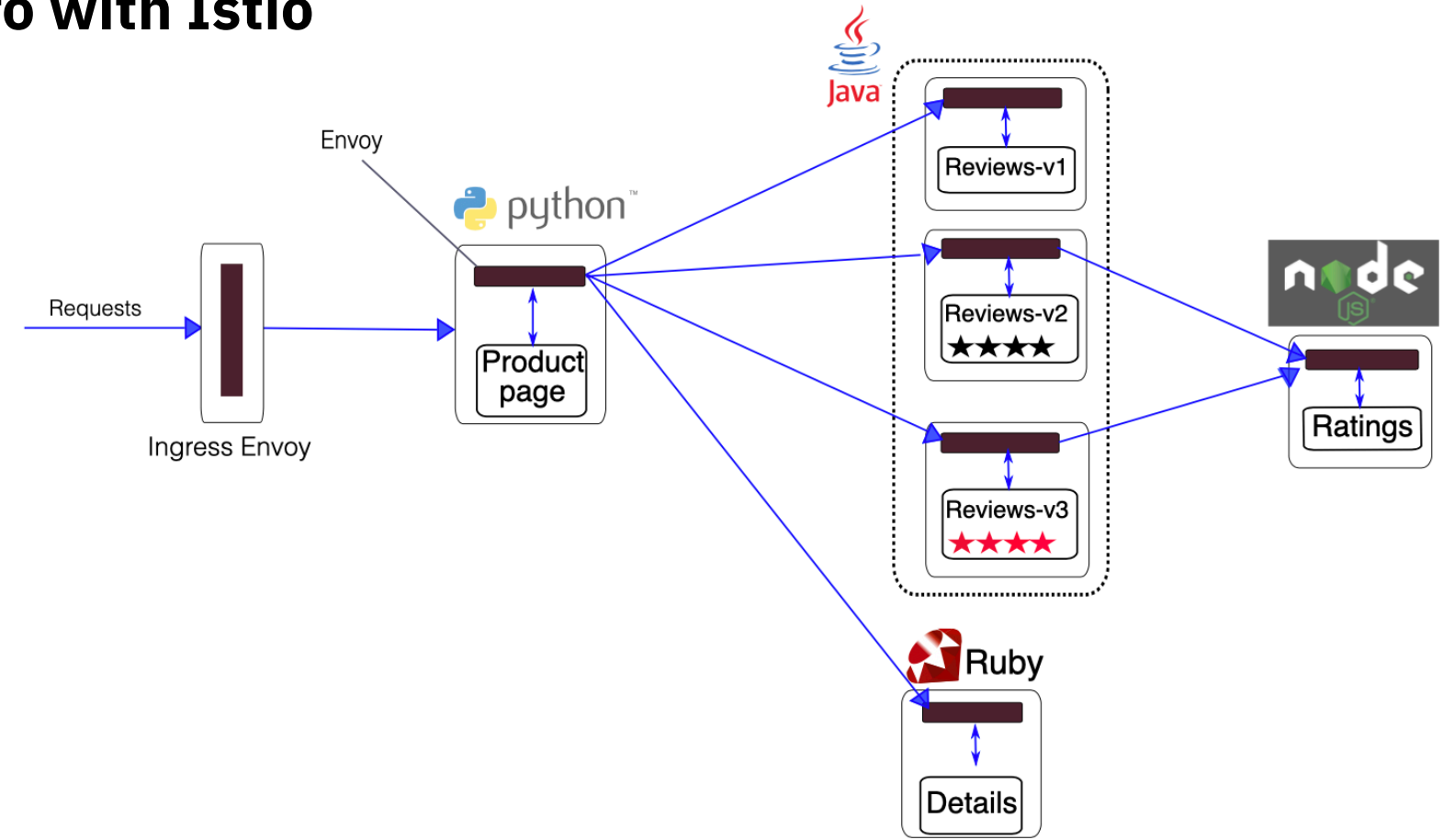
- Service authentication
- Authorization
- Rate limits
- Load shedding
- Telemetry
- Request Tracing
- Fault Injection

Bookinfo



Bookinfo Application without Istio

Bookinfo with Istio



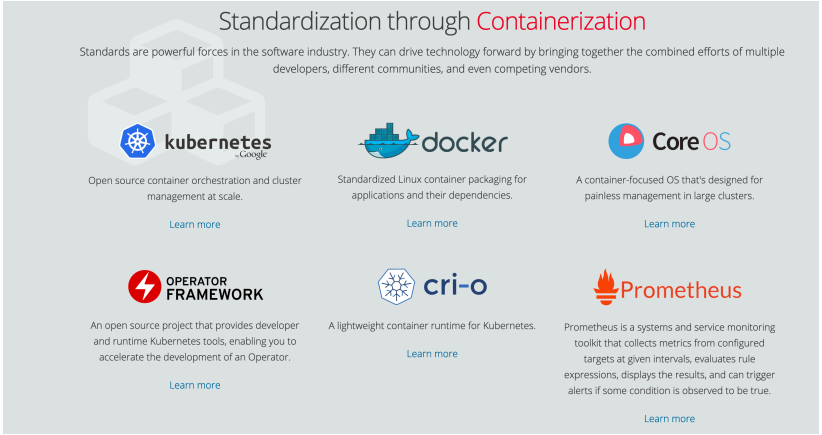
Bookinfo Application

ORIGIN COMMUNITY DISTRIBUTION OF KUBERNETES

okd

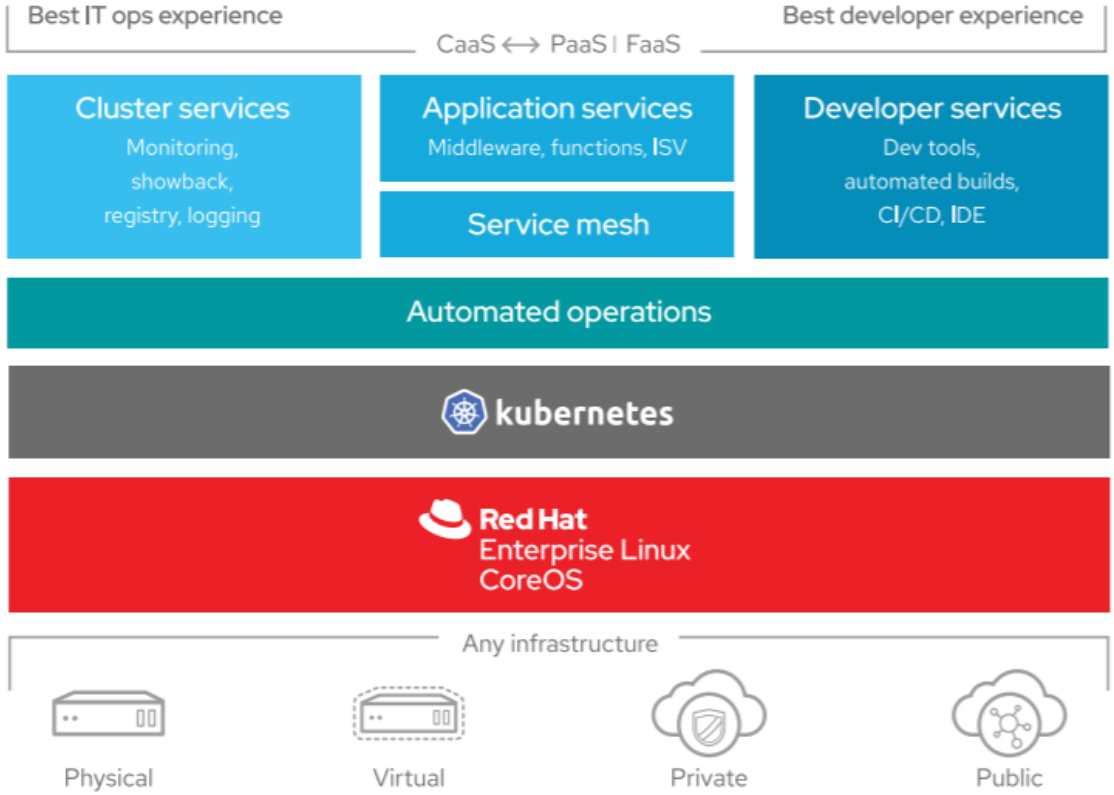
The Origin Community Distribution of Kubernetes that powers Red Hat OpenShift.

Built around a core of OCI container packaging and Kubernetes container cluster management, OKD is also augmented by application lifecycle management functionality and DevOps tooling. OKD provides a complete open source container application platform.

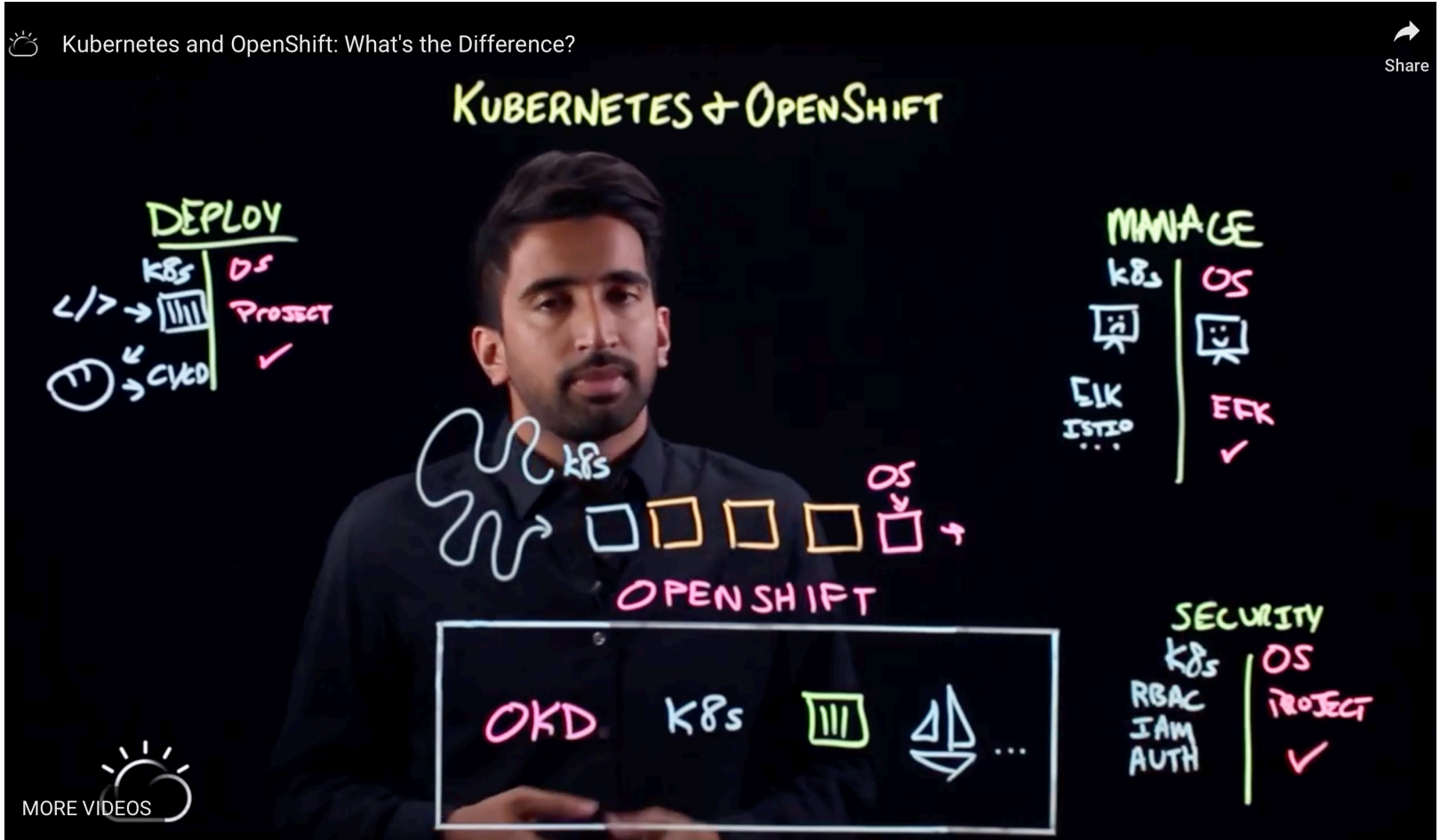


Open Source | Secure | Fully Certified | Portable

RED HAT® OPENSHIFT



#IBMDeveloper


Kubernetes and OpenShift: What's the Difference?
Share
KUBERNETES & OPENSHIFT
DEPLOY
k8s
OS
PROJECT
CI/CD
MANAGE
k8s
OS
ELK
ISTIO
EFK
k8s
OS
OPENSHIFT
OKD
k8s
SECURITY
k8s
OS
RBAC
IAM
AUTH
PROJECT
MORE VIDEOS

LAB with Red Hat – something like this:

- https://medium.com/@blumareks/running-a-microservice-containers-on-kubernetes-straight-from-a-github-ff73047e877b

Thank you!

Marek Sadowski

Developer Advocate

@blumareks