

How to Run Kubernetes with OpenShift on IBM Cloud

Marek Sadowski

Developer Advocate | IBM San Francisco City Team

<http://ibm.biz/202007-k8s>

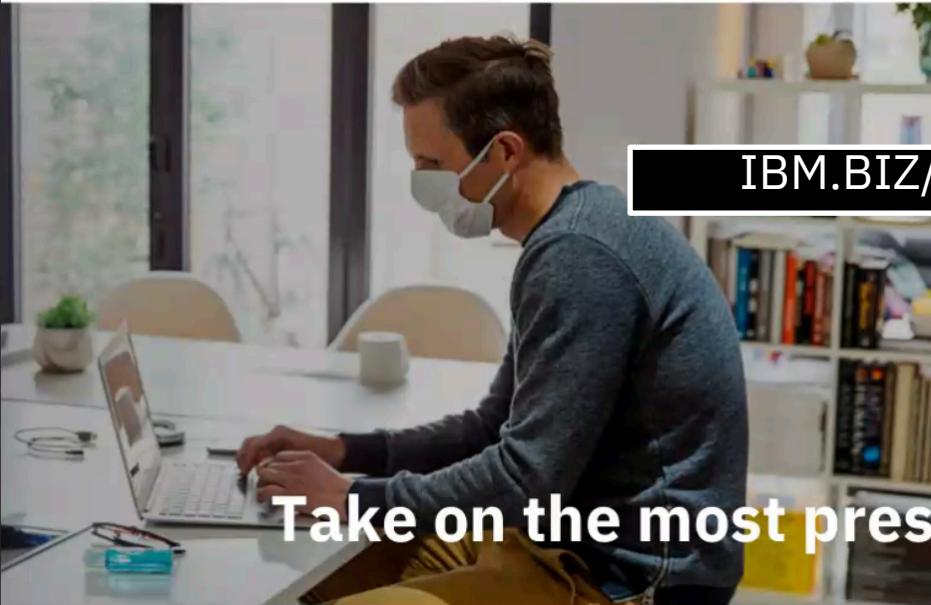
V1 2020 07 23

IBM Developer



2020 CALL FOR CODE®

Global Challenge



IBM.BIZ/callforcode

COVID-19 has revealed limits of the systems we take for granted, compromising our health, our planet, and our survival.

We've expanded the 2020 Call for Code Global Challenge to take on COVID-19, while we remain steadfastly committed to combating climate change.



Take on the most pressing issues of our time



Global Challenge

Participants:

Register for the challenge, and start building

Sponsors:

Show your full support with a sponsorship

COVID-19 Reference Materials:

We have the tools. You have the talent.

More information:

Find more details at ibm.biz/callforcode

Build solutions that fight back.

Join a **movement** with

- 200k+ devs, 165 nations, 8k+ apps!**

Help seize & reduce the impacts of the COVID-19 pandemic by addressing:

- Crisis Comms, Tele-edu, Communities**

Help halt & reverse climate change by addressing:

- Water, Energy Sustainability + Resiliency**

Have the chance to be **awarded**:

- \$200,000!**
- Support from Linux Foundation, C&R**

IBM.BIZ/callforcode



Call for Code
Founding Partner



Call for Code
Creator



Call for Code
Charitable Partner



Call for Code
Affiliate

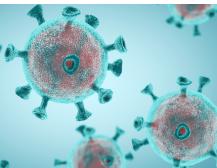
• Climate change

• COVID-19

Enter your COVID-19 solutions by April 27 for a chance to be featured as a top solution during IBM Think. Note: We will continue to accept COVID-19 submissions after April 27 until the Challenge deadline on July 31.

2020 Call for Code Global Challenge Timeline

February 26	Challenge kickoff	March 20	COVID-19 track added	April 27	Deadline for early COVID-19 submissions	May 5	Top 3 COVID-19 solutions announced at IBM Think	May 15	Initial deployment support starts for top 3 COVID-19 solutions	July 31	Final submission deadline for both tracks	October	Winners of the 2020 Call for Code Global Challenge announced at award ceremony
-------------	-------------------	----------	----------------------	----------	---	-------	---	--------	--	---------	---	---------	--



Agenda:

1. microservices in a cloud
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
10. Red Hat OpenShift hands-on

THE POLL

How you build a microservice

IT IS A SIMPLE PROGRAM – RESPONDING TO CRUD OPERATIONS

WHAT is a microservice?

Why are there microservices?

A large system might consist of many services: HR, a basket checkout, payments, CRM operations, etc.

These services might be built out of microservices – change the prescription's content: add, remove, update, delete

- These are microservices – that might be deployed as containers

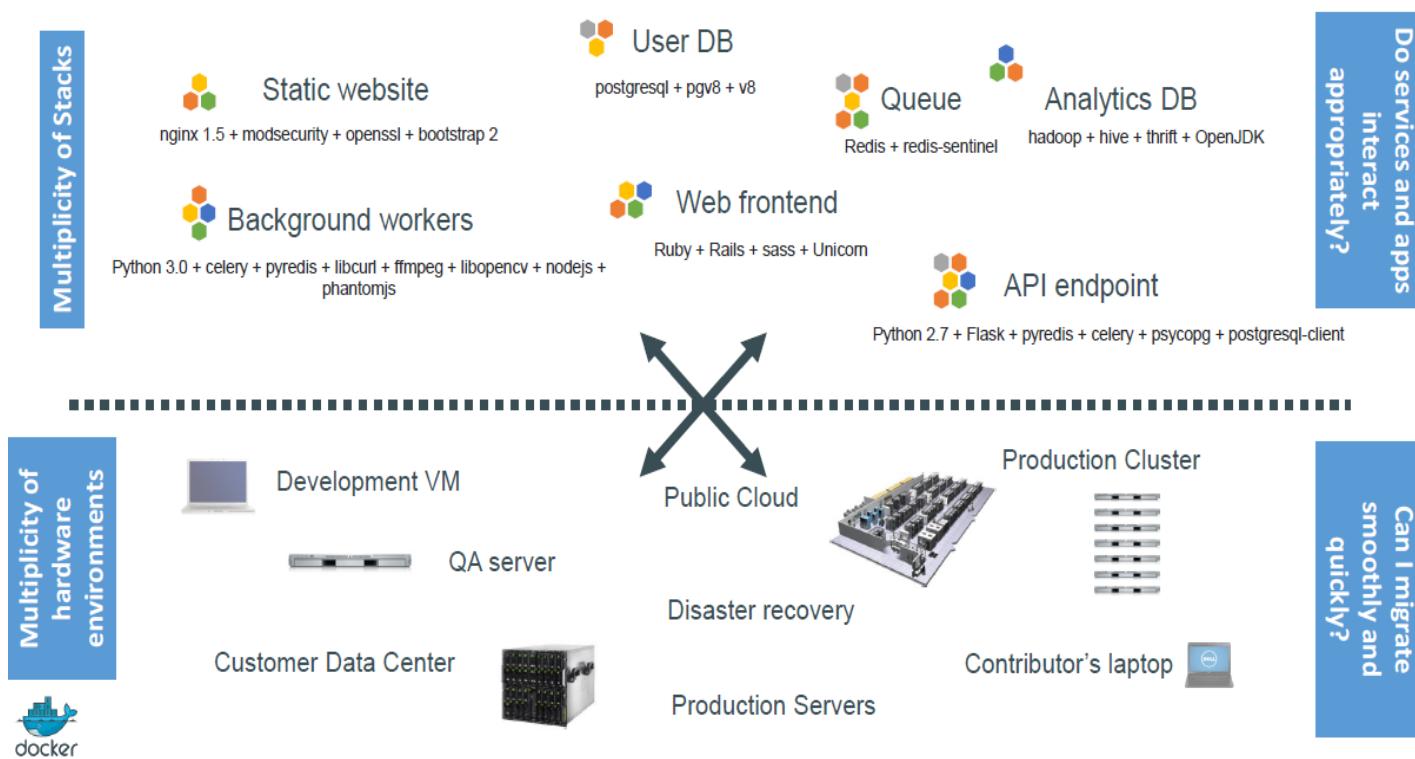
Why Containers?

Everyone Loves Containers

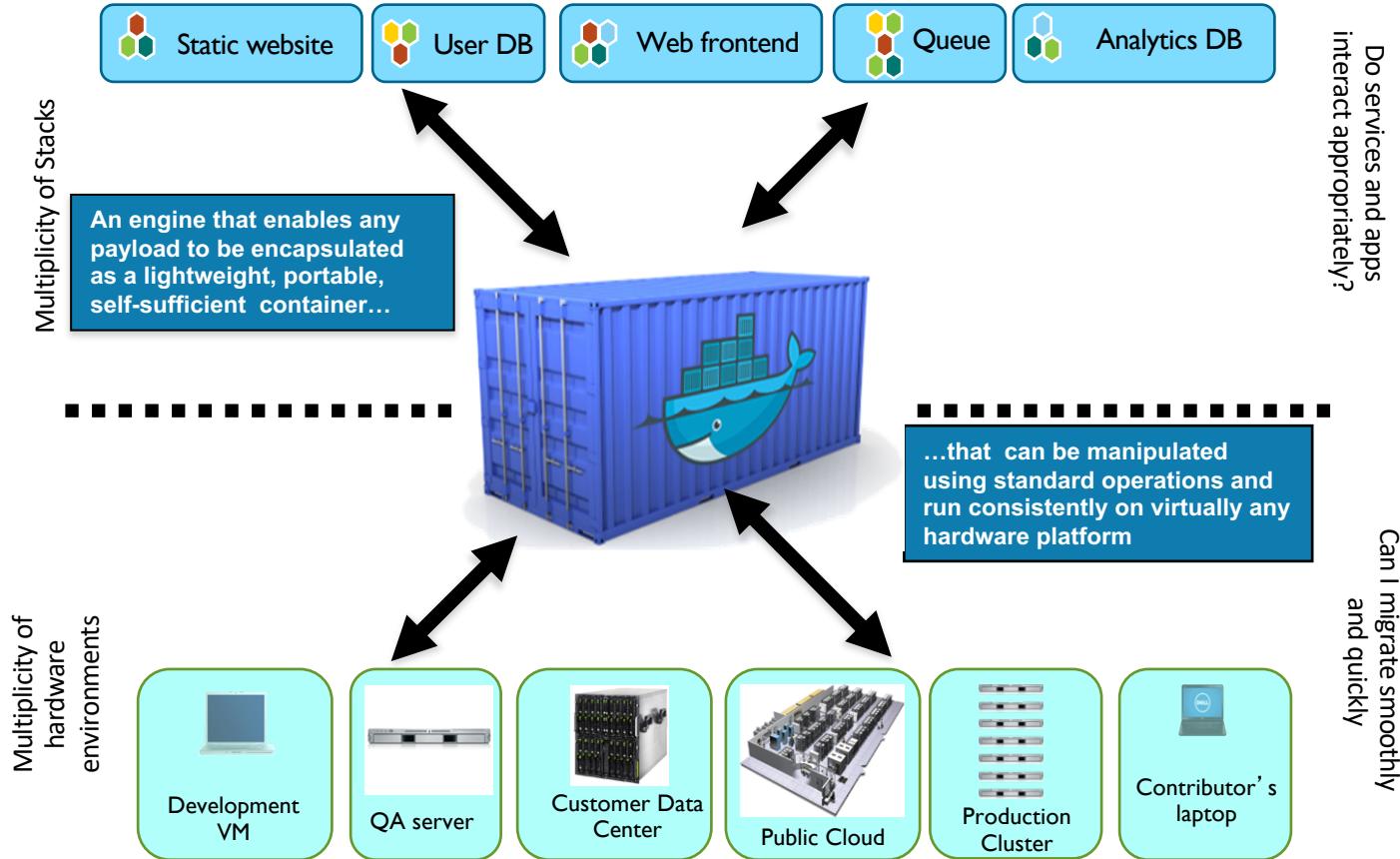


- A standard way to package an application and all its dependencies so that it can be moved between environments and run without changes
- Containers work by isolating the differences between applications inside the container so that everything outside the container can be standardized

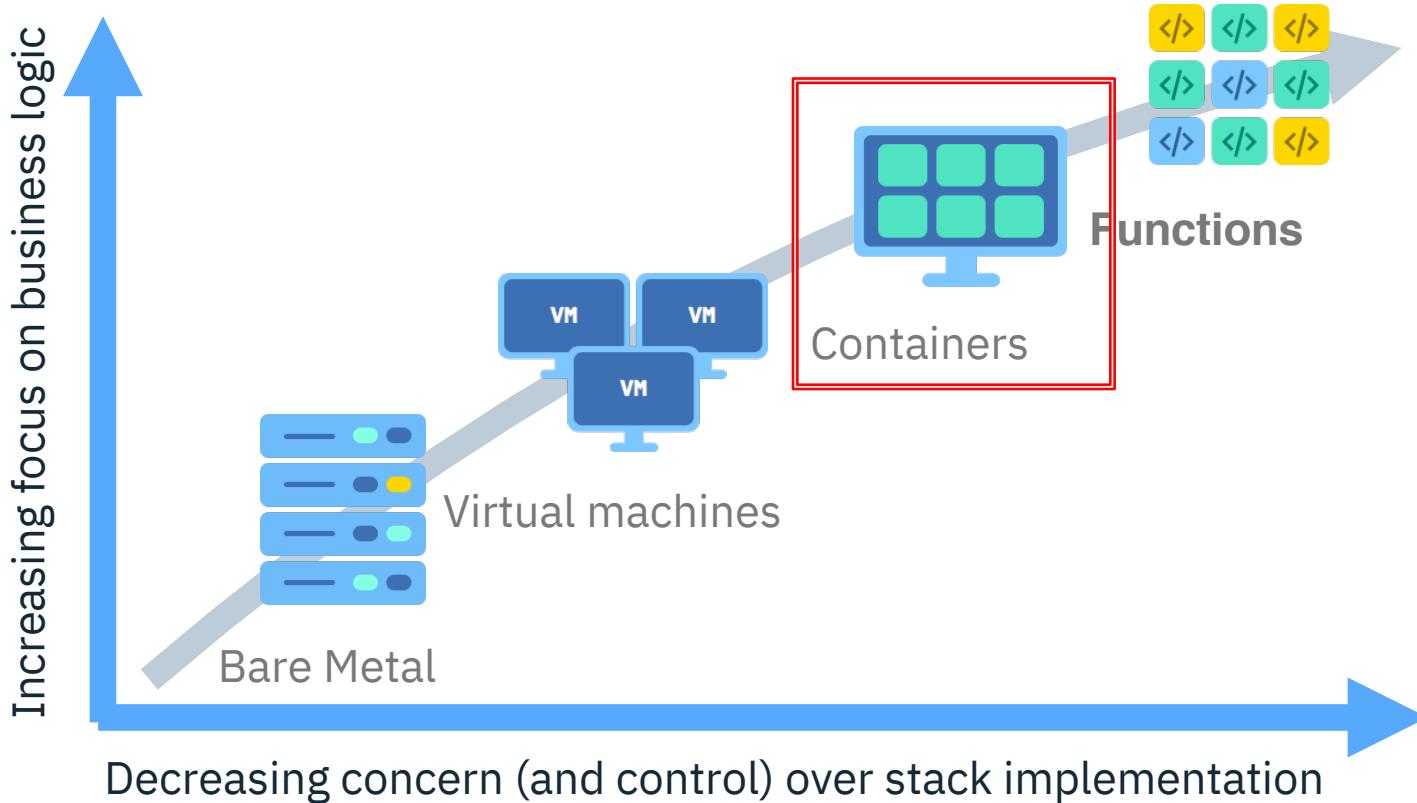
The Challenge



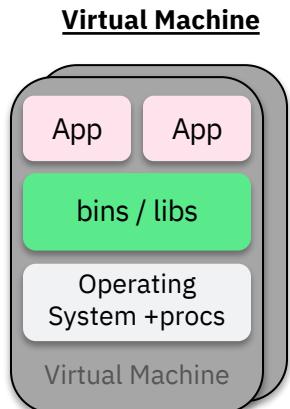
The Solution - A Shipping Container for Code



How we got here



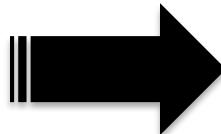
VM vs Container



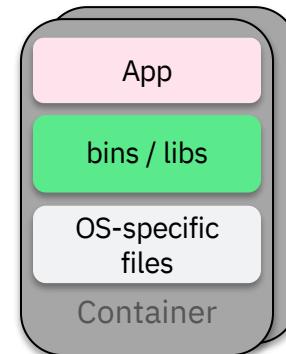
Hypervisor

Hardware

Each VM has its own
OS



Container



Base OS/Kernel

Hardware

App, bins/libs/OS must all be
runnable on the shared kernel

If OS files aren't needed they can be
excluded.

VM ?

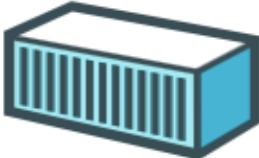
Containers share
the same base
Kernel

Docker Mission

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



Build



Ship



Run



IBM Code



Any App

Anywhere

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

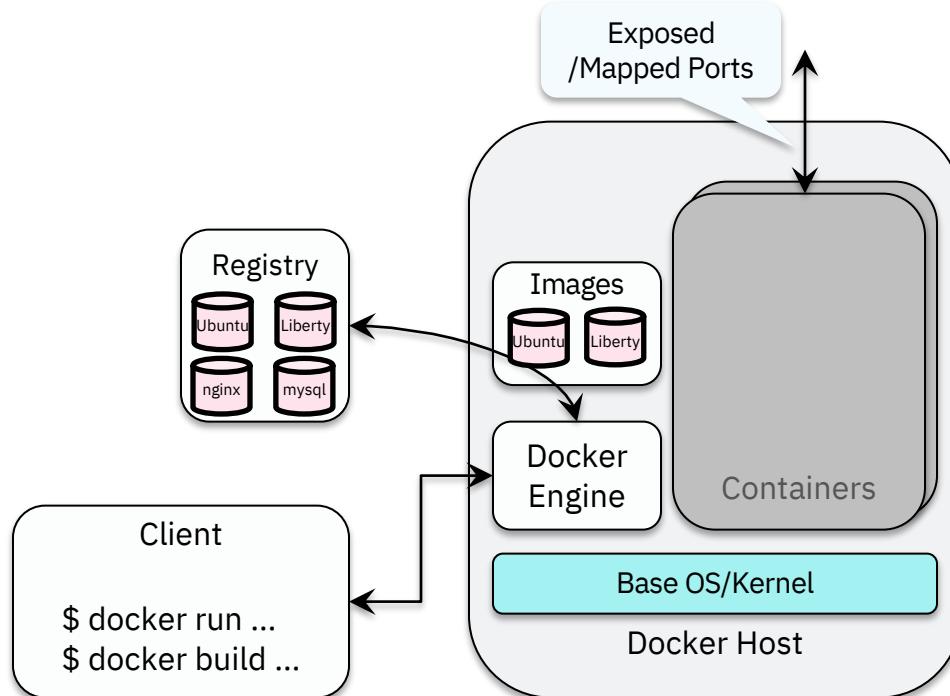
Images

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

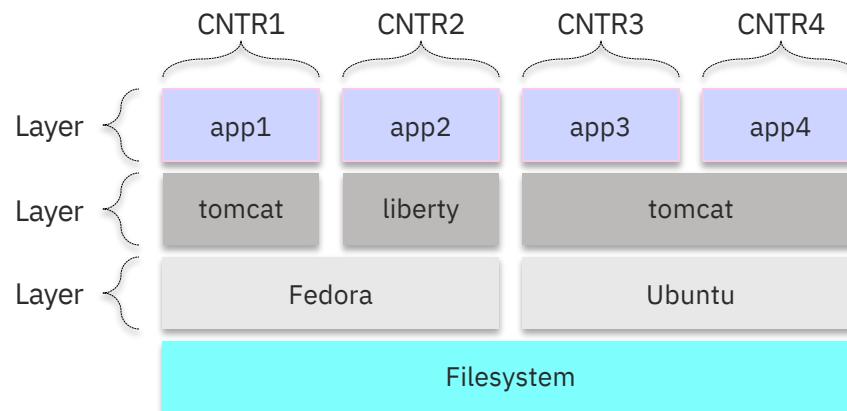
Layers allow for reuse

- More containers per host
- Faster start-up/download time

Images

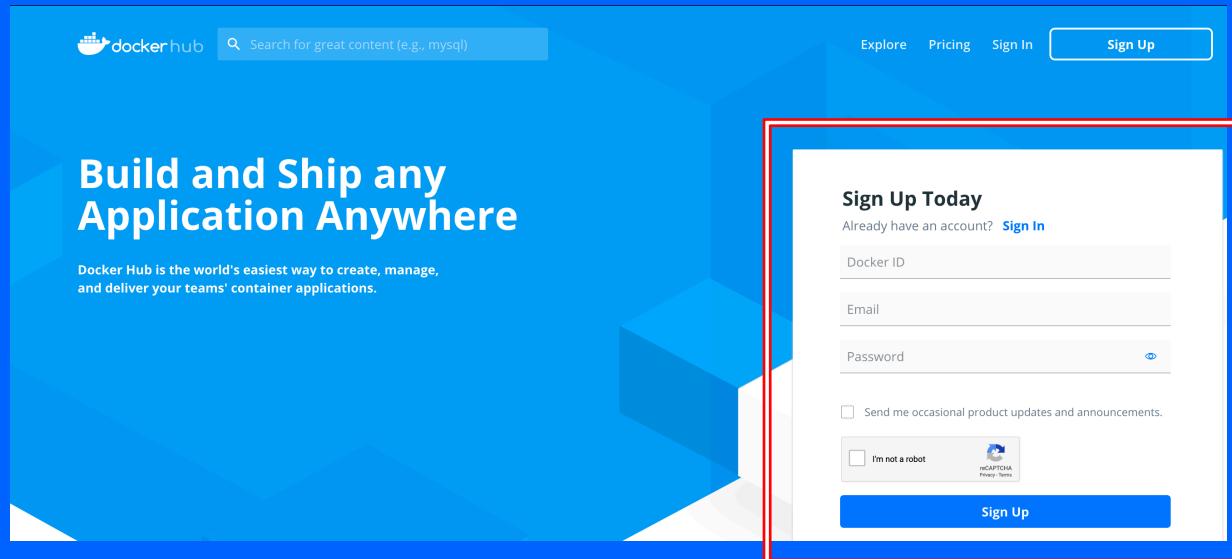
- Tarball of layers

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        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

LAB
building a translation service with Watson:

<http://ibm.biz/202007-k8s-docker>

IBM Cloud Catalog

<https://cloud.ibm.com/catalog>

The screenshot shows the IBM Cloud Catalog homepage. At the top, there's a navigation bar with links for Catalog, Docs, Log in, and Sign up. Below the navigation is a search bar labeled "Search the catalog...". A horizontal menu bar has two tabs: "Services" (which is highlighted with a blue underline) and "Software". To the left, a sidebar lists various service categories under "All Categories": VPC Infrastructure, Compute, Containers, Networking, Storage, AI, Analytics, Databases, Developer Tools, Integration, Internet of Things, and Security and Identity. On the right, the main content area is titled "Services" and describes the broad portfolio of managed services. It features a "Featured" section with a card for the "Kubernetes Service". The card includes a small icon of a cube, the service name, its category (IBM • Compute • Containers), and a brief description: "Deploy secure, highly available apps in a native Kubernetes experience." At the bottom of the card, it says "Free • IAM-enabled • Service Endpoint Supported". A vertical "FEEDBACK" button is located on the far right edge of the page.

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

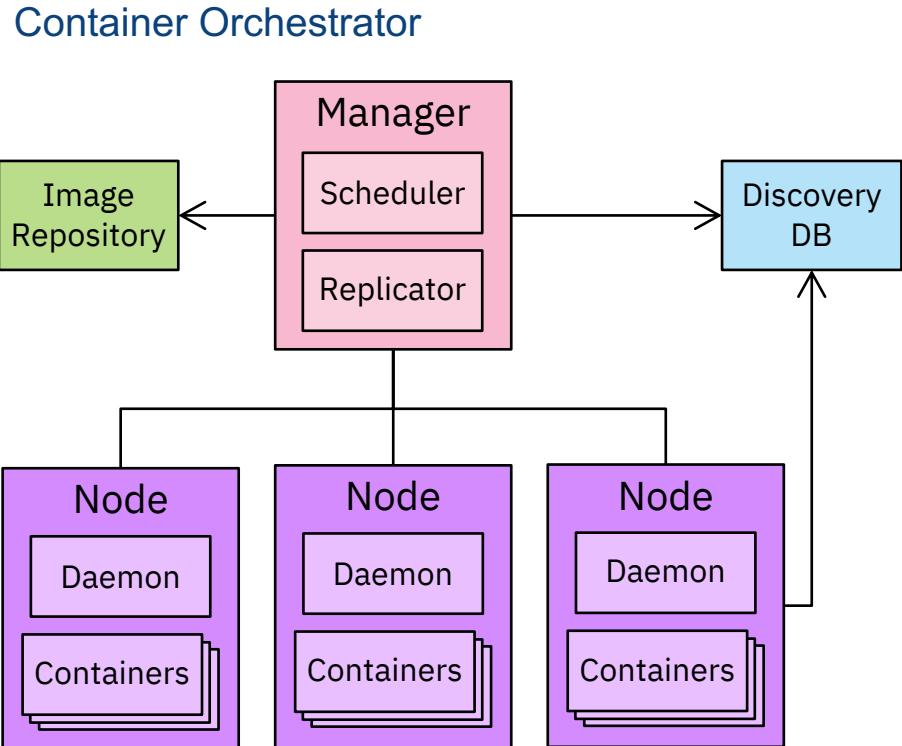
Container Orchestration

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

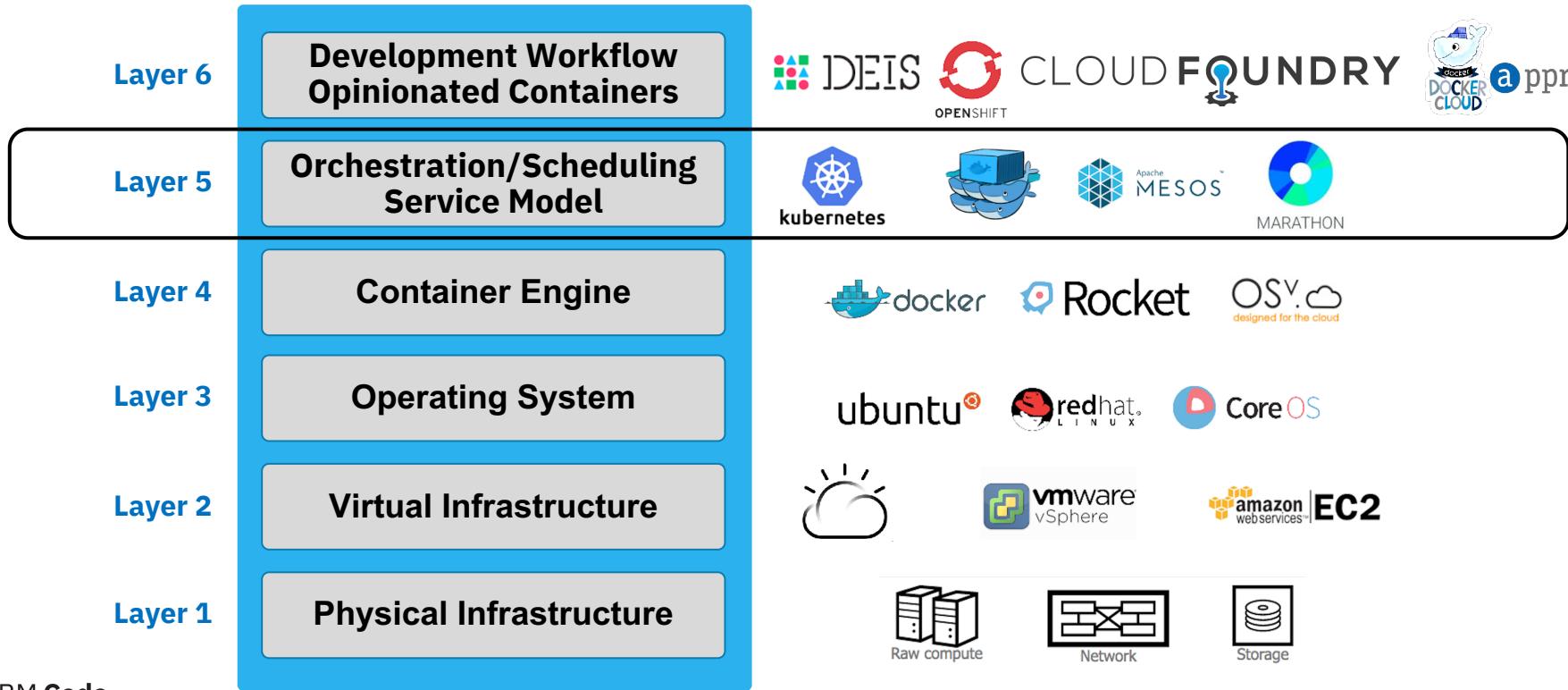
- Scheduling
- Cluster management
- Service discovery
- Provisioning
- Monitoring
- Configuration management

What is Container Orchestration?

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



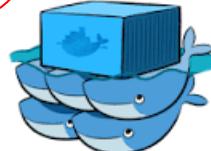
Container Ecosystem Layers



Kubernetes – Mesos – Marathon – Docker Swarm



Kubernetes is a cluster manager for containers (large deployments)



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)



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 built to be run on mesos.

(Kubernetes can be run on Mesos)



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?

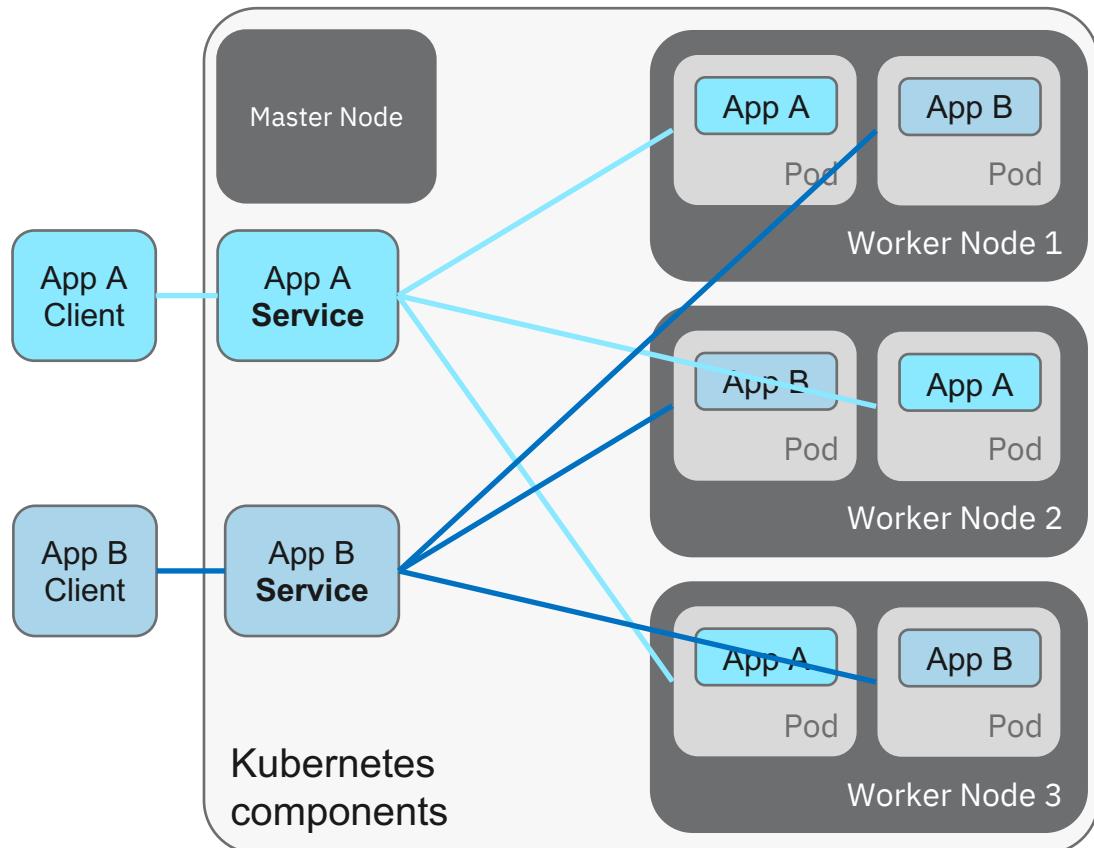


- Container orchestrator
- Manage applications, **not machines**
- Designed for extensibility
- Open source project managed by the Linux Foundation



Kubernetes Architecture: Workloads

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



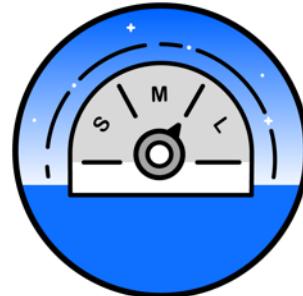
Kubernetes



Intelligent Scheduling



Self-healing



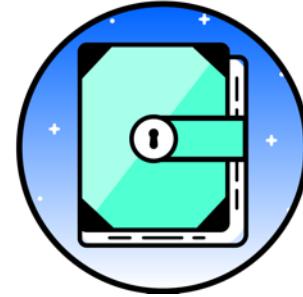
Horizontal scaling



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

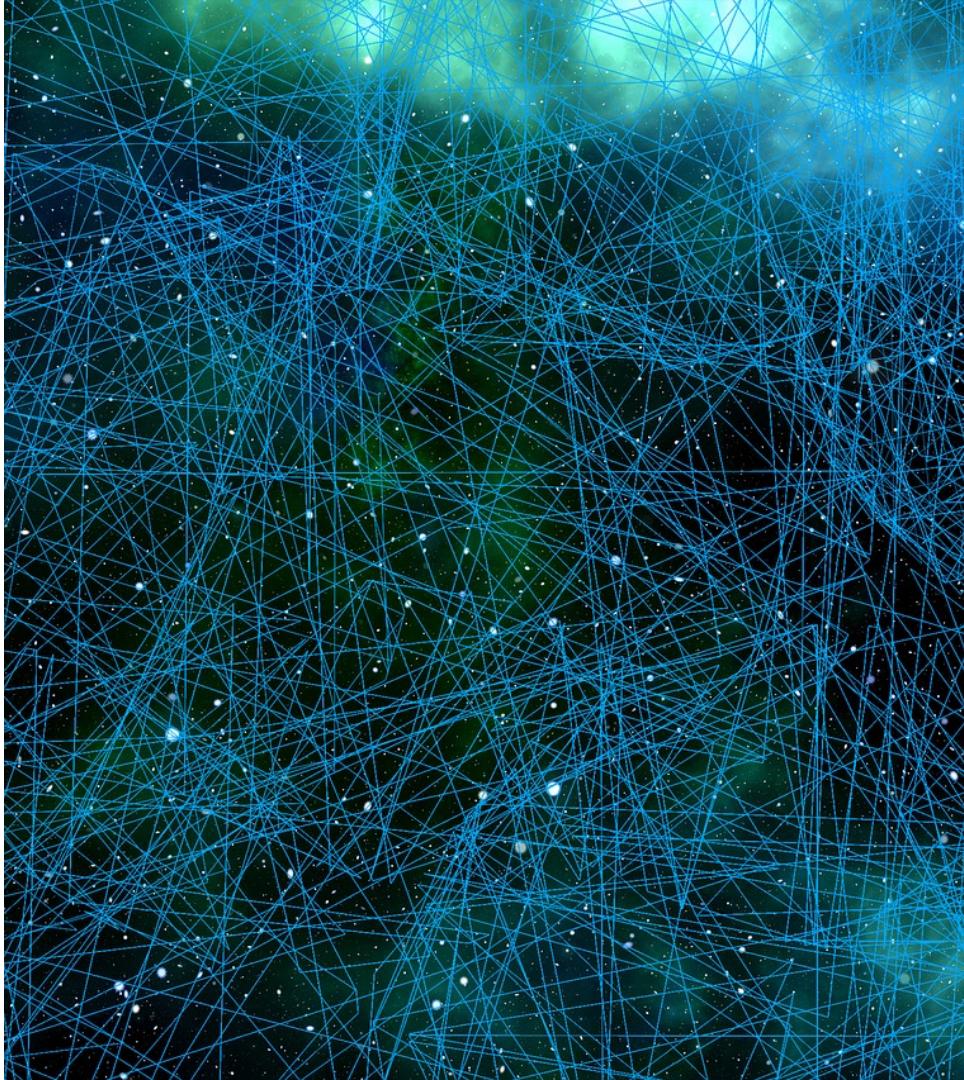
Retries and Circuit breaking

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>

Istio community partners

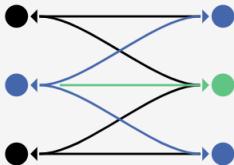
- RedHat
- Pivotal
- WeaveWorks
- Tigera
- Datawire
- Scytale (SPIFFE)
- Microsoft
- Uber (Jaeger)





Istio

Connect, secure, control, and observe services.



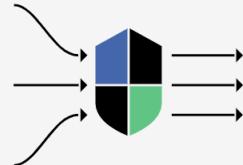
Connect

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



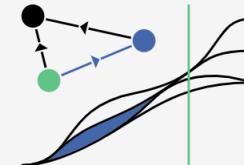
Secure

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.

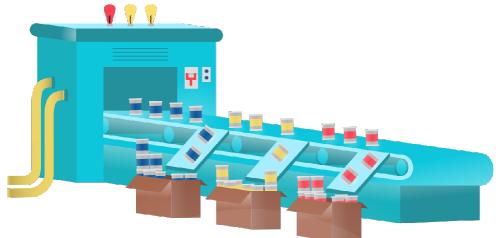


Observe

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

WHERE ISTIO FINDS ITS PLACE

Intelligent
Routing and
Load Balancing



Resiliency
across
Languages and
Platforms



Fleet Wide
Policy
Enforcement



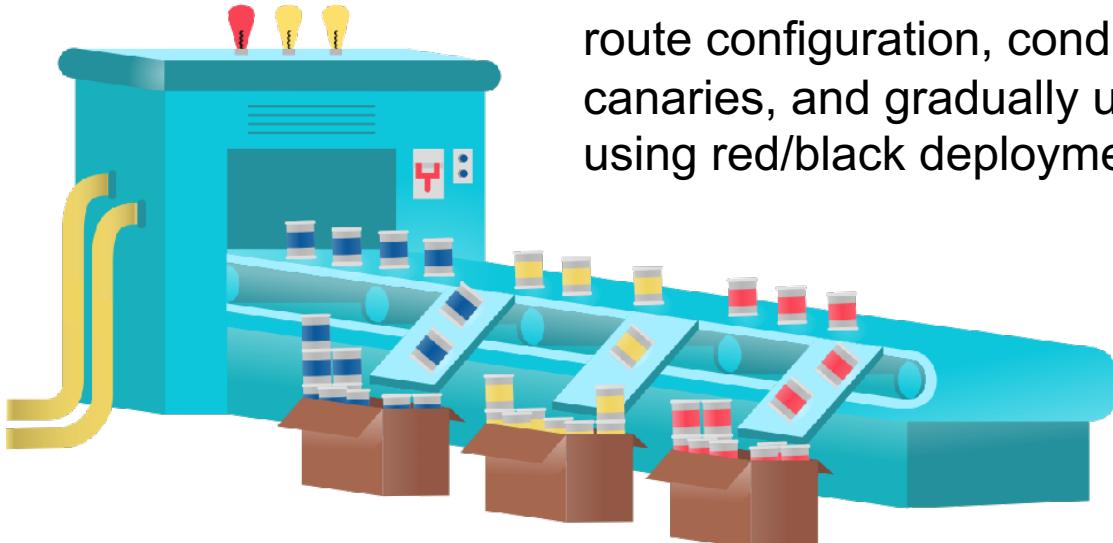
In-Depth
Telemetry and
Reporting



WHERE ISTIO FINDS ITS PLACE

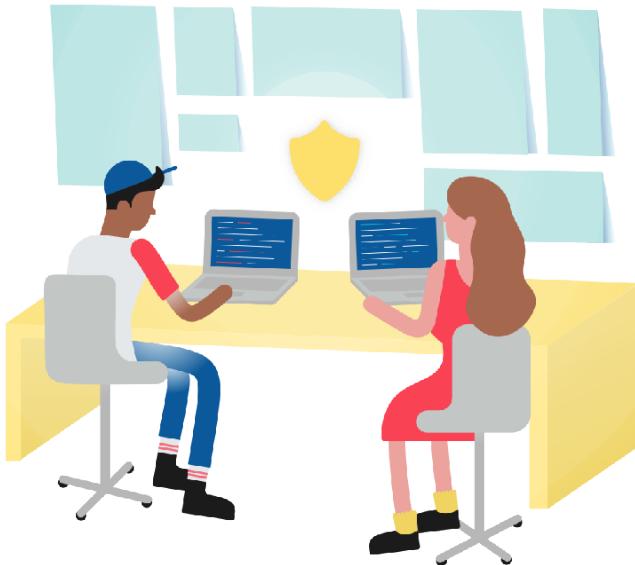
Intelligent Routing and Load Balancing

Control traffic between services with dynamic route configuration, conduct A/B tests, release canaries, and gradually upgrade versions using red/black deployments.



WHERE ISTIO FINDS ITS PLACE

Resiliency across Languages and Platforms

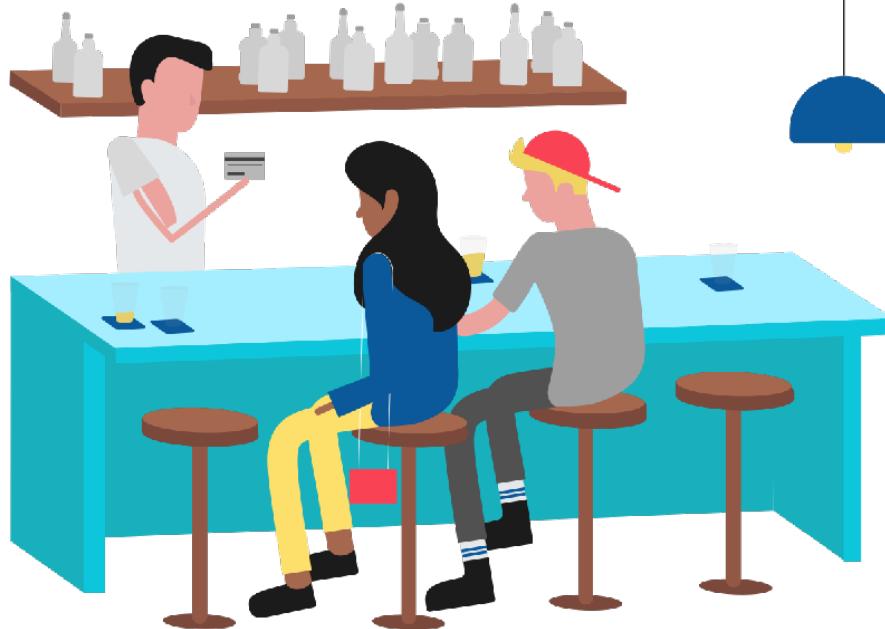


Increase reliability by shielding applications from flaky networks and cascading failures in adverse conditions.



WHERE ISTIO FINDS ITS PLACE

Fleet Wide Policy Enforcement

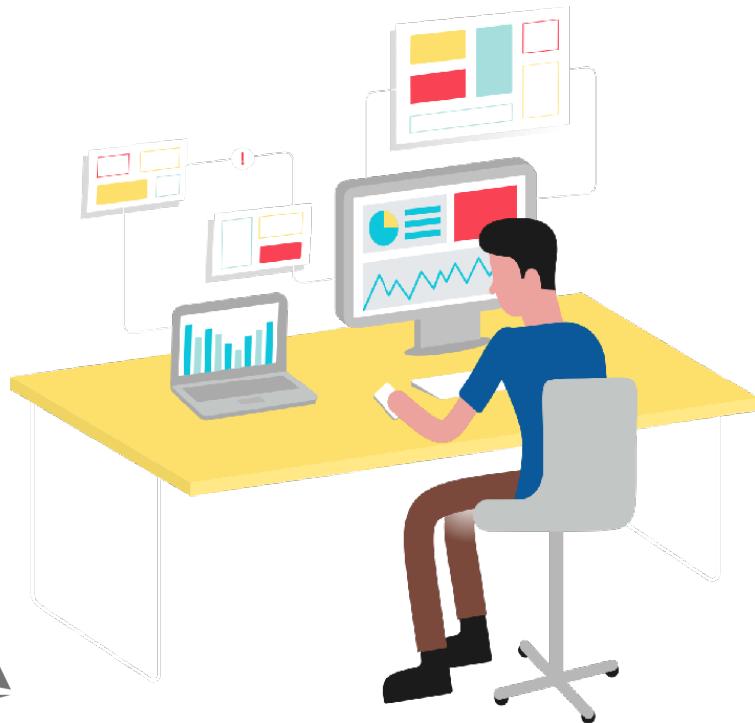


Apply organizational policy to the interaction between services, ensure access policies are enforced and resources are fairly distributed among consumers.

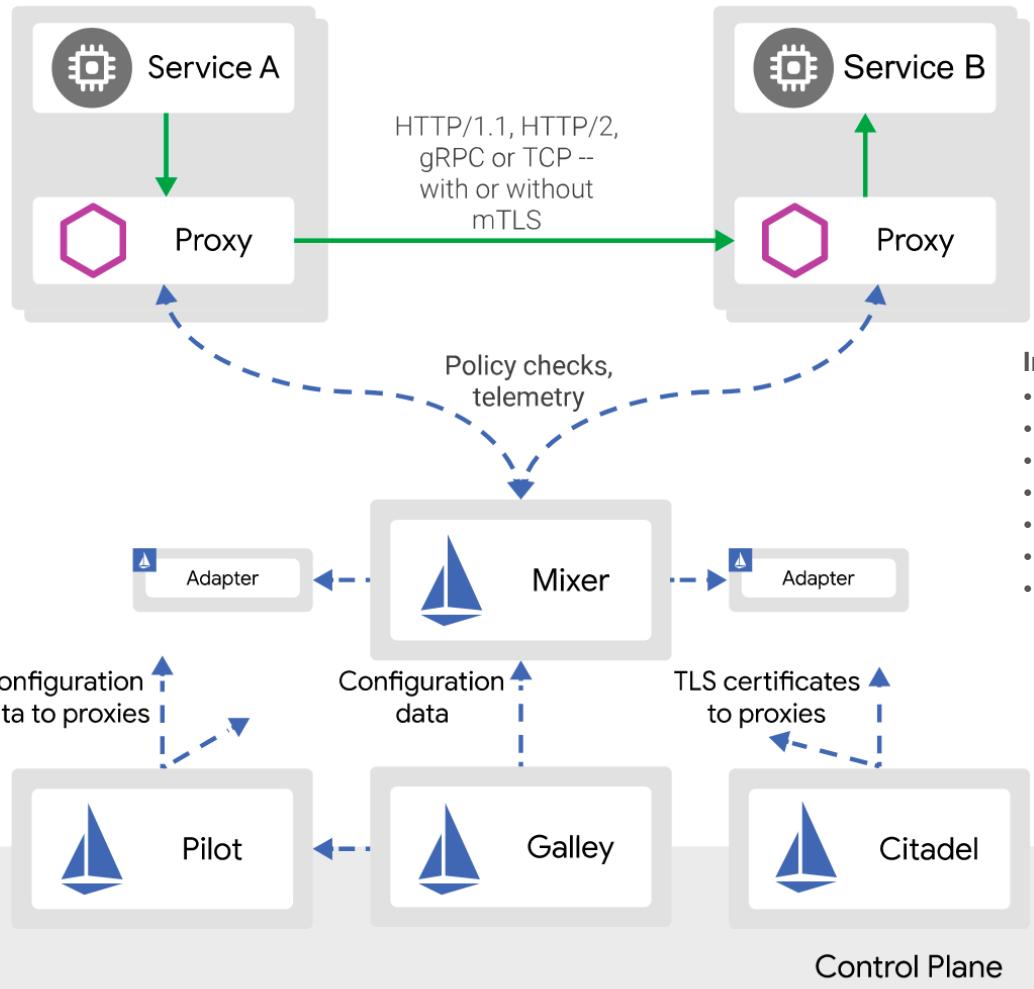


WHERE ISTIO FINDS ITS PLACE

In-Depth Telemetry and Reporting



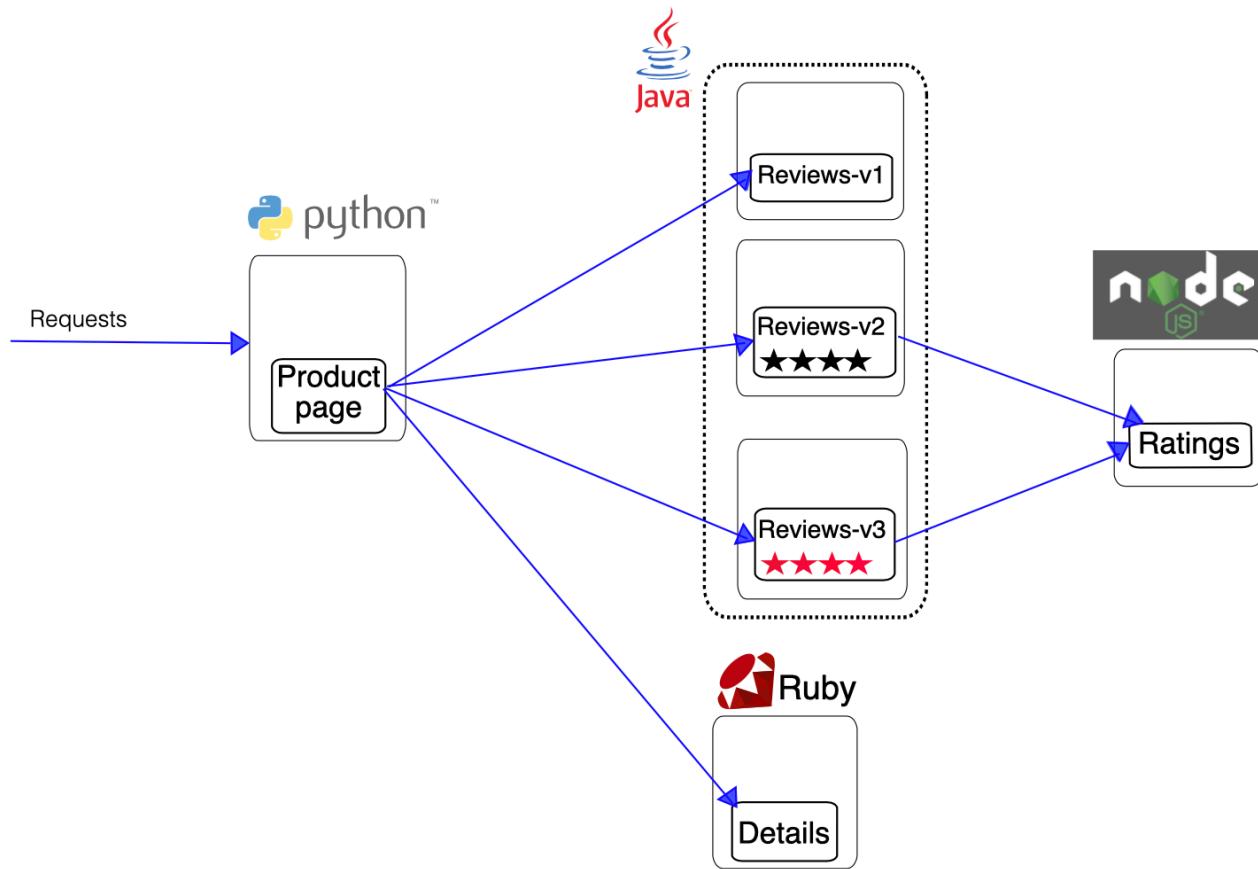
Understand the dependencies between services, the nature and flow of traffic between them and quickly identify issues with distributed tracing.



What is Envoy

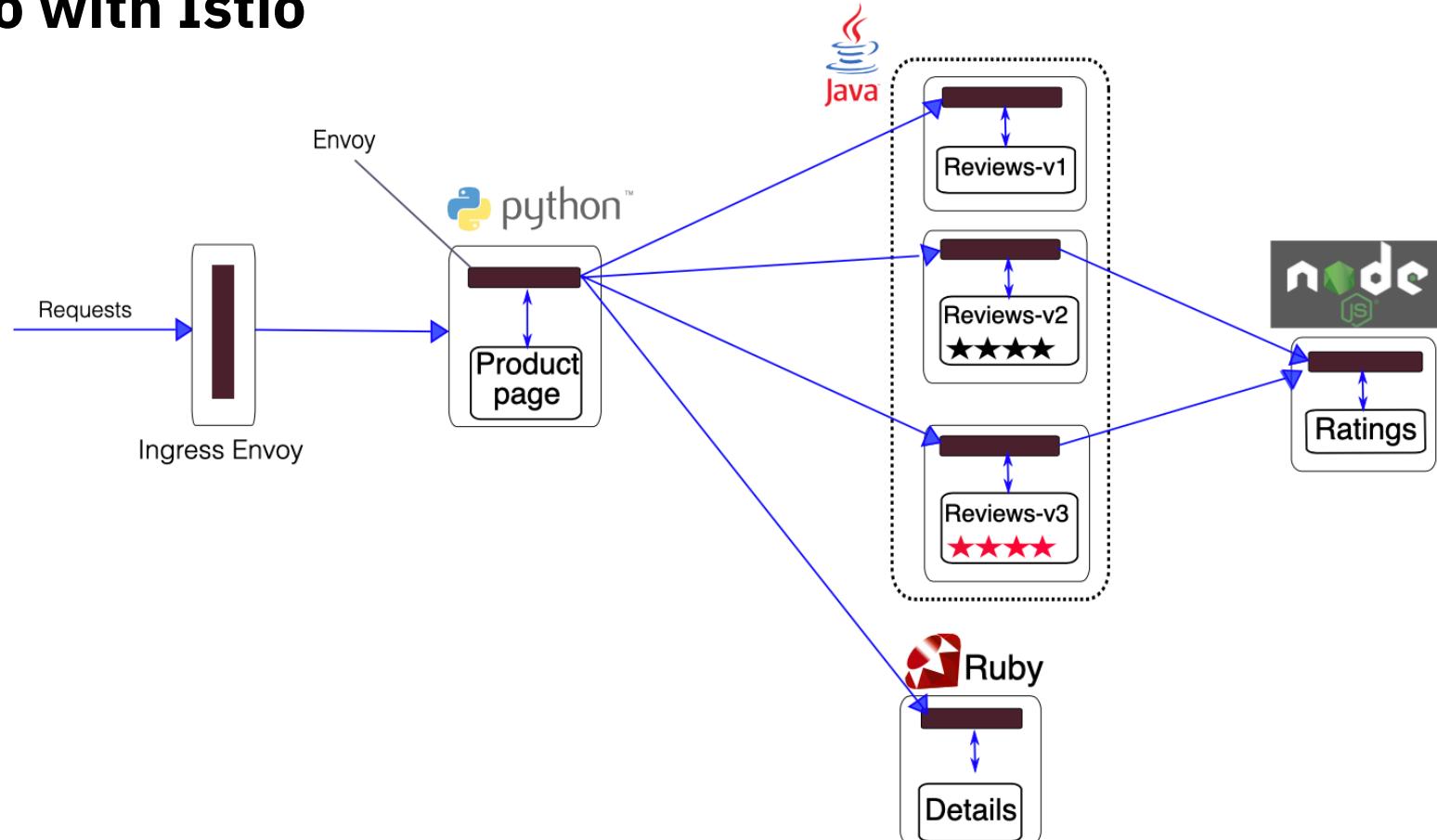
- **Out of process architecture:** Let's do a lot of really hard stuff in one place and allow application developers to focus on business logic.
- **Modern C++11 code base:** Fast and productive.
- **L3/L4 filter architecture:** A TCP proxy at its core. Can be used for things other than HTTP (e.g., MongoDB, redis, stunnel replacement, TCP rate limiter, etc.).
- **HTTP L7 filter architecture:** Make it easy to plug in different functionality.
- **HTTP/2 first!** (Including gRPC and a nifty gRPC HTTP/1.1 bridge).
- **Service discovery and active health checking.**
- **Advanced load balancing:** Retry, timeouts, circuit breaking, rate limiting, shadowing, etc.
- Best in class **observability:** stats, logging, and tracing.
- **Edge proxy:** routing and TLS.

Bookinfo



Bookinfo Application without Istio

Bookinfo with Istio



Bookinfo Application

ORIGIN COMMUNITY DISTRIBUTION OF KUBERNETES



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.

Standardization through Containerization

Standards are powerful forces in the software industry. They can drive technology forward by bringing together the combined efforts of multiple developers, different communities, and even competing vendors.



kubernetes
Google

Open source container orchestration and cluster management at scale.

[Learn more](#)



docker

Standardized Linux container packaging for applications and their dependencies.

[Learn more](#)



Core OS

A container-focused OS that's designed for painless management in large clusters.

[Learn more](#)



OPERATOR
FRAMEWORK

An open source project that provides developer and runtime Kubernetes tools, enabling you to accelerate the development of an Operator.

[Learn more](#)



cri-o

A lightweight container runtime for Kubernetes.

[Learn more](#)

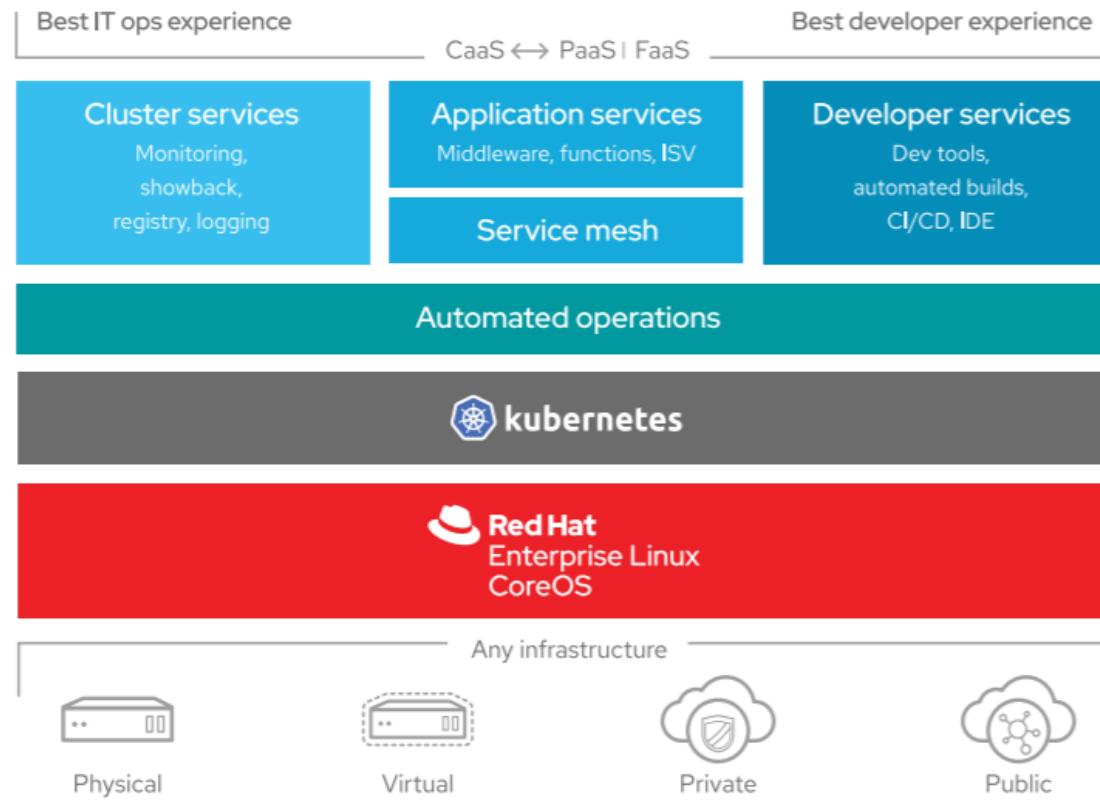


Prometheus

Prometheus is a systems and service monitoring toolkit that collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true.

[Learn more](#)

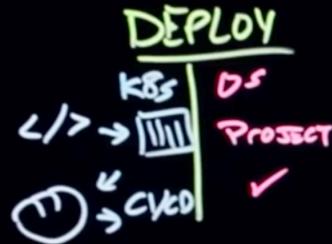
Open Source | Secure | Fully Certified | Portable



#IBMDeveloper



KUBERNETES + OPENSHIFT



Thank you!

Marek Sadowski

Developer Advocate

