Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Informatik

CAS Cloud and Platform Management (CAS CPM 1)

Der Weg in die Cloud
Modul 01 - Introduction

Niklaus Hirt
Cloud and DevOps Architect

nikh@ch.ibm.com

Rotkreuz, 06.12.2019

Who am I?

Niklaus Hirt

Passionate about tech for over 35 years

- High-school in Berne
- Degree in Computer Science at EPFL
- ELCA
- CAST
- IBM

nikh@ch.ibm.com

@nhirt

Agenda - Kubernetes Basic Concepts

Modules

Module 0: Prepare the Labs

Module 1: Microservices

Module 2: Containers with Docker

Module 3: Kubernetes

Module 4: Let’s get real

BREAK

Module 5: Docker Hands-On

LUNCH – 12:30

Agenda - Kubernetes Advanced Concepts

Modules

Module 6: Kubernetes Hands-On

BREAK

Module 7: Kubernetes Security

oder

Module 8: Mesh Networking with ISTIO

END of course – ca 16:15

Sources and documentation available here:

https://github.com/niklaushirt/k8s_training_public

https://github.com/niklaushirt/training

The Journey to Cloud
Prepare the Labs

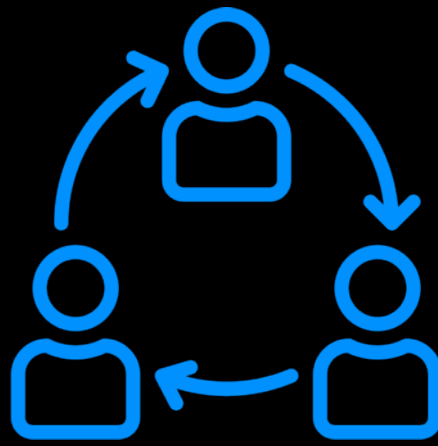
00

IBM **Cloud**

Session Objectives

Attendees will be grouped in ***teams*** wherever it makes sense to facilitate collaborative work.

Following some lectures will be ***hands-on*** work that each team collaborates to complete.



Teams

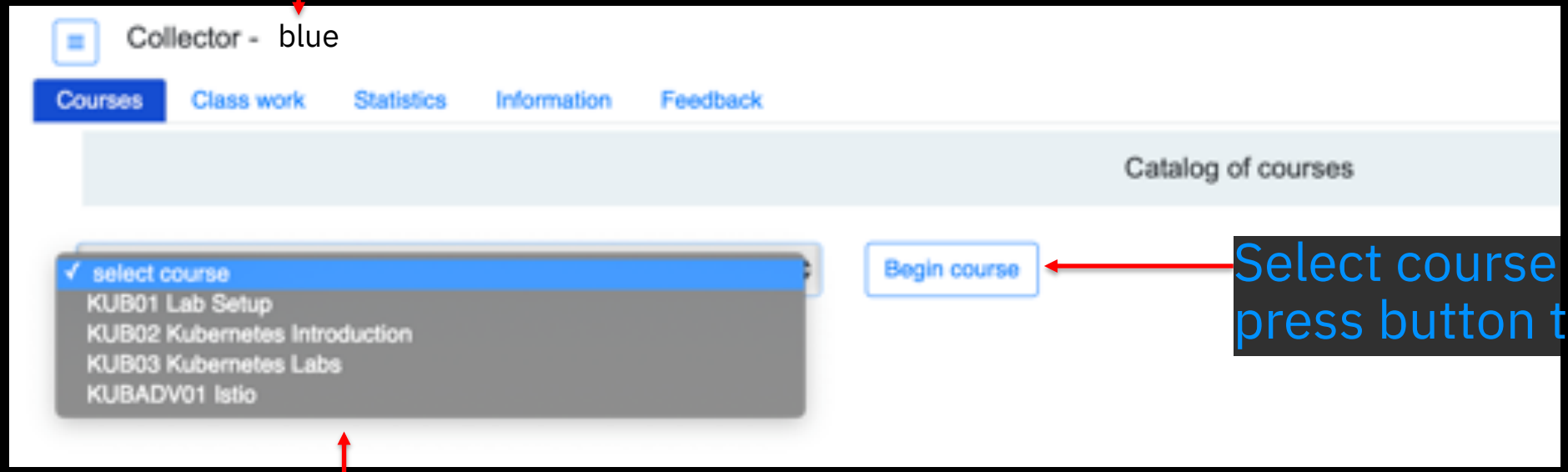
black 31701	olive 31711	peru 31715
white 31702	brown 31712	chocolate 31716
red 31703	lightblue 31713	orchid 31717
blue 31704	orange 31708	gold 31718
yellow 31705	purple 31709	pink 31719
lime 31706	maroon 31710	violet 31720
cyan 31707	firebrick 31714	

Collector - Accessing team web site

http://158.177.137.195:{port#}

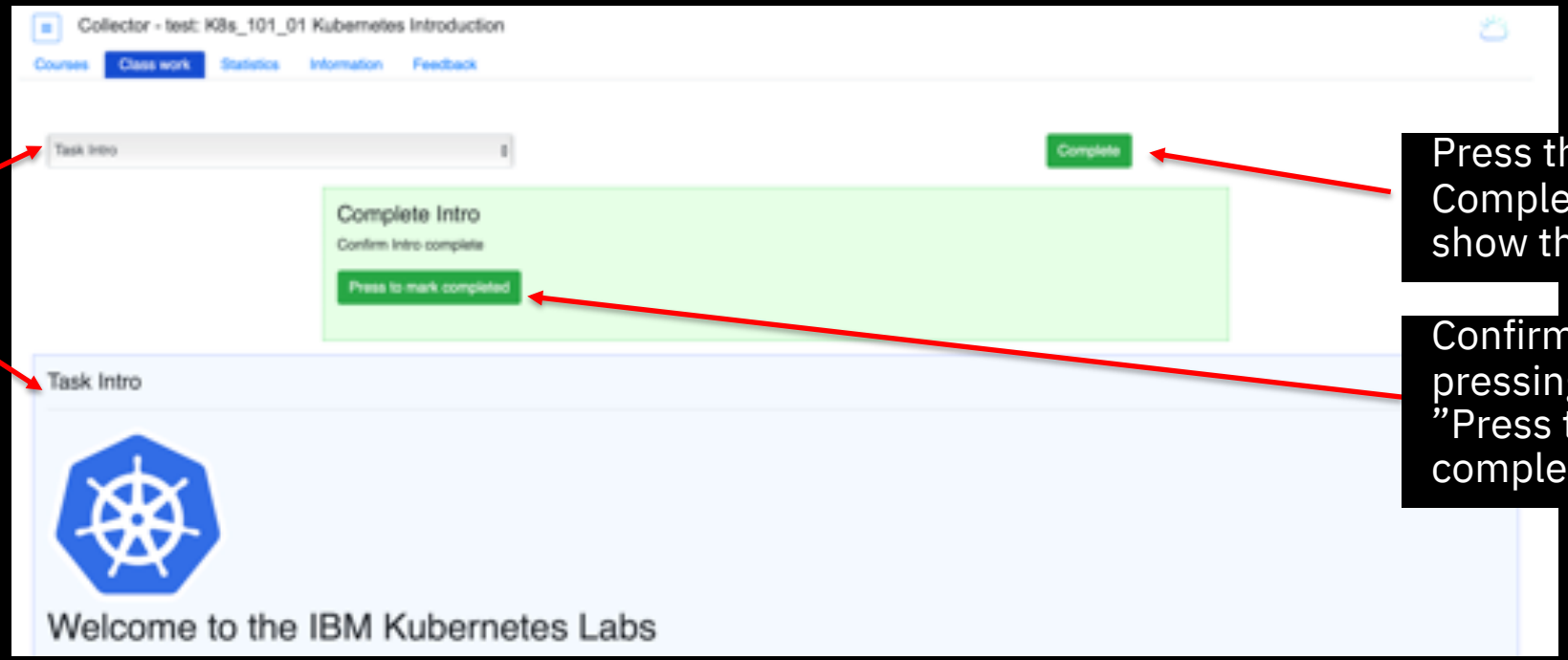
blue 31704

Team name / color will be shown



Collector – Class work

Select class work and the blue portion of the screen is shown



Press the green Complete button to show the green portion.

Confirm completion by pressing the green ”Press to mark completed” button.

! The Complete Button might not show instantly depending on the course settings

Collector – Track course completed work

Course title

The number of items tracked will change based on the current course selected.

Green checkmark - item is completed
Red circle - item is waiting to be completed

Collector – Instructor Dashboard

Remaining Time for the Lab

JTC90 Lab Setup

Task 1: Download Training VM

Task 2: Setup VirtualBox >6.14

Task 3: Start Training VM

Task 4: Login / Check

JTC90 Lab Setup

Task 1: Download Training VM

Task 2: Setup VirtualBox >6.14

Task 3: Start Training VM

Task 4: Login / Check

EVERYBODY OK ?

IBM

The Journey to Cloud
Microservices

01

IBM **Cloud**

Kodak

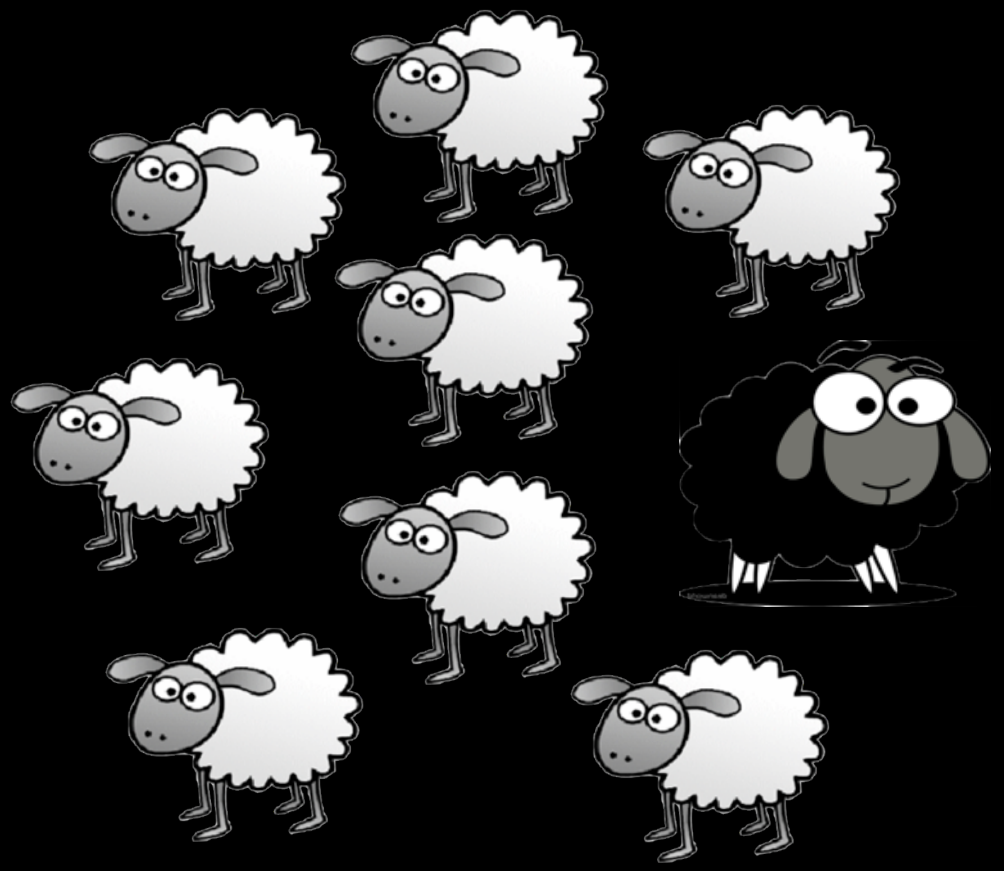
disruption

dɪsˈrʌpʃn/

noun

Business. a **radical change** in an industry, business strategy, etc., especially involving the introduction of a **new product or service** that creates a **new market**

Disruption is new reality



disruption

dɪsˈrʌpʃn/

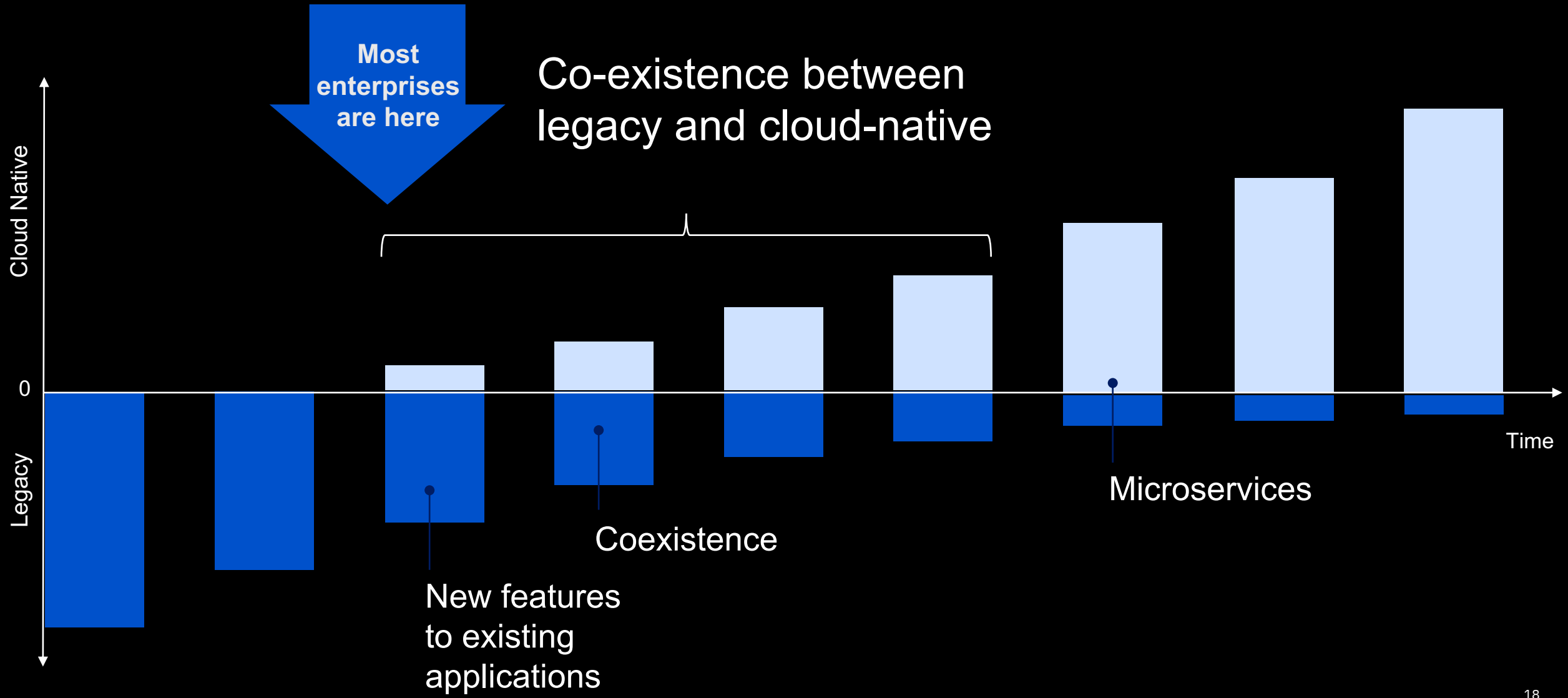
noun

Business. a **radical change** in an industry, business strategy, etc., especially involving the introduction of a **new product or service** that creates a **new market**

Digital Transformation is the new normal

Digital Transformation - Way to go

Cloud native and legacy apps will co-exist for the next 10+ years



How do you achieve digital transformation?

Adopt new **Processes**	Adopt new **Technology**	Adopt new **Tools**	Adopt **Cloud**
Continuous Integration	Architectural patterns (Microservices)	Git, Github, Gitlab	Docker, Kubernetes
Continuous Build	Frameworks (Java MicroProfile, Spring ...)	Jenkins	Virtualization, VMs
Continuous Deploy	Node.js , Swift	UrbanCode	Monitoring, Dashboards, Alerts
Agile Dev Models		Docker	
		Kubernetes	

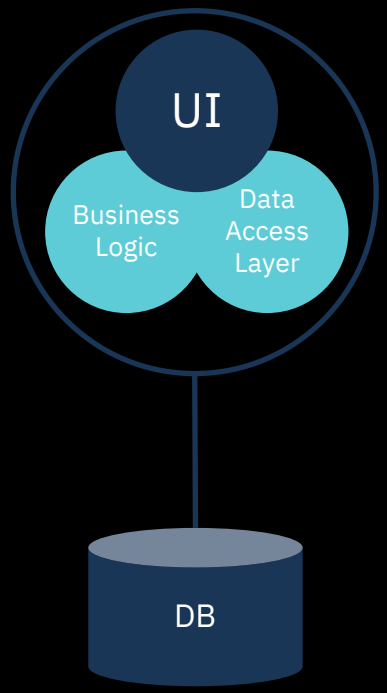
Microservices

Decomposing an application into **single function modules** which are **independently deployed and operated**

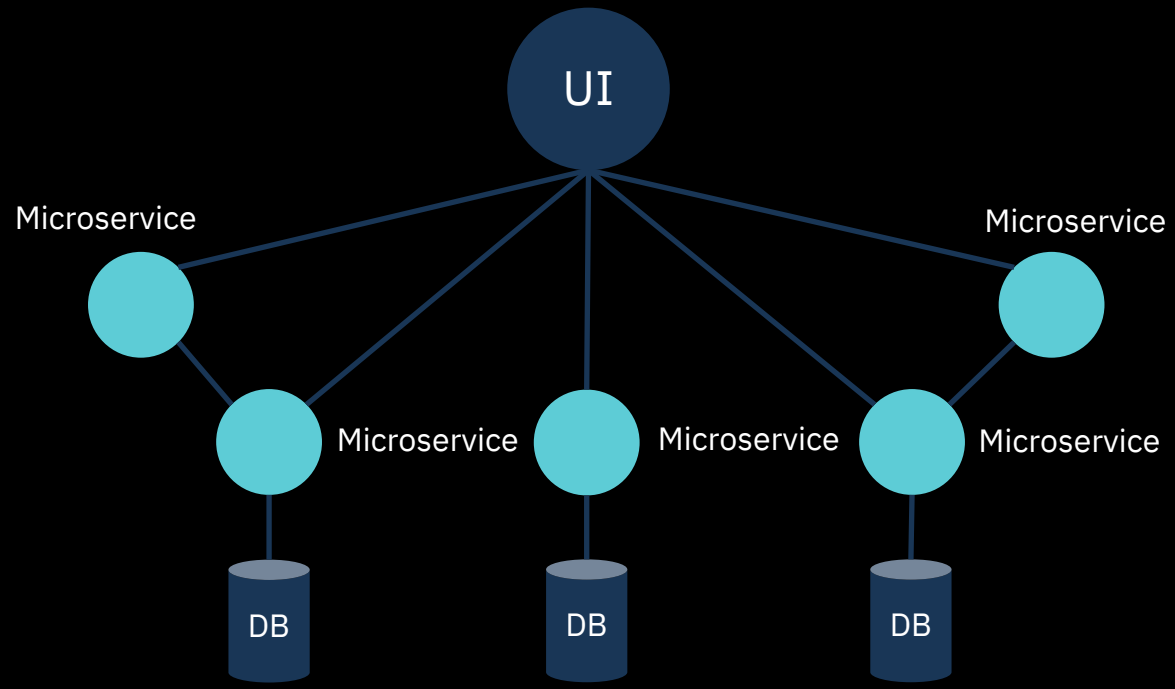
Accelerate delivery by minimizing communication and coordination between people

Microservices architecture

Simplistically, microservices architecture is about breaking down large silo applications into more manageable, fully decoupled pieces



Monolithic Architecture

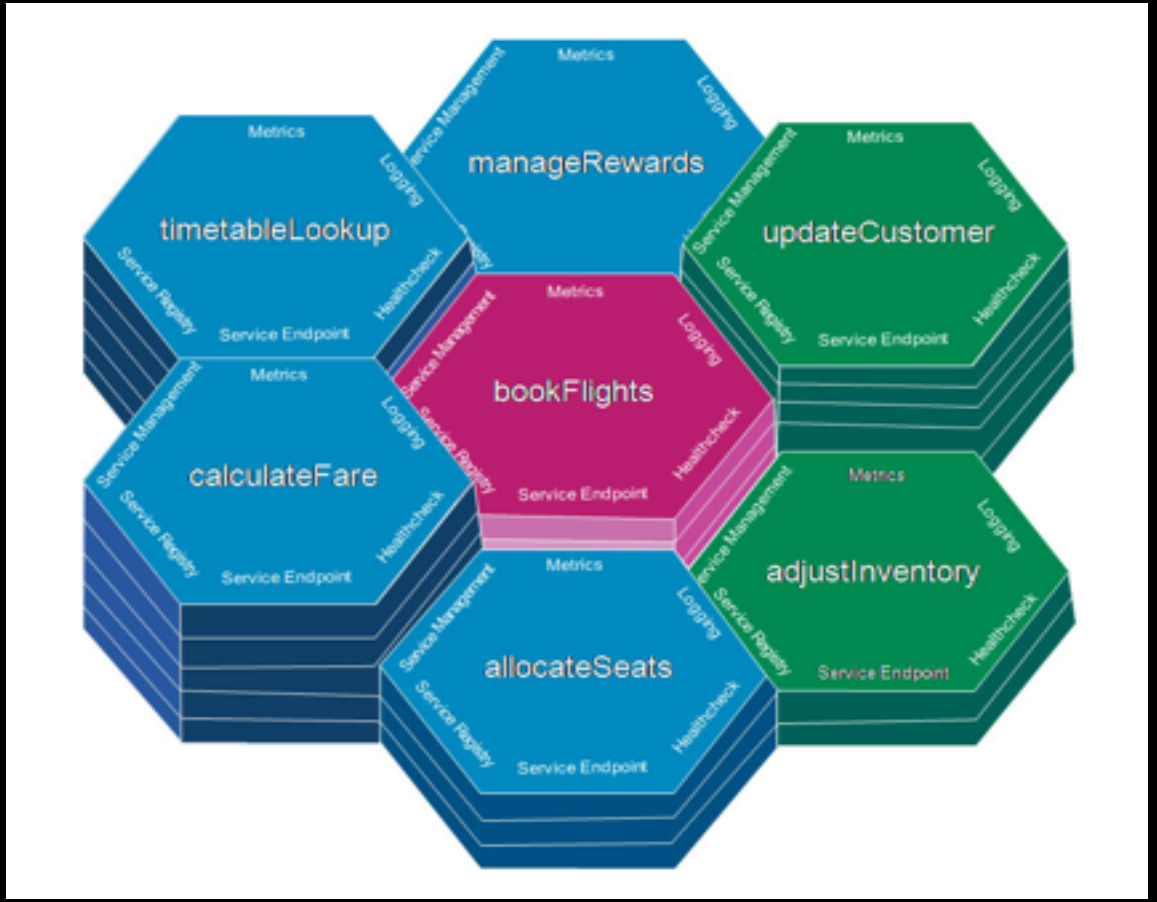


Microservices Architecture
A microservice is a granular, decoupled component within a broader application

Example application using microservices

Airline reservation application

- Book flights
- Timetable lookup
- Calculate fare
- Allocate seats
- Manage rewards
- Update customer
- Adjust inventory



Microservices – key tenets

Large monoliths are **broken down into many small services**

- Each service runs its own process
- There is one service per container

Services are **optimized for a single function**

- There is only one business function per service
- The Single-responsibility Principle: A microservice should have one, and only one, reason to change

Communication via **REST API** and **message brokers**

- Avoid tight coupling introduced by communication through a database

Per-service continuous delivery (CI/CD)

- Services evolve at different rates
- Let the system evolve, but set architectural principles to guide that evolution

Per-service high availability and clustering decisions

- One size or scaling policy is not appropriate for all
- Not all services need to scale; others require autoscaling up to large numbers

Source: IBM Architecture Center

Cloud-Native Application Goals

Horizontal scaling

- Application runs in multiple runtimes spread across multiple hosts (VIII)

Immutable deployment

- A runtime is not patched, it’s replaced (IX)
- A runtime is stateless (VI)
- Shared functionality in backing services (IV)

Elasticity

- Automatic scale-out and scale-in to maintain performance
- Achieved via containerization

Pay-as-you-go charging model

- Pay for what you use

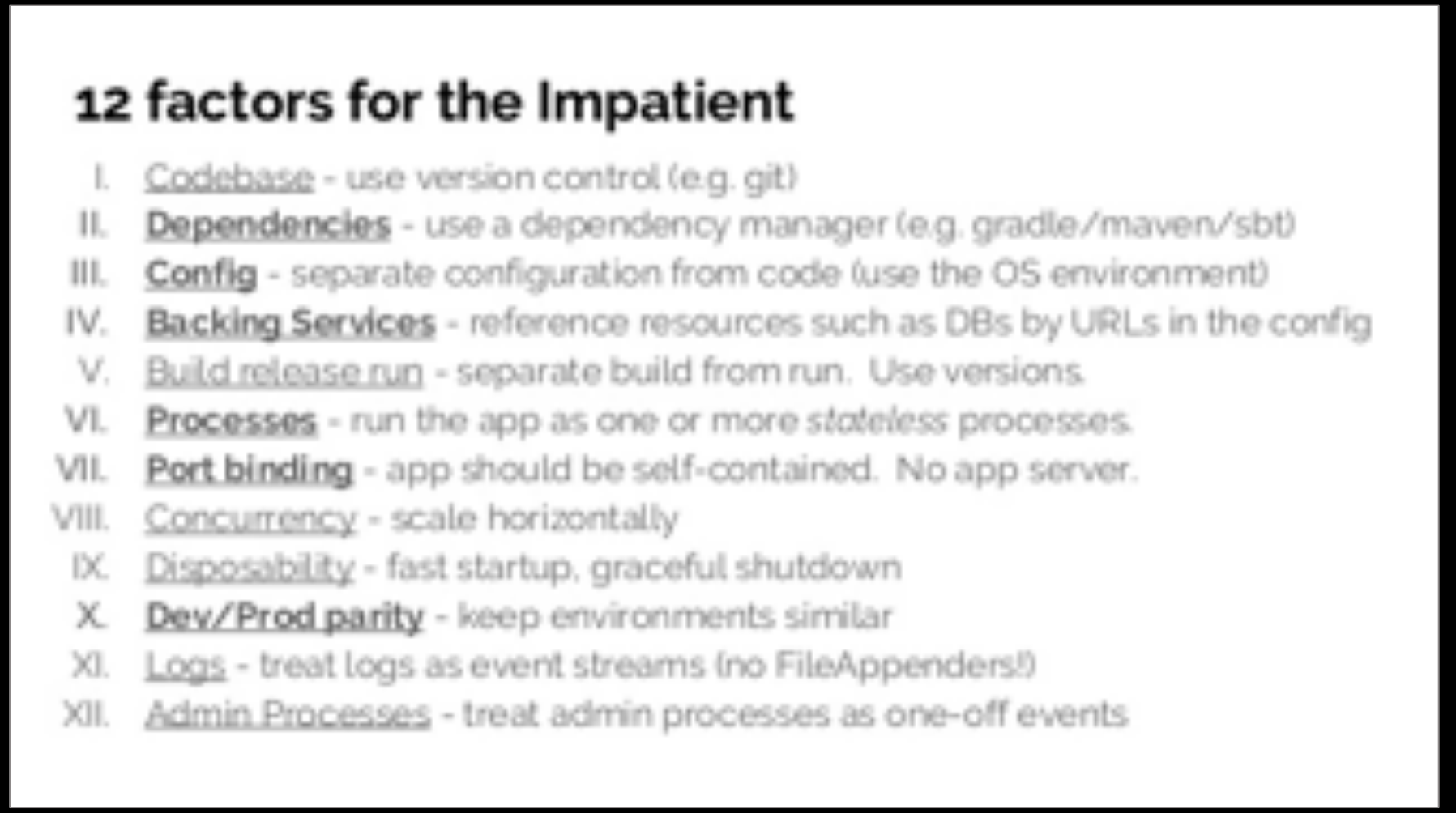
The 12 Factor App

1. Codebase - One codebase tracked in revision control, many deploys
2. Dependencies - Explicitly declare and isolate dependencies
3. Config - Store config in the environment
4. Backing services - Treat backing services as attached resources
5. Build, release, run - Strictly separate build and run stages
6. Processes - Execute the app as one or more stateless processes
7. Port binding - Export services via port binding
8. Concurrency - Scale out via the process model
9. Disposability - Maximize robustness with fast startup and graceful shutdown
10. Dev/prod parity - Keep development, staging, and production as similar as possible
11. Logs - Treat logs as event streams
12. Admin processes - Run admin/management tasks as one-off processes

Not a methodology to do cloud-native design!

https://12factor.net/

The 12 Factor App



12 factors for the Impatient

I. Codebase - use version control (e.g. git)
II. **Dependencies** - use a dependency manager (e.g. gradle/maven/sbt)
III. **Config** - separate configuration from code (use the OS environment)
IV. **Backing Services** - reference resources such as DBs by URLs in the config
V. Build release run - separate build from run. Use versions.
VI. **Processes** - run the app as one or more *stateless* processes.
VII. **Port binding** - app should be self-contained. No app server.
VIII. Concurrency - scale horizontally
IX. Disposability - fast startup, graceful shutdown
X. **Dev/Prod parity** - keep environments similar
XI. Logs - treat logs as event streams (no FileAppenders!)
XII. Admin Processes - treat admin processes as one-off events

Not a methodology to do cloud-native design!

https://12factor.net/

Microservices - principles

One job

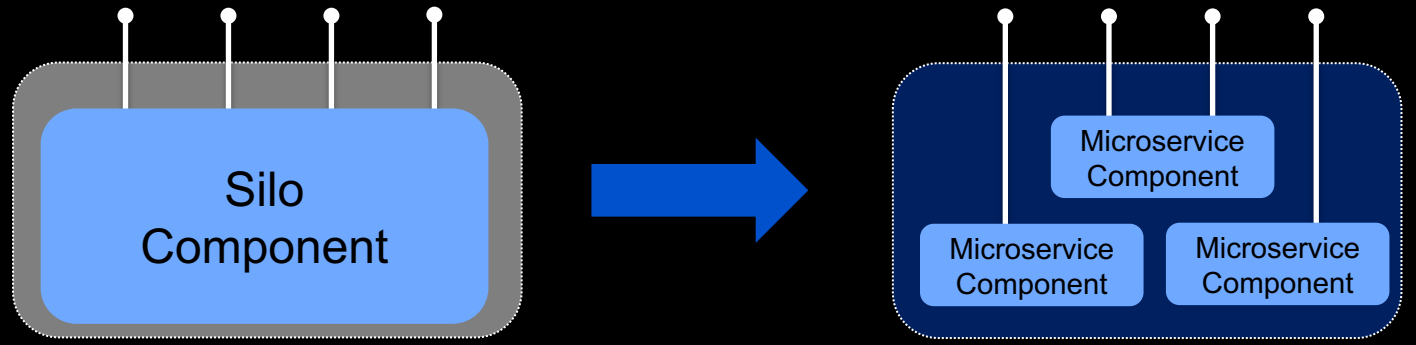
Each microservice must be **optimized for a single function**. Each service is smaller and simpler to write, maintain, and manage. Robert Martin calls this principle the "single responsibility principle."

Separate processes

Communication between microservices must be conducted through **REST API and message brokers**. All communication from service to service must be through the service API or must use an explicit communication pattern, such as the Claim Check Pattern from Hohpe.

Execution scope

Although microservices can expose themselves through APIs, the focus is not on interfaces, but on the running components. The granularity of a microservices application is highlighted in this figure:



Source: IBM Architecture Center

Microservices - principles

CI/CD

Each microservice can be continuously integrated (CI) and continuously delivered (CD). When you build a large application that is composed of many services, you soon realize that different services evolve at different rates. If each service has a unique continuous integration or continuous delivery pipeline, the service can proceed at its own pace. In the monolithic approach, different aspects of the system are all released at the speed of the slowest moving part of the system.

Resiliency

You can apply high availability and clustering decisions to each microservice. When you build large systems, another realization that you have is that when it comes to clustering, one size does not fit all. The monolithic approach of scaling all the services in the monolith at the same level can lead to the overuse or underuse of services. Even worse, when shared resources are monopolized, services might be neglected. In a large system, you can deploy services that do not need to scale to a minimum number of servers to conserve resources. Other services require scaling up to large numbers.

Microservices - rules for developing cloud applications

1. Don’t code your application directly to a specific topology
2. Don’t assume the local file system is permanent
3. Don’t keep session state in your application
4. Don’t log to the file system
5. Don’t assume any specific infrastructure dependency
6. Don’t use infrastructure APIs from within your application
7. Don’t use obscure protocols
8. Don’t rely on OS-specific features
9. Don’t manually install your application

Read the article : http://www.ibm.com/developerworks/websphere/techjournal/1404_brown/1404_brown.html

Microservices - Advantages

In a microservices architecture each component:

- Is **developed independently** and has **limited, explicit dependencies** on other services
- Is developed by a **single, small team** in which all team members can understand the entire code base
- Is developed on its **own timetable** so new versions are delivered independently of other services
- **Scales and fails independently** which isolates any problems
- Can be developed in a different **language**
- **Manages its own data** to select the best technology and schema

Source: IBM Architecture Center

Microservices - Beneifts

- **Efficient teams**
- **Simplified deployment**
- **Right tools for the job**
- **Improved application quality**
- **Scalability**

Source: IBM Architecture Center

That said....

Microservices are hard

..but we're getting ahead of ourselves

IBM

QUESTIONS?

IBM **Cloud**

IBM

The Journey to Cloud
Docker

02

IBM **Cloud**

Everybody 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

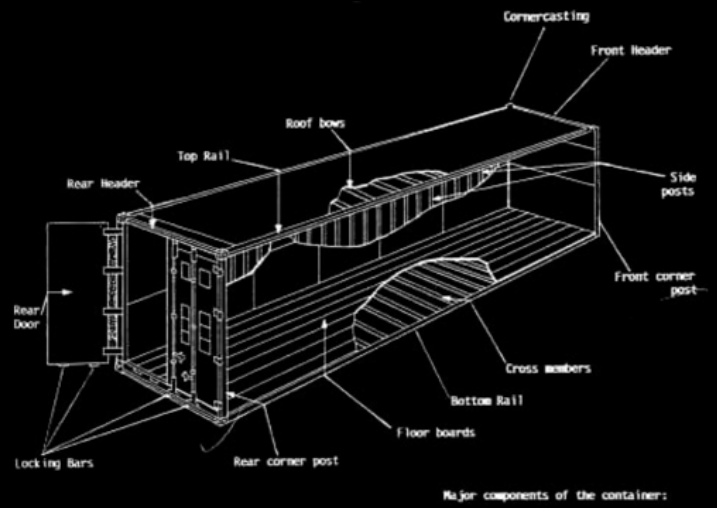
Containers work by **isolating the differences between applications** inside the container so that everything outside the container can be standardized

Microservices implementation with Containers

Why it works – separation of concerns

Development

- Worries about what's "**inside**" the container
  - Code
  - Libraries
  - Package Manager
  - Apps
  - Data
- All Linux servers look the same

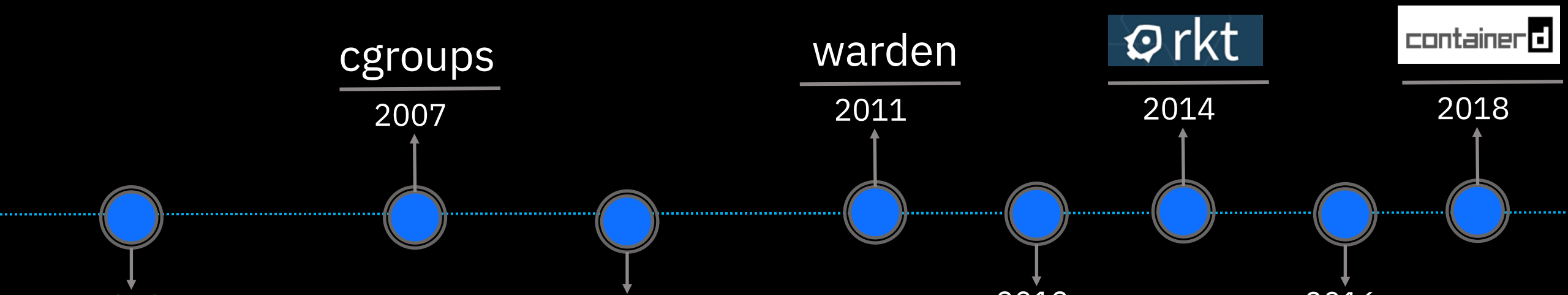


Operations

- Worries about what's "**outside**" the container
  - Logging
  - Remote Access
  - Monitoring
  - Network Config
- All containers start, stop, copy, attach, migrate, etc… the same way

Clear ownership boundary between
Dev and IT Ops drives DevOps adoption and fosters agility

Container History



chroot

A chroot on Unix operating systems is an operation that changes the apparent root directory for the current running process and its children. A program that is run in such a modified environment cannot name (and therefore normally cannot access) files outside the designated directory tree. ... The modified environment is called a "chroot jail".

https://en.wikipedia.org/wiki/ChrootChr

LXC

LXC (Linux Containers) is an operating-system-level virtualization environment for running multiple isolated Linux systems (containers) on a single Linux control host.

...

LXC combines kernel's **cgroups** and support for **isolated namespaces** to provide an isolated environment for applications.

https://en.wikipedia.org/wiki/LXC

Docker

Garden runC

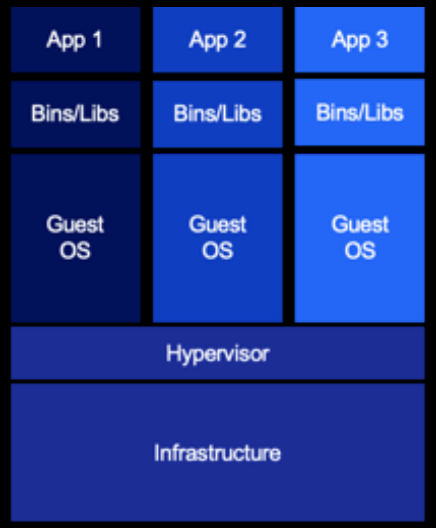
https://containerd.io/

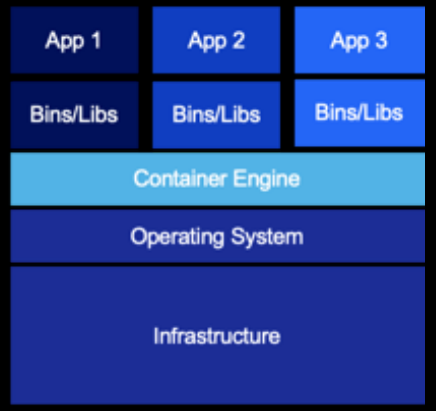
The life of Containers

The process inside the container does not know that it is inside a container

It might feel a little bit lonely but the outside world looks perfectly normal to it

VMs vs. Containers

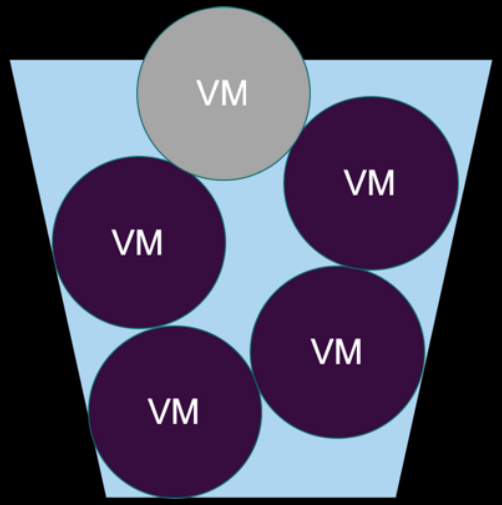


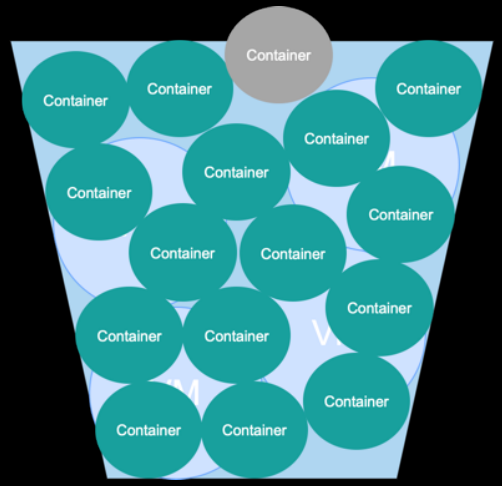


+ VM Isolation
- Complete OS
- Static Compute
- Static Memory

+ Container Isolation
+ Shared Kernel
+ Burstable Compute
+ Burstable Memory

VMs vs. Containers





+ VM Isolation
- Complete OS
- Static Compute
- Static Memory
- Low Resource Utilization

+ Container Isolation
+ Shared Kernel
+ Burstable Compute
+ Burstable Memory
+ High Resource Utilization

VM vs. Containers

Features	Virtual Machines (VMs)	Application Containers	Application Container Benefits
Virtualization	In VMs, the virtualization occurs in server hardware.	In containers, the virtualization occurs at the OS level.	Faster time-to-market
Standardization	VMs are standardized systems that have capabilities similar to that of bare metal computers.	Containers are not standardized, as their kernel OS has varying degrees of complexity.	Makes applications more portable by isolating them from the underlying infrastructure
Resource management	Each VM has its own kernel OS, binary, and library.	Containers share the same host OS, and if needed, both binary and library.	Lower overhead costs
Density	VMs require a few gigabytes (GB) of space that limits them into a single server.	Containers require a few megabytes (MB) of space that enables many containers to fit into a single server.	Lighter weight than VMs
Boot time	Minutes	Seconds	Easy to update and release newer versions of applications

Advantages of Containers

Containers are **portable**

Containers are **easy to manage**

Containers provide **“just enough” isolation**

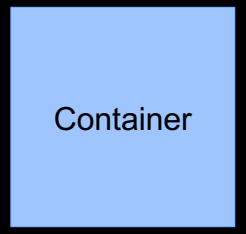
Containers use hardware **more efficiently**

Containers are **immutable**

Docker Components

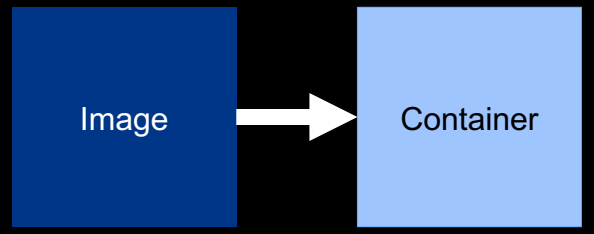
Container

Smallest compute unit



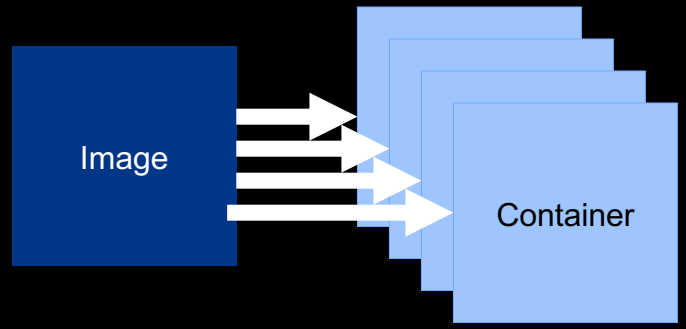
Docker Components

Containers
are created from
Images



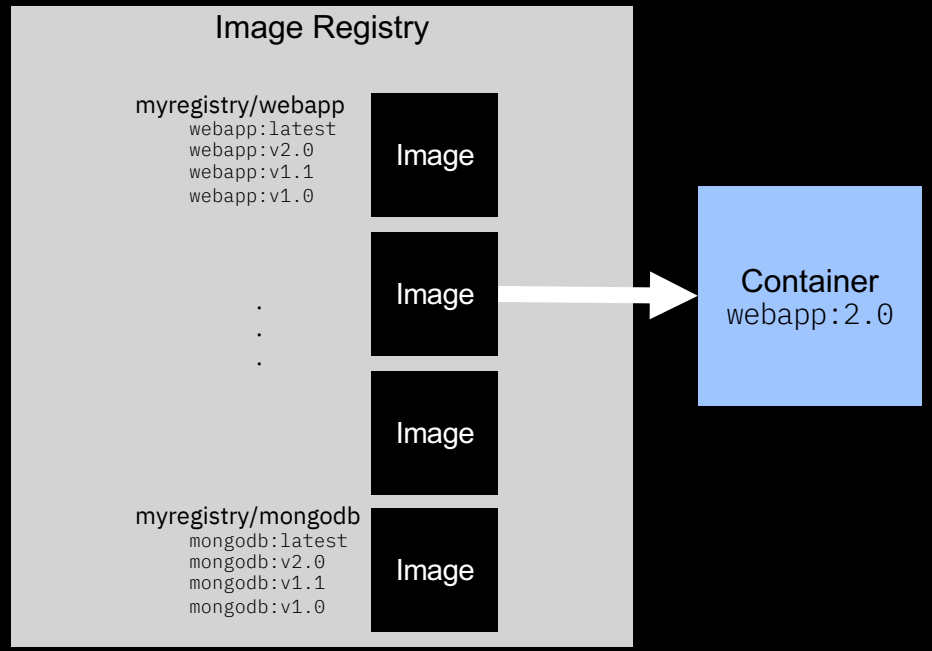
Docker Components

As **many Containers**
as needed can be created from
Images



Docker Components

The **Image Registry**
stores the versioned
Images
to create
Containers



Docker Components

Image

A read-only snapshot of a container stored in Docker Hub to be used as a template for building containers

Container

The standard unit in which the application service resides or transported

Docker Hub/Registry

Available in SaaS or Enterprise to deploy anywhere you choose
Stores, distributes, and shares container images

Docker Engine

A program that creates, ships, and runs application containers
Runs on any physical and virtual machine or server locally, in private or public cloud. Client communicates with Engine to execute commands

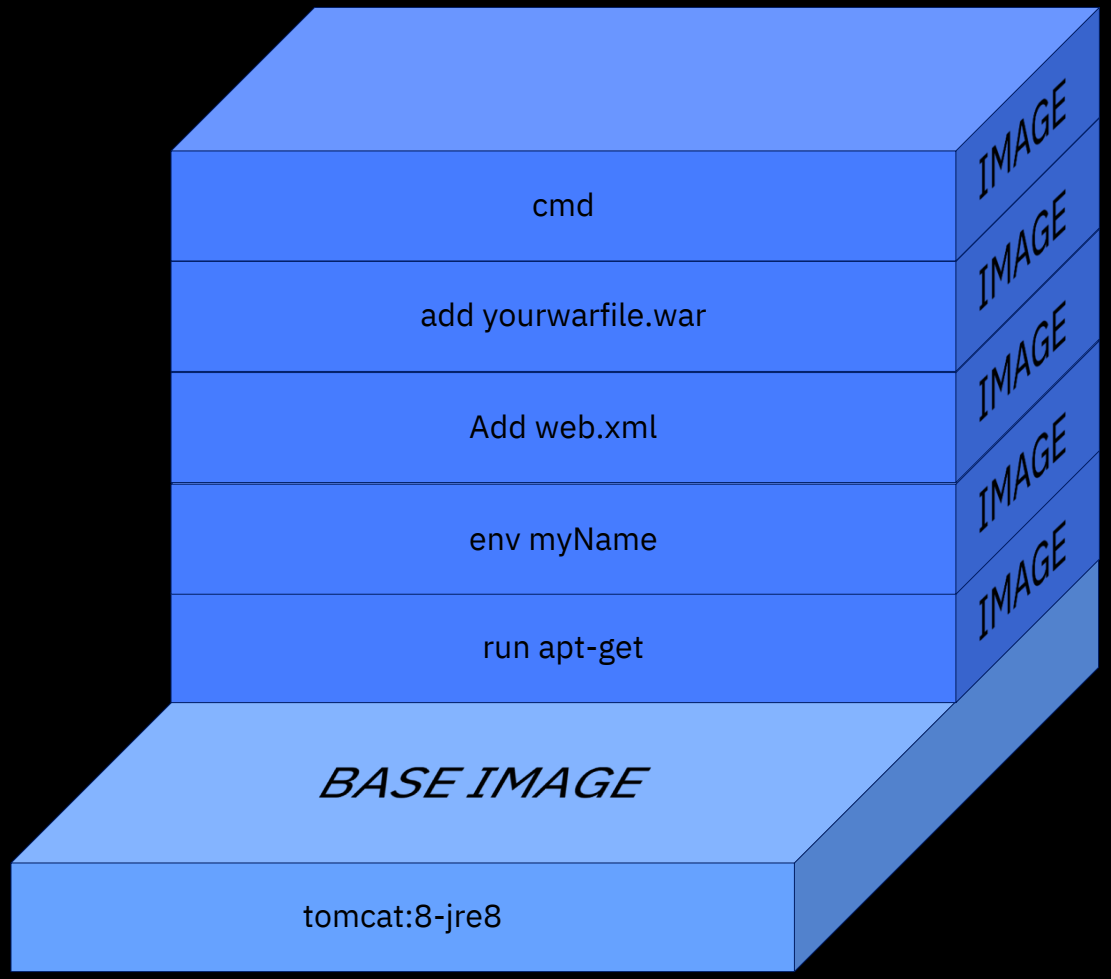
Docker Images

Docker Layers

Inheritance

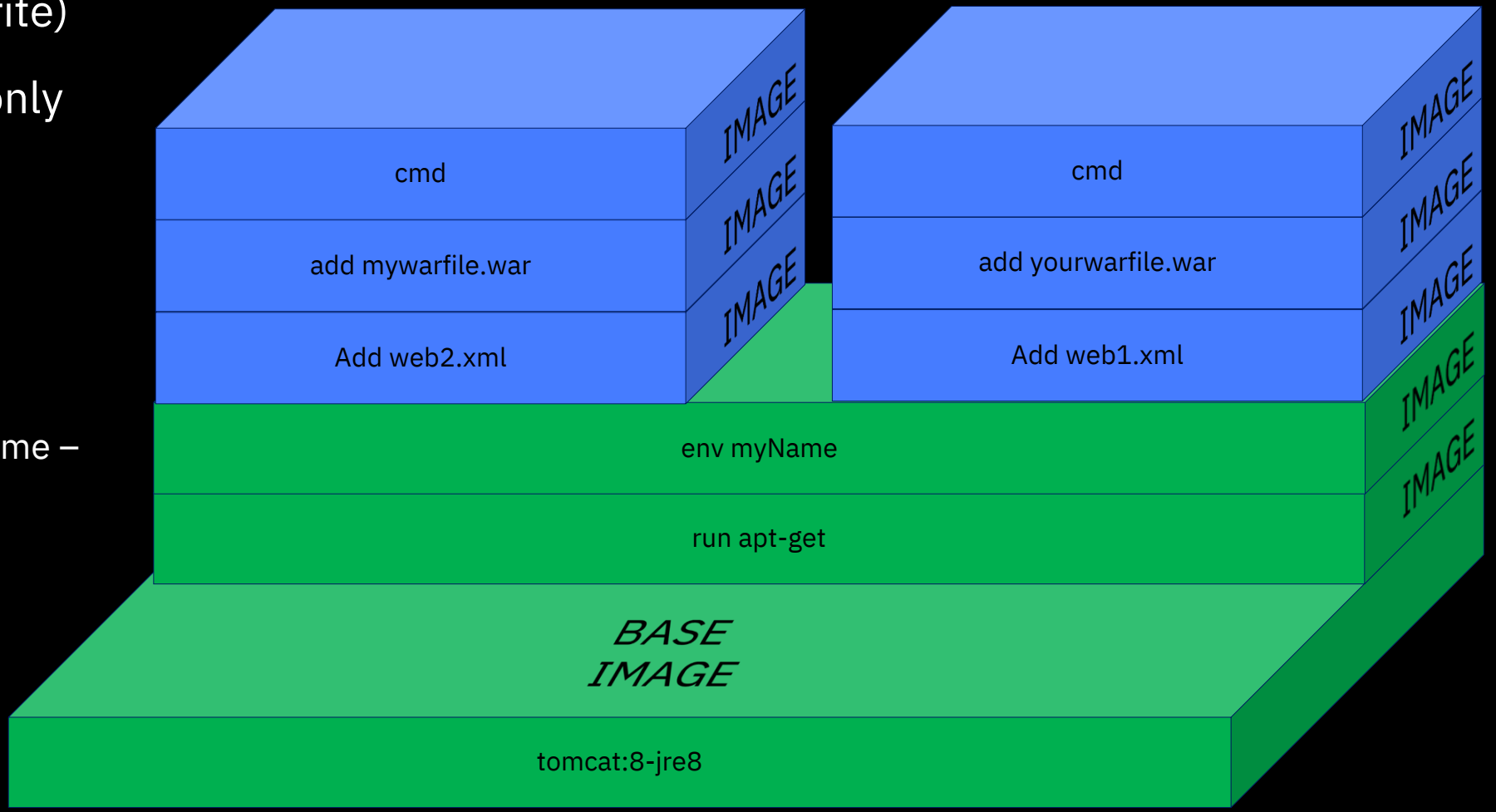
- Build on the work of those who came before you

```
# Pull base image
FROM tomcat:8-jre8

# Maintainer
MAINTAINER "youremailaddress"

# Run command
RUN apt-get update && apt-get -y upgrade

# Set variables
ENV myName John Doe

# Copy to images tomcat path
ADD web.xml /usr/local/tomcat/conf/
ADD yourwarfile.war /usr/local/tomcat/webapps/

# Run server
CMD ["catalina.sh", "run"]
```

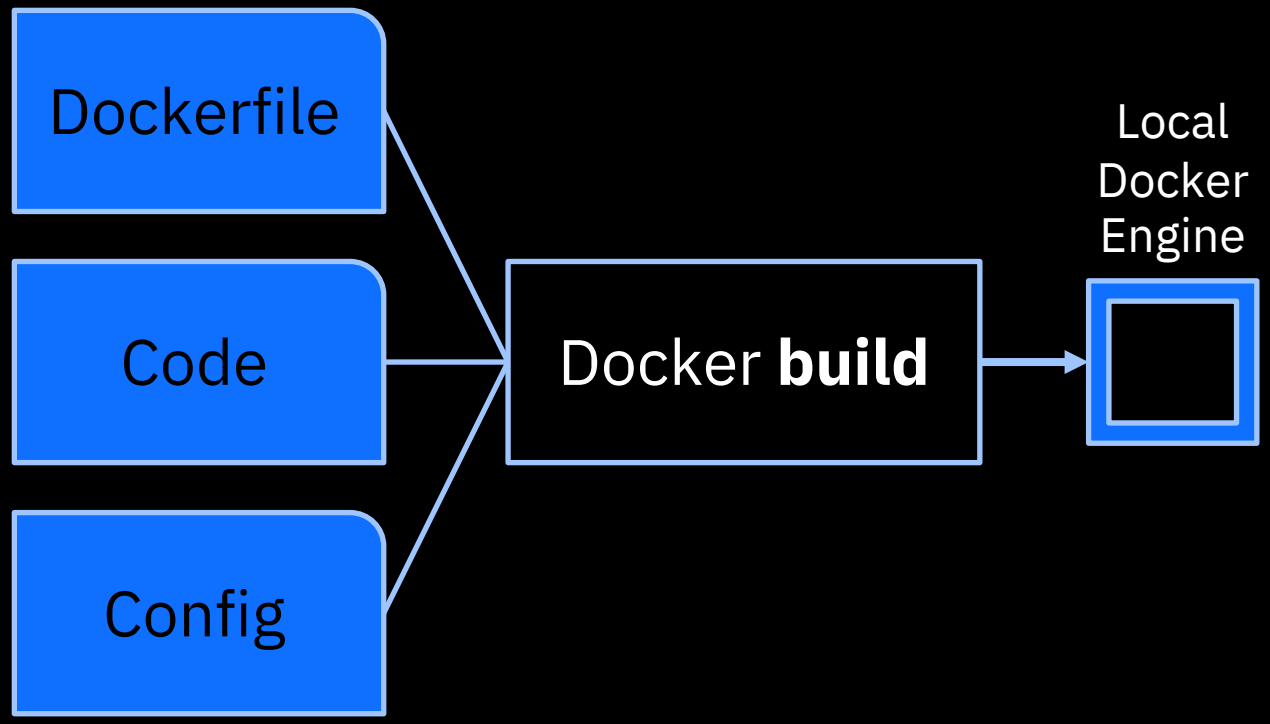


Docker Images

- Docker uses a **layered filesystem** (copy-on-write)
- New files (& edits) are only visible to current/above layers
- Layers allow for reuse
  - More containers per host
  - Faster start-up/download time – base layers are "cached"



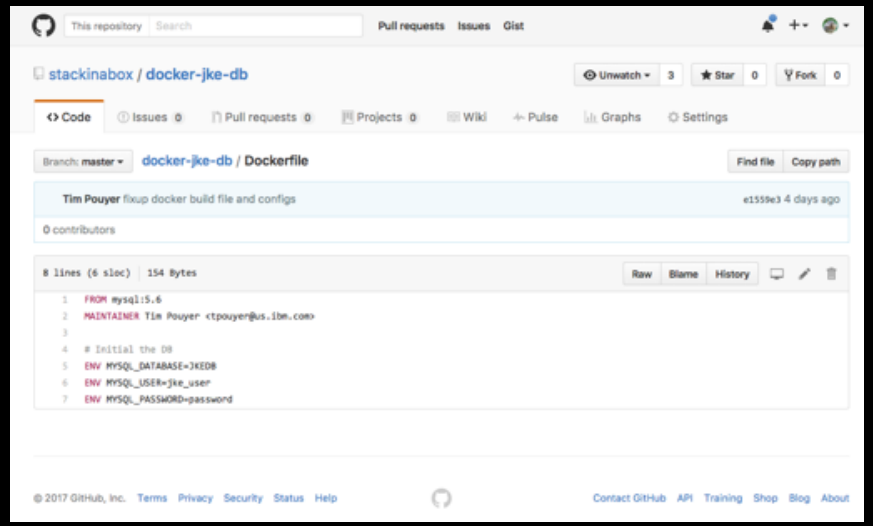
Docker Basics – Build



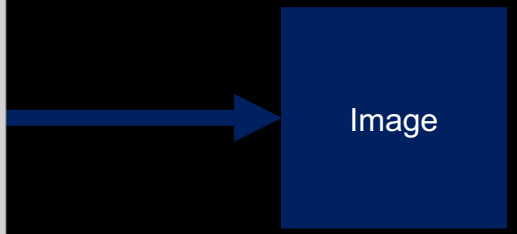
Docker Basics - Build

Dockerfile

- Text based file describing:
  - Previous Layer
  - Environment Variables
  - Commands used to populate data/software/frameworks/etc...
  - Command to run when executed




```
# Pull base image
FROM tomcat:8-jre8

# Maintainer
MAINTAINER "youremailaddress"

# Run command
RUN apt-get update && apt-get -y upgrade

# Set variables
ENV myName John Doe

# Copy to images tomcat path
ADD web.xml /usr/local/tomcat/conf/
ADD yourwarfile.war /usr/local/tomcat/webapps/

# Run server
CMD ["catalina.sh", "run"]
```



Docker Basics – Build -Dockerfile

one process per container

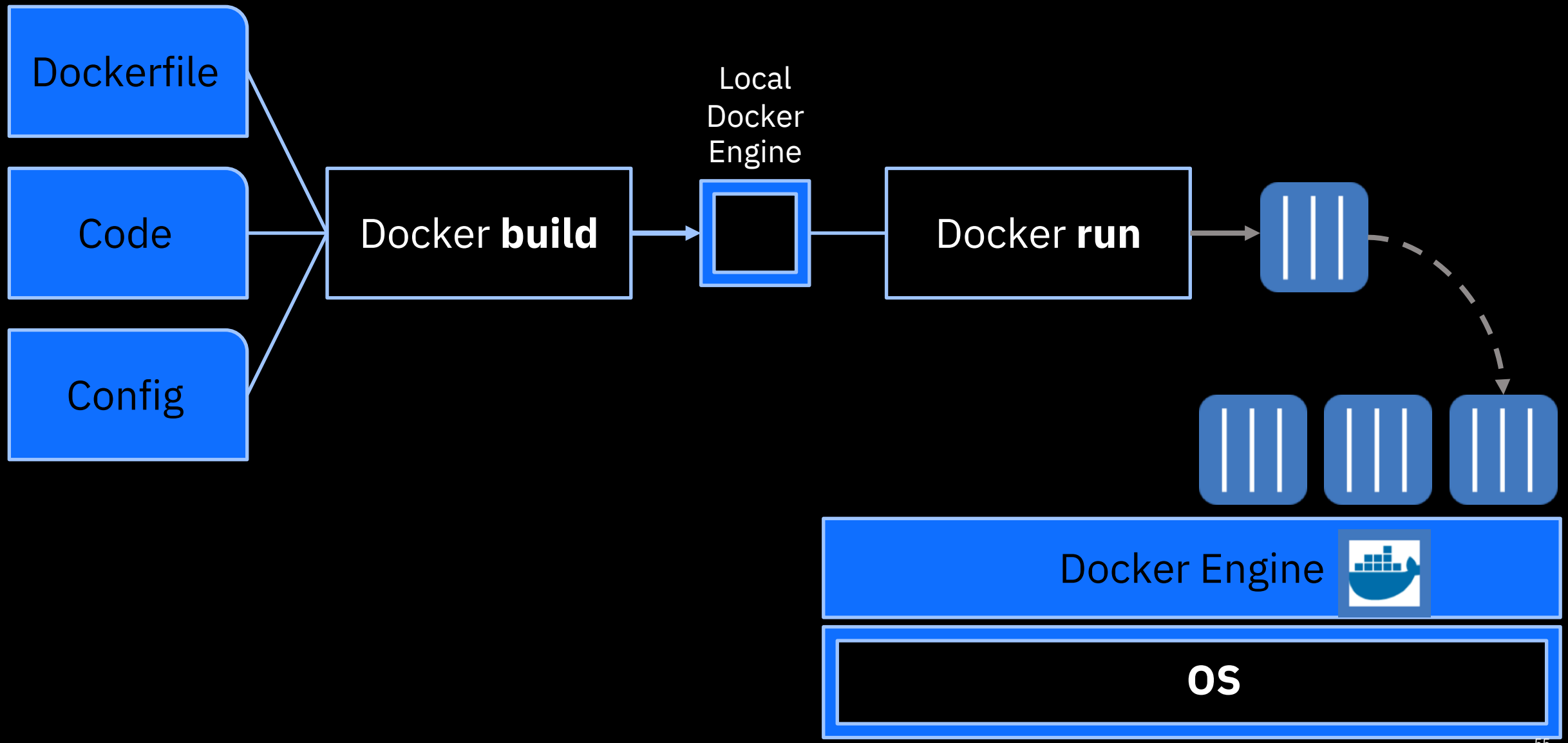
- A text file that builds an image using Docker directives

- FROM
- RUN
- COPY
- ENTRYPOINT
- LABEL
- ENV
- ARG
- USER
- EXPOSE

```
# Pull base image
FROM tomcat:8-jre8

# Maintainer
MAINTAINER "youremailaddress"

# Run command
RUN apt-get update && apt-get -y upgrade

# Set variables
ENV myName John Doe

# Copy to images tomcat path
ADD web.xml /usr/local/tomcat/conf/
ADD yourwarfile.war /usr/local/tomcat/webapps/

# Run server
CMD ["catalina.sh", "run"]
```

```
docker build –t myimage ./Dockerfile
```

Docker Basics – Run

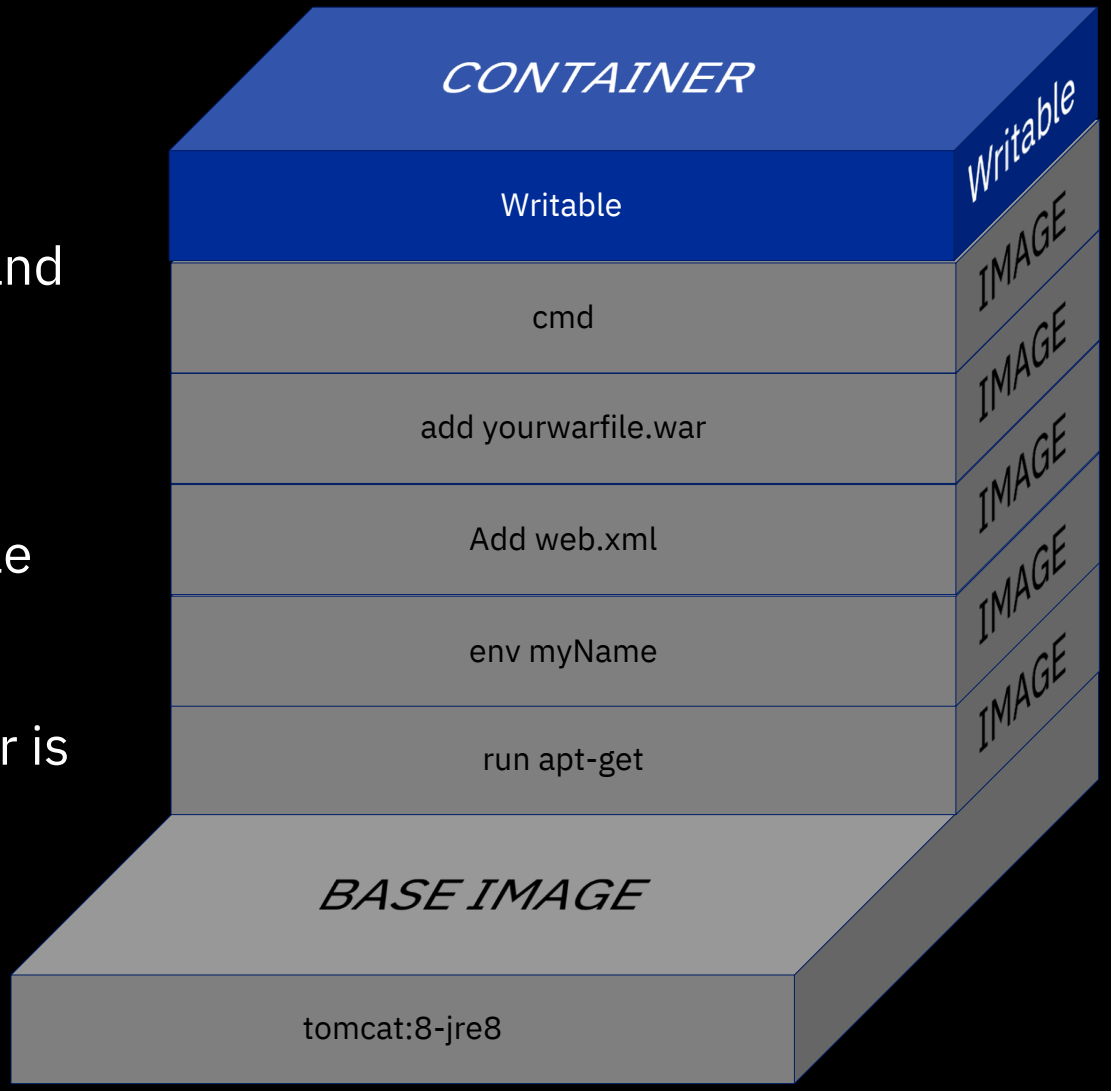


Docker Basics – Run

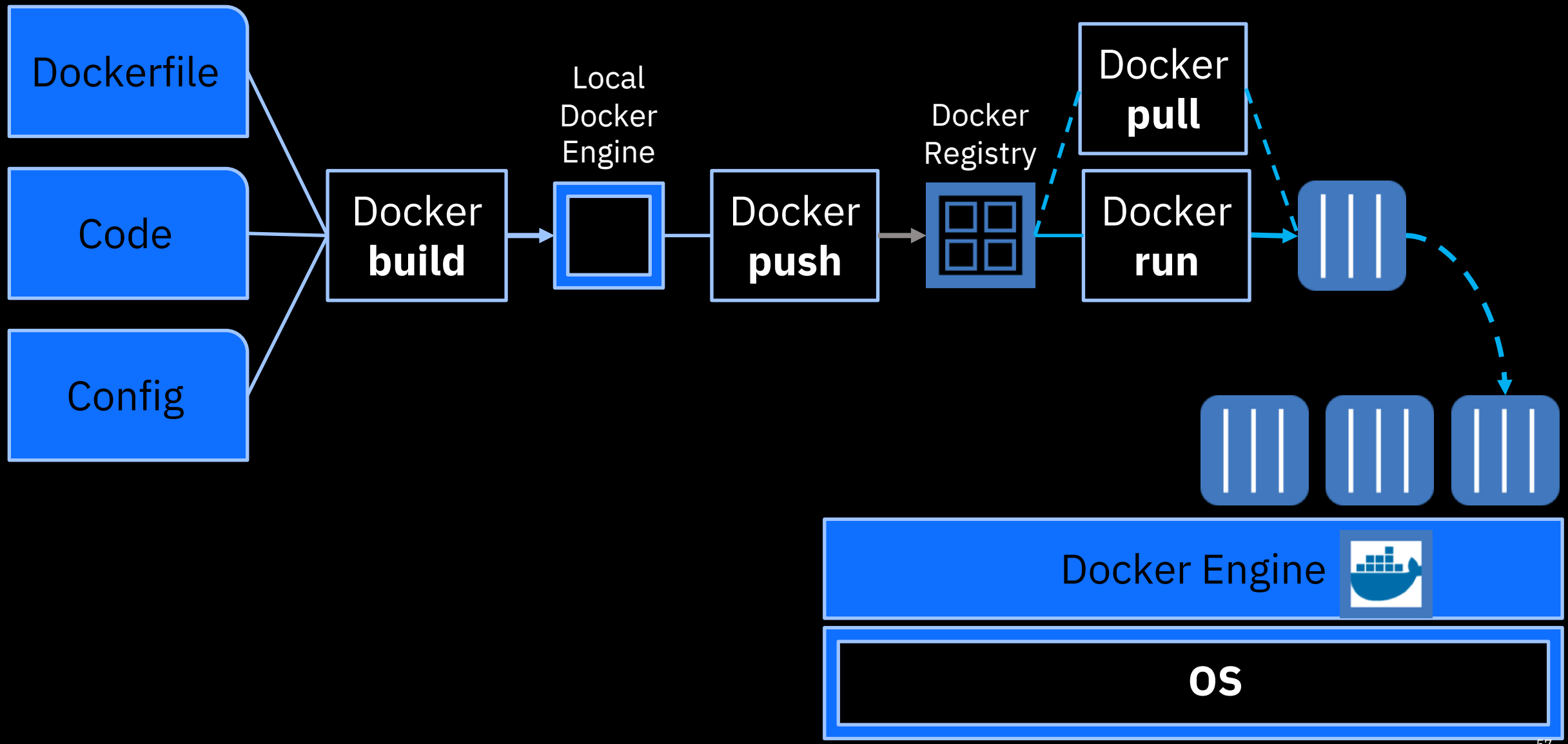
Docker Layers

Inheritance

- The biggest difference between a container and an image is the **top writable layer**.
- All writes to the container that add new or modify existing data are stored in this writable layer.
- When a container is deleted, its writable layer is also deleted. The underlying image is unchanged.



Docker Basics – Store, Retrieve & Run with registry



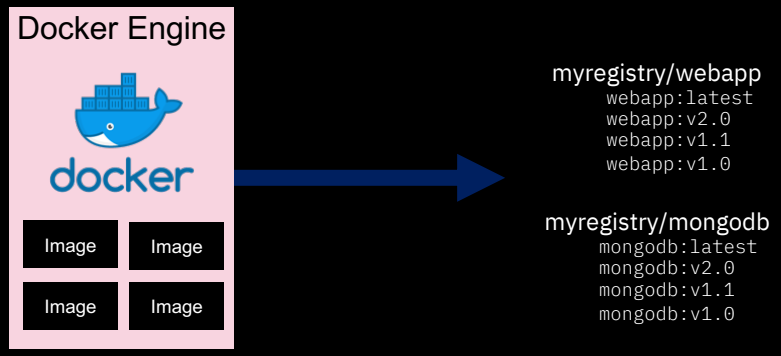
Docker Basics – Store & Retrieve

Docker Registry

- Private Local
- Public Docker Hub
- Private Shared

```
docker images
```

```
tpouyer at Laptop in ~/Development/workspace/docker/docker-jke-db on master
$ docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
stackinabox/demo-docker             dev                 4e2a1f6cc1a4        25 hours ago        528.3 MB
stackinabox/demo-jke                1.1                 a596fbf2c3b7        3 days ago          672 MB
stackinabox/demo-jke                dev                 a596fbf2c3b7        3 days ago          672 MB
stackinabox/demo-jke                latest              a596fbf2c3b7        3 days ago          672 MB
stackinabx/demo-jke                 1.1                 a596fbf2c3b7        3 days ago          672 MB
stackinabx/demo-jke                 latest              a596fbf2c3b7        3 days ago          672 MB
stackinabox/jke-web                 latest              a4be0cdad8bc        4 days ago          462.3 MB
stackinabox/demo-jke-db             dev                 54bdbf42b999        4 days ago          327.5 MB
stackinabox/jke-db                  latest              54bdbf42b999        4 days ago          327.5 MB
stackinabox/urbancode-deploy-client 6.2.2.0             53c878fbf034        7 days ago          477 MB
websphere-liberty                   javaee7             7e27c3fb9c96        10 days ago         445.7 MB
mysql                               5.6                 a896fd82dcd5        13 days ago         327.5 MB
mysql                               latest              f3694c67abdb        13 days ago         400.1 MB

tpouyer at Laptop in ~/Development/workspace/docker/docker-jke-db on master
$
```



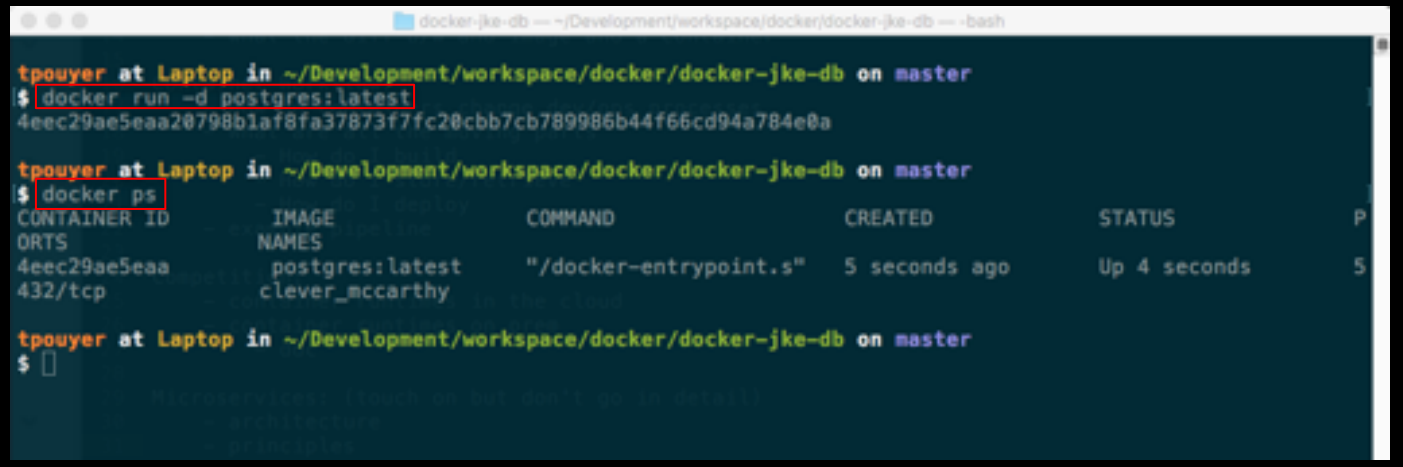
myregistry/webapp
- webapp:latest
- webapp:v2.0
- webapp:v1.1
- webapp:v1.0

myregistry/mongodb
- mongodb:latest
- mongodb:v2.0
- mongodb:v1.1
- mongodb:v1.0

Docker Basics – Run

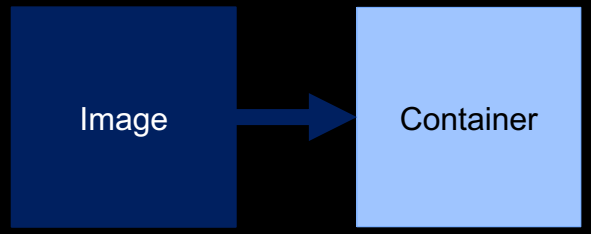
Docker run

- Local (containerd)




```
docker run -d -e MYVAR=foo myimage:1.0.0
docker run -ti myimage:1.0.0 /bin/bash
```

…



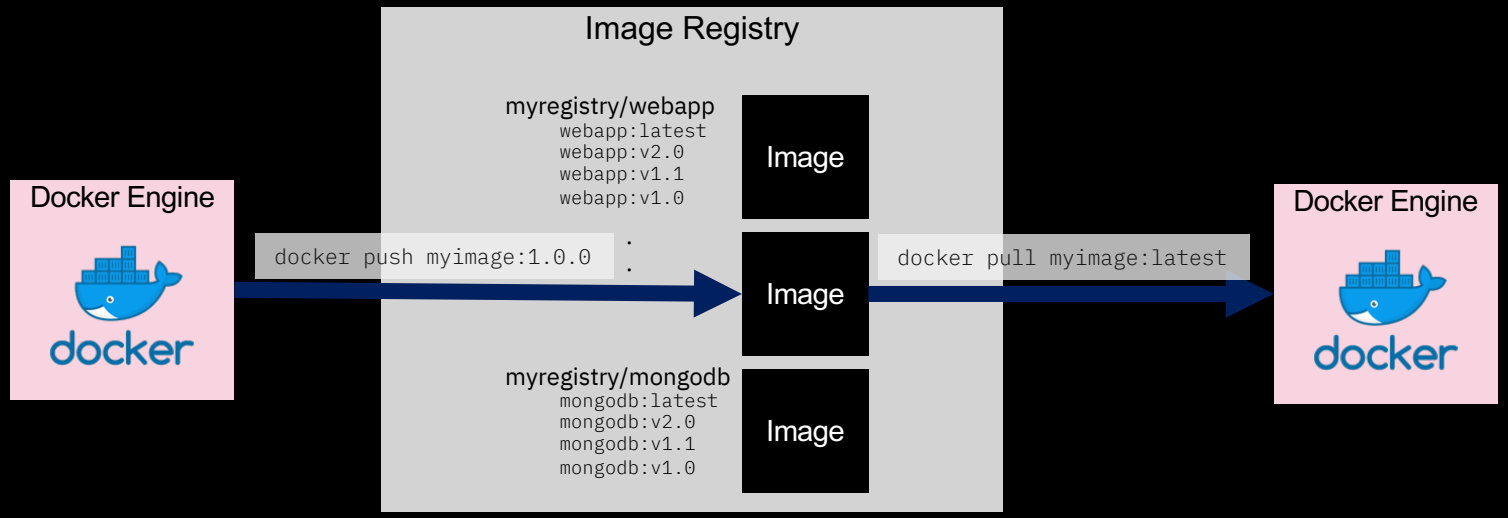
Docker Basics – Store & Retrieve with registry

Docker Registry

- Private Local
- Public Docker Hub
- Private Shared

```
docker pull myimage:latest
docker push myimage:1.0.0
```

```
tpouyer at Laptop in ~/Development/workspace/docker/docker-jke-db on master
$ docker pull postgres:latest
latest: Pulling from library/postgres

5040bd298390: Already exists
f08454c3c700: Pull complete
4db038cdfe03: Pull complete
e1d9ba315f03: Pull complete
25e0ee93170e: Pull complete
3f28084c3f51: Pull complete
78c91f0aedcd: Pull complete
93ab52dbcbb8: Pull complete
27ec75825613: Pull complete
28ef691a9920: Pull complete
0f0dd20755c9: Pull complete
2a4a824861f7: Pull complete
Digest: sha256:0842a7ef786aa2658623085160cb38451eb3d40856e7d222ae0069b6e6296877
Status: Downloaded newer image for postgres:latest

tpouyer at Laptop in ~/Development/workspace/docker/docker-jke-db on master
$
```



Docker Basics – Registries

Hosting image repositories

- You can define your own registry
- A registry is managed by a registry container

Public and Private registries

- Public Registry like **Docker Hub**
- **https://hub.docker.com**

Login into the registry

- Docker login domain:port

Docker Recap

- **Containers are not VMs**
- **Containers provide many benefits:**
  - Efficiency
  - Portability
  - Consistency
- **New challenges with containers:**
  - Production apps dependent on open-source projects
  - Existing tools may not be sufficient for container
  - Need to focus on business objectives

IBM

QUESTIONS?

IBM **Cloud**

IBM

The Journey to Cloud

Kubernetes

03

IBM **Cloud**

Everybody Loves Containers

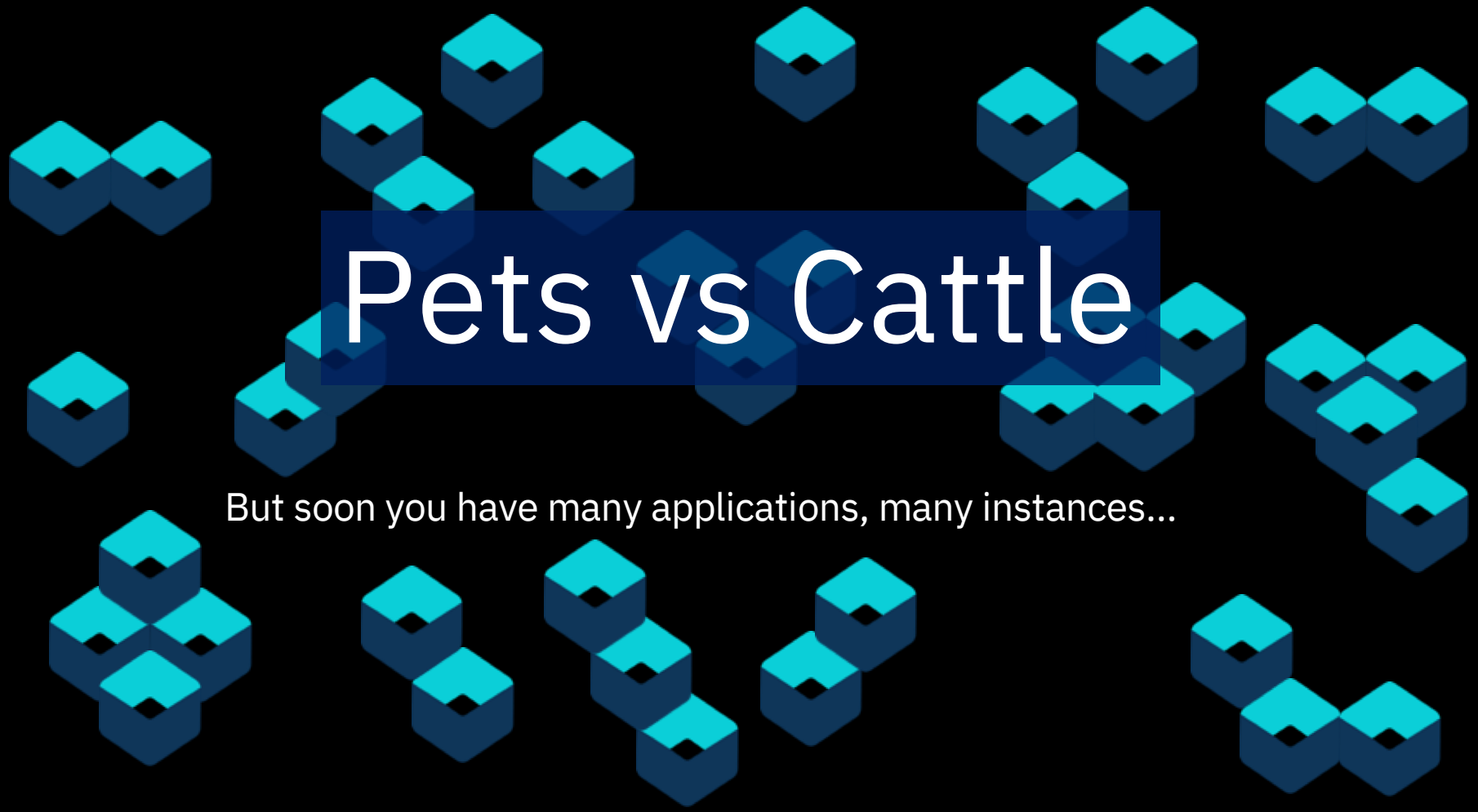
But when you go

From this....

To this....

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

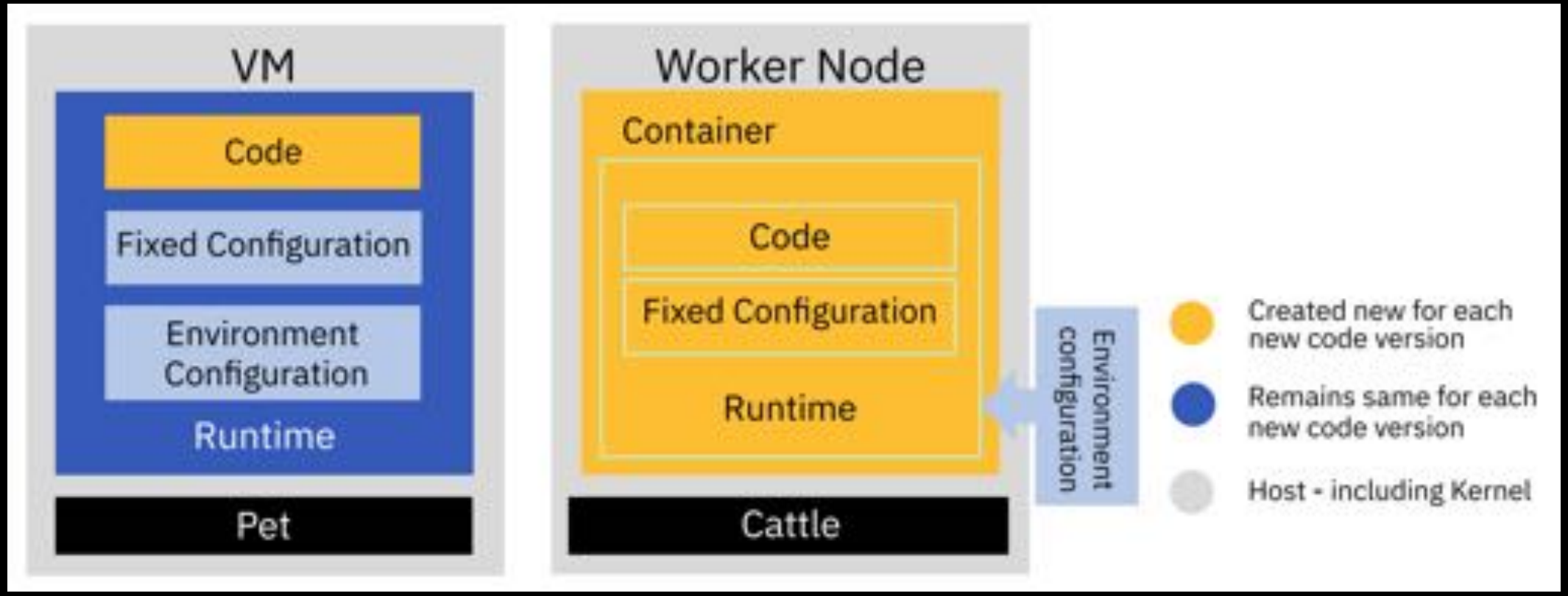
At first the growth is easy to handle....



Pets vs Cattle

But soon you have many applications, many instances...

Pets vs Cattle



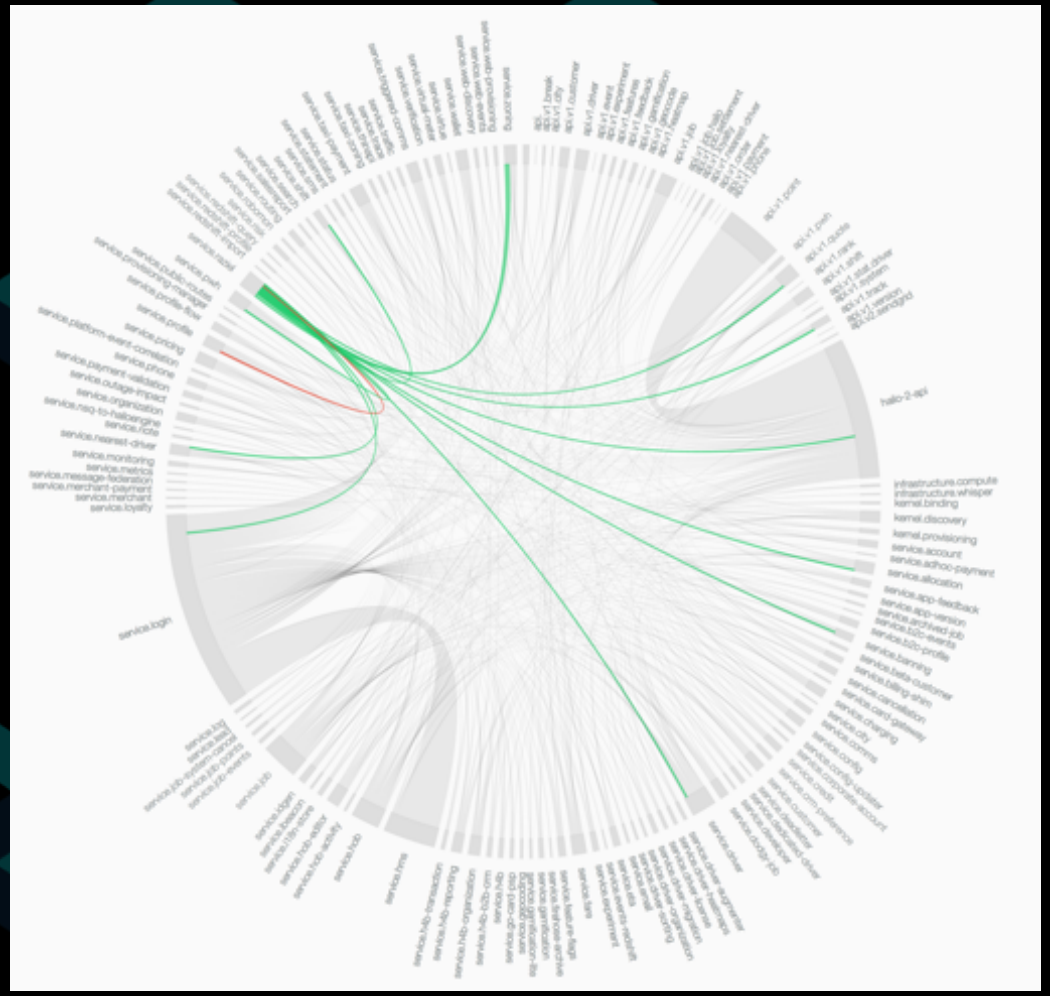
https://developer.ibm.com/tutorials/get-started-with-agile-integration-architecture-integration-modernization-in-action/

The trade off

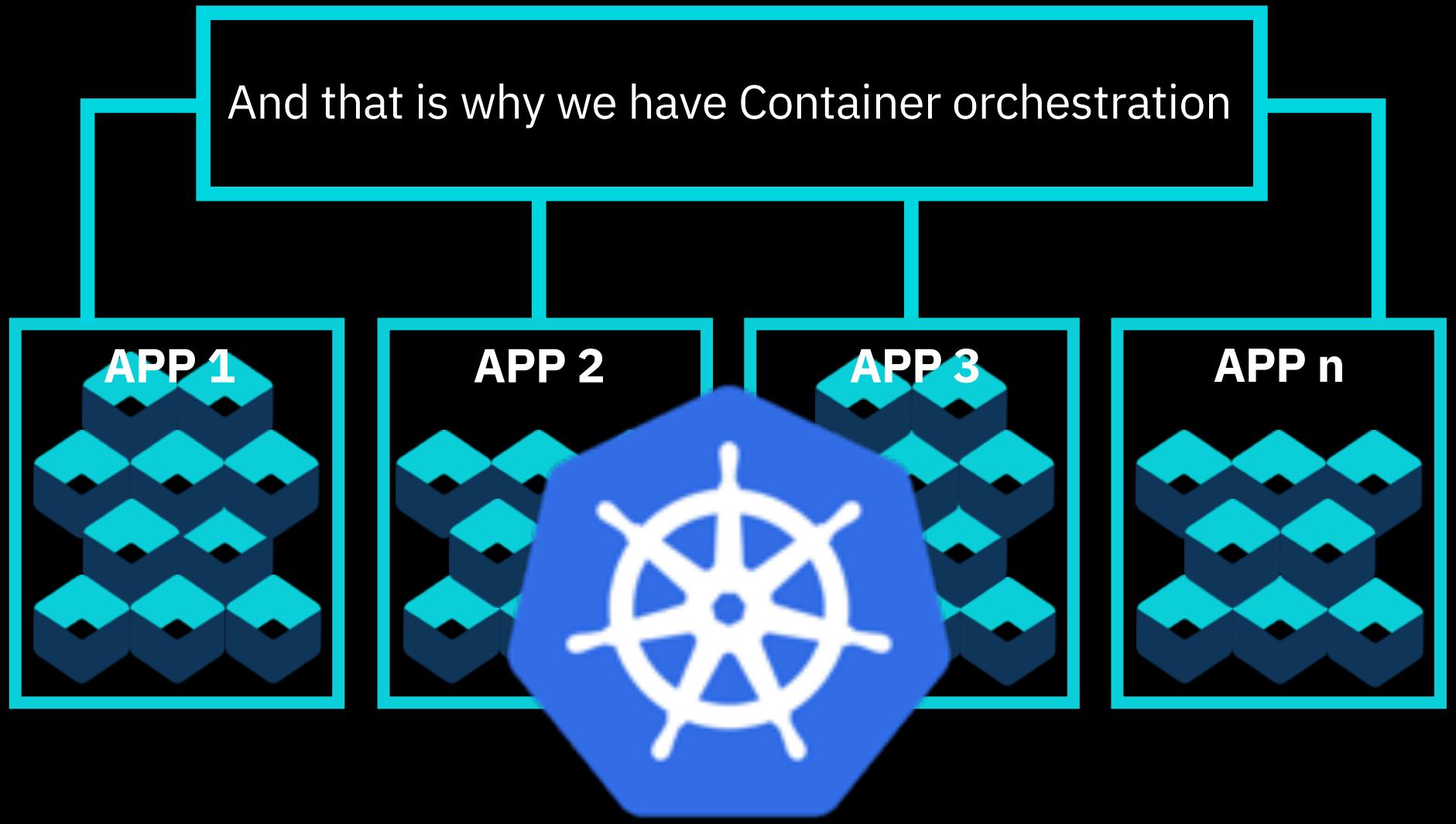
Improved delivery velocity

in exchange for

increased operational complexity



Enter Kubernetes (K8s)



Kubernetes – Declarative System

Imperative Systems

In an imperative system, **the user** knows the desired state and **the user** determines the sequence of commands to transition the system to the desired state.

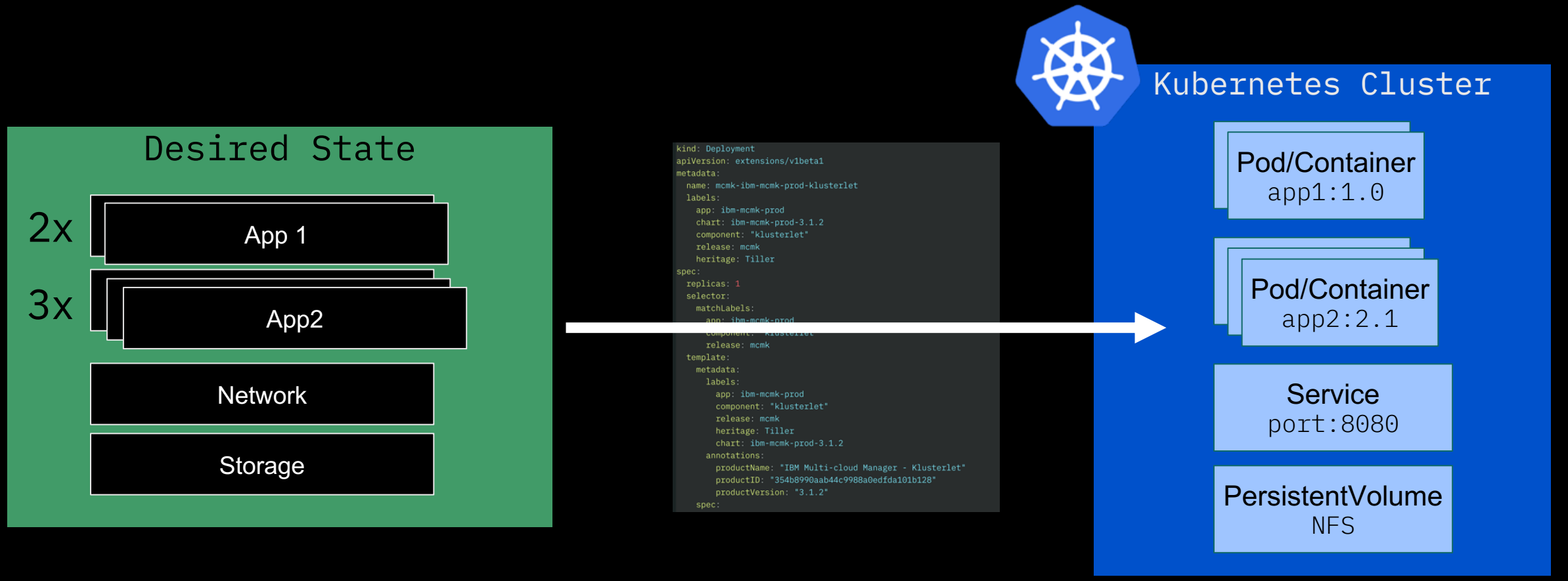
Declarative Systems

By contrast, in a declarative system, **the user** knows the desired state, supplies a representation of the desired state to the system, then **the system** reads the current state and determines the sequence of commands to transition the system to the desired state.

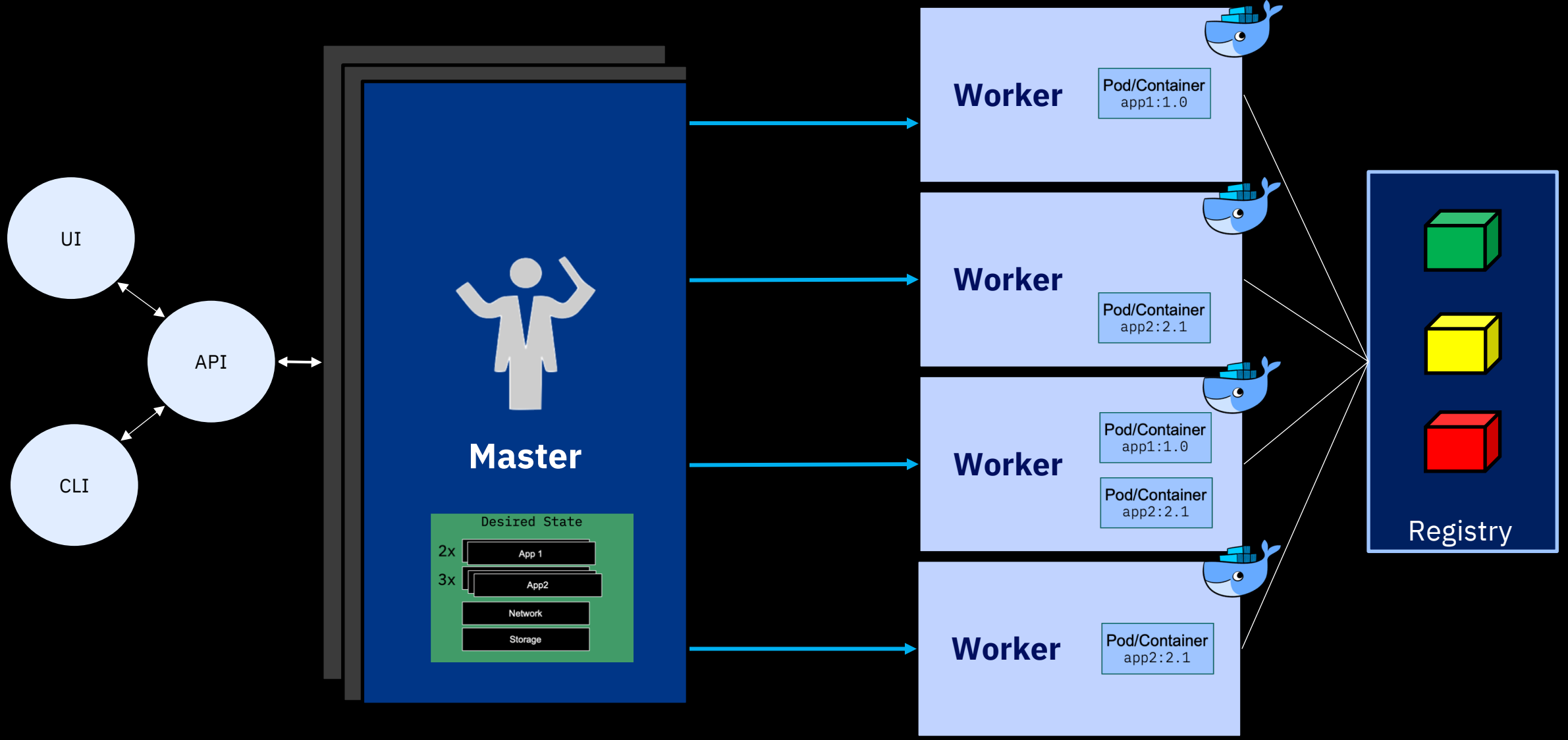
Kubernetes – Declarative System

The Desired State

Kubernetes ensures that all the containers running across the cluster are in the desired state at any moment.



Kubernetes Management Architecture



What is Kubernetes?

Container orchestrator

- Runs and manages containers
- Unified API for deploying web applications, batch jobs, and databases
- Maintains and tracks the global view of the cluster
- Supports multiple cloud and bare-metal environments

Manage applications, not machines

- Rolling updates, canary deploys, and blue-green deployments

Designed for extensibility

- Rich ecosystem of plug-ins for scheduling, storage, networking

Open source project managed by the Linux Foundation

- Inspired and informed by Google's experiences and internal systems
- 100% open source, written in Go

Kubernetes Strengths

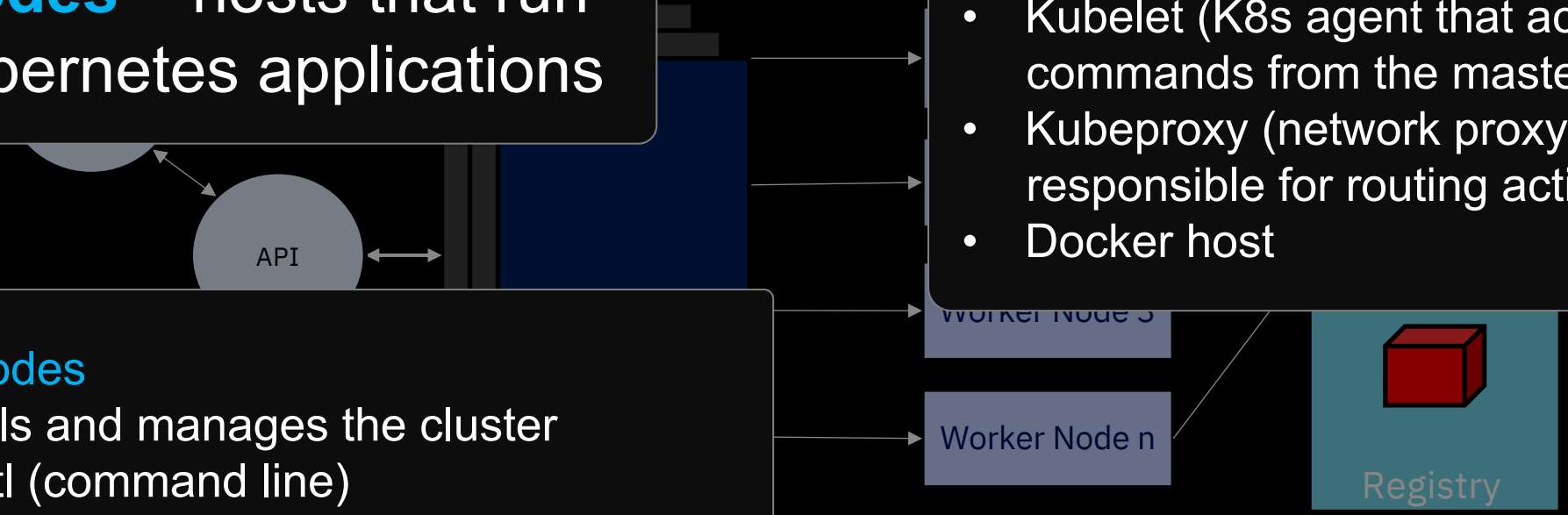
- Kubernetes has a **clear governance model** managed by the Linux Foundation
- A growing and **vibrant** Kubernetes **ecosystem**
- Kubernetes **avoids dependency and vendor lock-in**
- Kubernetes supports a **wide range of deployment options**

Kubernetes Management Architecture

Nodes – hosts that run Kubernetes applications

Worker nodes

- Hosts the K8s services
- Kubelet (K8s agent that accepts commands from the master)
- Kubeproxy (network proxy service responsible for routing activities)
- Docker host



Master nodes

- Controls and manages the cluster
- Kubectl (command line)
- REST API (communication with workers)
- Scheduling and replication logic

- Controller Manager Server
- Scheduler

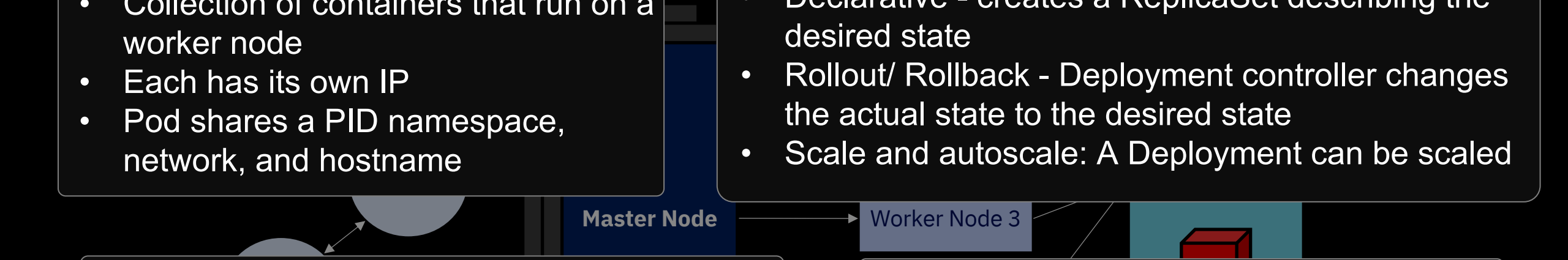
Kubernetes Management Architecture

Pods

- Smallest deployment unit in K8s
- Collection of containers that run on a worker node
- Each has its own IP
- Pod shares a PID namespace, network, and hostname

Deployment

- A set of pods to be deployed together
- Declarative - creates a ReplicaSet describing the desired state
- Rollout/ Rollback - Deployment controller changes the actual state to the desired state
- Scale and autoscale: A Deployment can be scaled



ReplicationSet

- Ensures availability and scalability
- Maintains the number of pods as requested by user
- Uses a template that describes specifically what each pod should contain

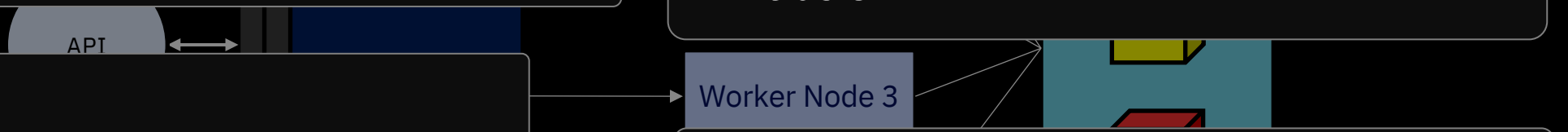
Service

- Collections of pods exposed as an endpoint
- A service provides a way to refer to a set of Pods (selected by labels) with a single static IP address

Kubernetes Management Architecture

ConfigMap
- Configuration values to be used by containers in a pod
- Stores configuration outside of the container image, making containers more reusable

Labels
- Metadata assigned to K8s resources
- Key-value pairs for identification
- Critical to K8s as it relies on querying the
- cluster for resources that have certain labels



Secrets
- Sensitive info that containers need to consume
- Encrypted in special volumes mounted automatically

etcd - a highly-available key value store which K8s uses for persistent storage of all of its REST API objects

- Etcd
- API Server
- Controller Manager Server

API Server – Kubernetes API server

Scheduler – schedules pods in worker nodes

What is container orchestration?

Container orchestration
- Manages the deployment, placement, and lifecycle of workload containers

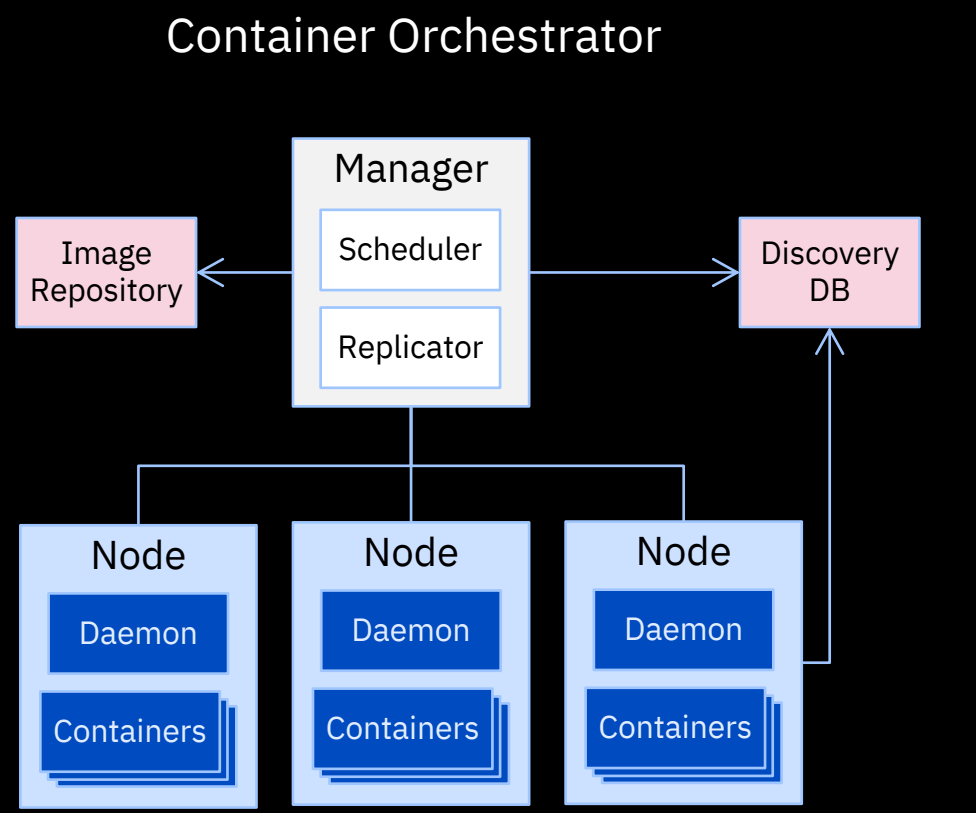
Cluster management
- Federates multiple hosts into one target

Scheduling
- Distributes containers across nodes

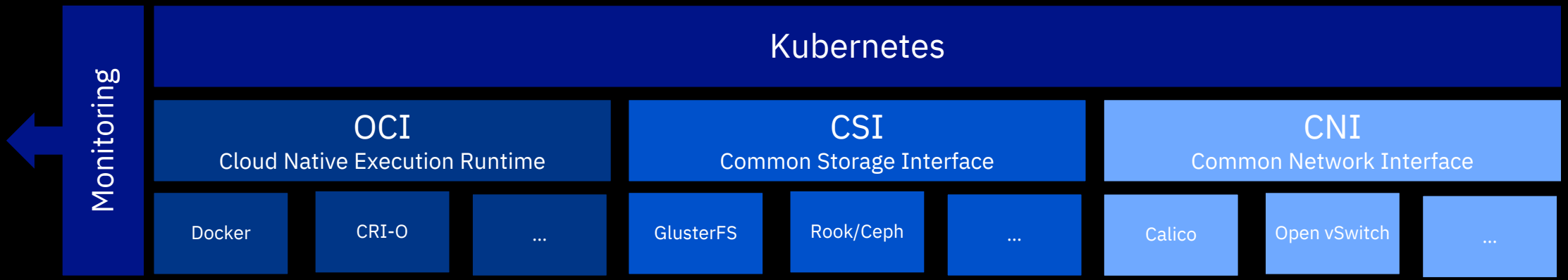
Service discovery
- Knows where the containers are located
- Distributes client requests across the containers

Replication
- Ensures the right number of nodes and containers

Health management
- Replaces unhealthy containers and nodes



Kubernetes Cluster Architecture



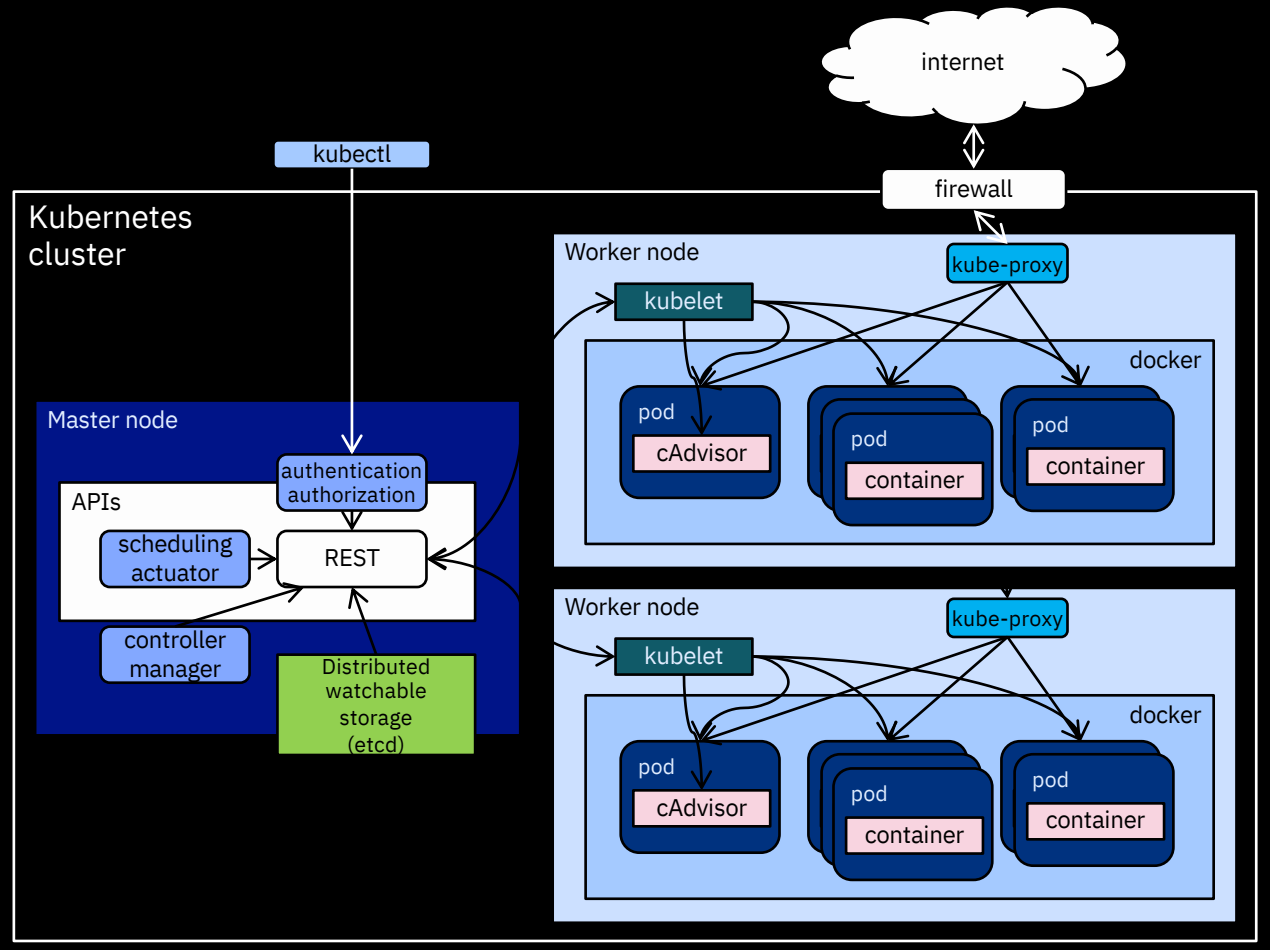
Kubernetes Cluster Architecture

Master node

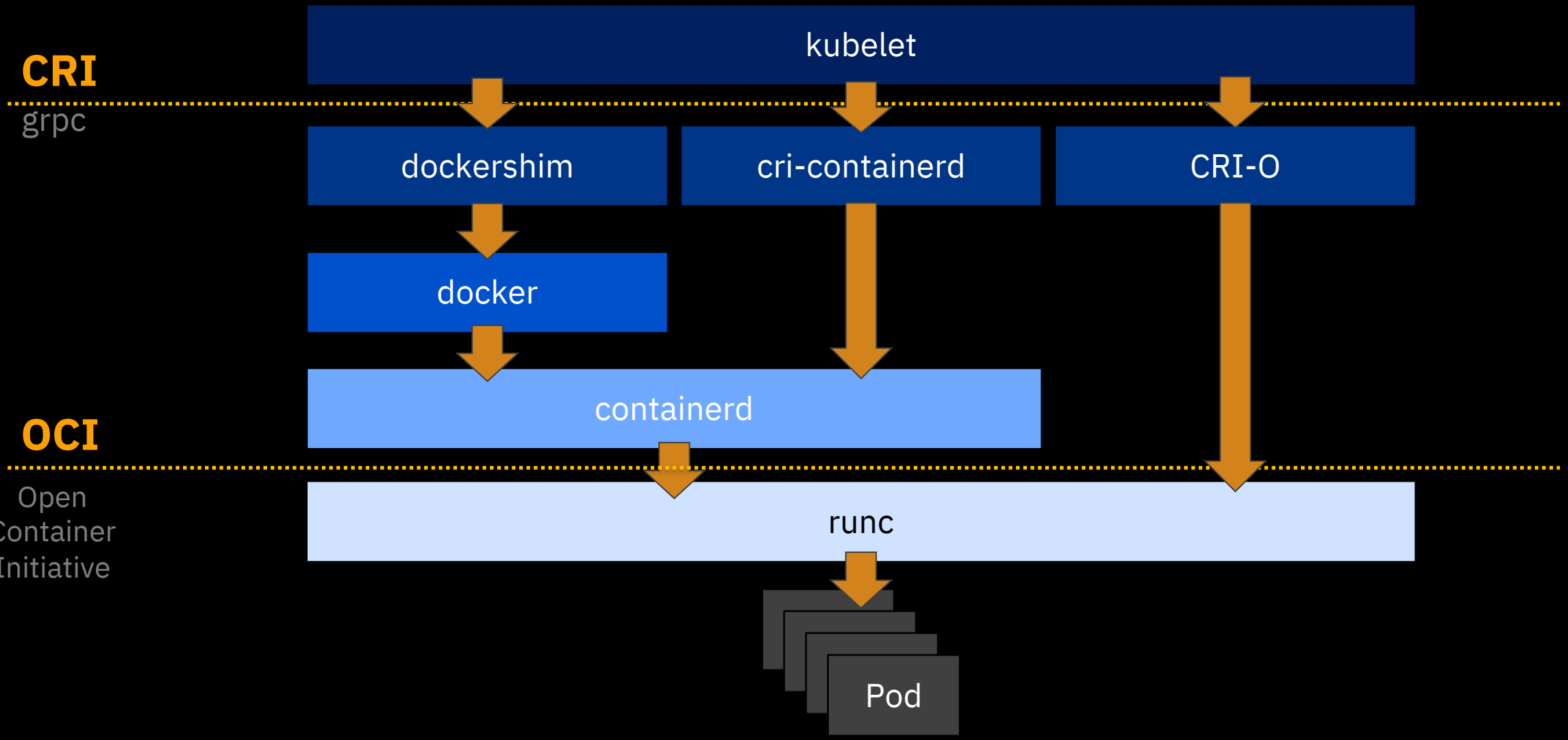
- Node that manages the cluster
- Scheduling, replication & control
- Multiple nodes for HA

Worker nodes

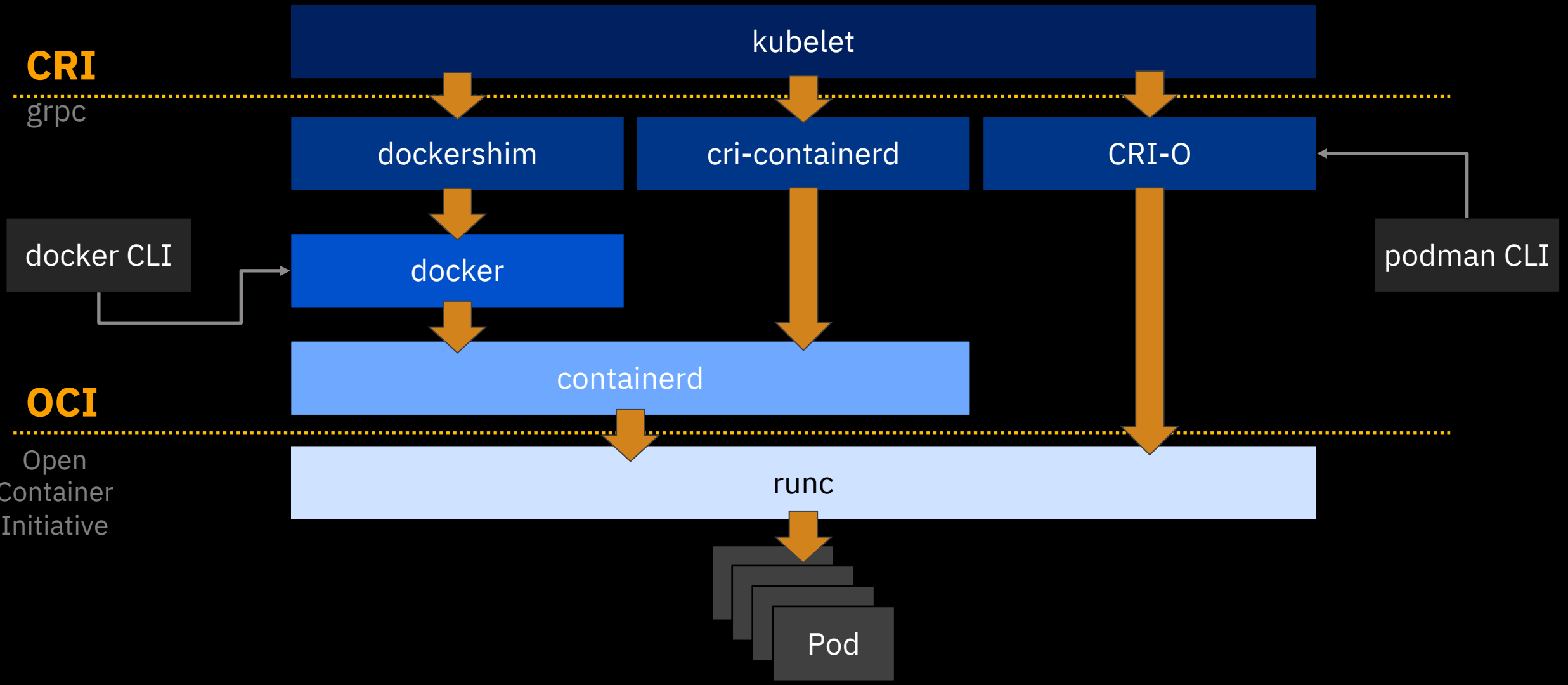
- Node where pods are run
- Docker engine
- kubelet agent accepts & executes commands from the master to manage pods
- cAdvisor – Container Advisor provides resource usage and performance statistics
- kube-proxy – routes inbound or ingress traffic



Kubernetes – Common Runtime Interface



Kubernetes – Common Runtime Interface



Master Node Components

Etcd

- A highly-available key value store
- All cluster data is stored here

API Server

- Exposes API for managing Kubernetes
- Used by kubectrl CLI

Controller manager

- Daemon that runs controllers, which are the background threads that handle routine tasks in the cluster
- Node Controller – Responsible for noticing and responding when nodes go down
- Replication Controller – Replaced by ReplicaSet
- Endpoints Controller – Populates the Endpoints object (that is, joins services and pods)
- Service Account & Token Controllers – Create default accounts and API access tokens for new namespaces

Scheduler

- Selects the worker node each pods runs in

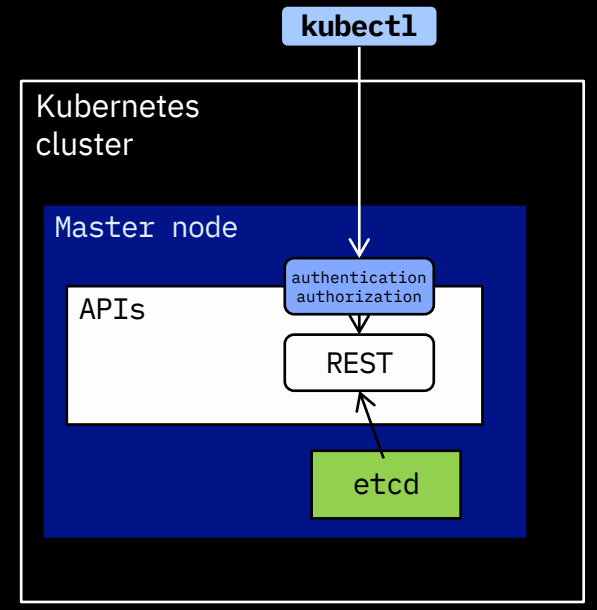
kubectl – talking to the Cluster

`kubectl` is a command line interface for running commands against Kubernetes clusters (read state, create objects, ...).

`kubectl` looks for a file named config in the $HOME/.kube directory.

Kubernetes uses etcd as a key-value database store.
It stores the configuration of the Kubernetes cluster in etcd.
It also stores the *actual* state of the system and the *desired* state of the system in etcd.

Anything you might read from a `kubectl get xyz` command is stored in etcd.
Any change you make via `kubectl create` will cause an entry in etcd to be updated.



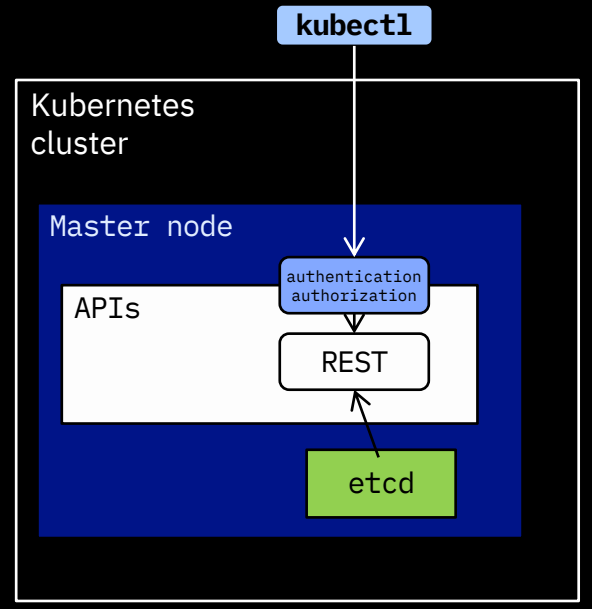
kubectl – talking to the Cluster

`kubectl` is a command line interface for running commands against Kubernetes clusters.

`kubectl` looks for a file named config in the $HOME/.kube directory.

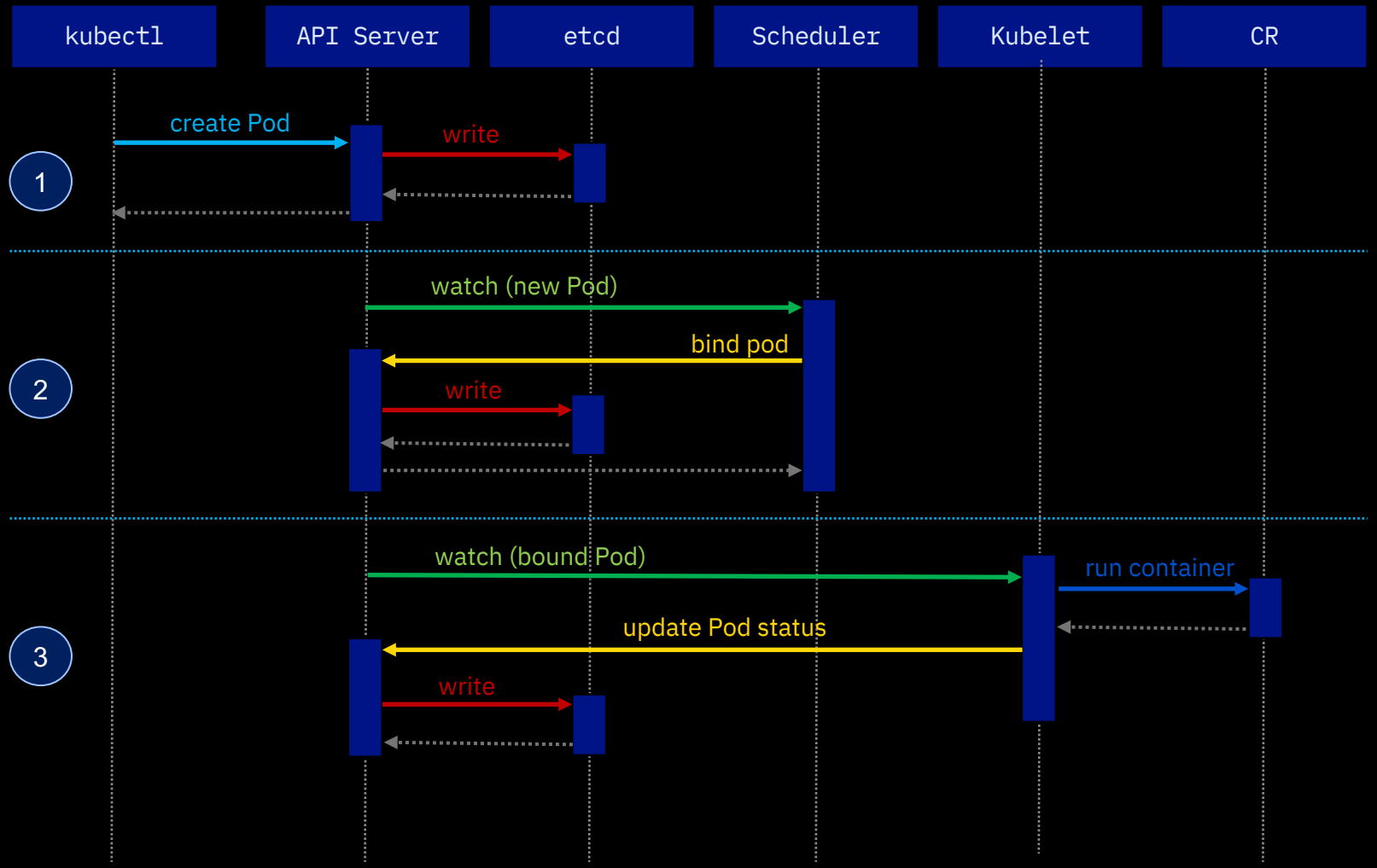
kubectl [command] [TYPE] [NAME]

kubectl create -f example.yaml	Create objects in yaml file
kubectl apply -f example.yaml	Modify objects in yaml file
kubectl delete -f example.yaml	Delete objects in yaml file
kubectl get pods	List Pods in Namespace
kubectl describe nodes <node-name>	Details about K8s object
kubectl logs <pod-name>	Get the logs for a Pod



kubectl – talking to the Cluster

What does this actually do: `kubectl run nginx --image= nginx:1.7.9`



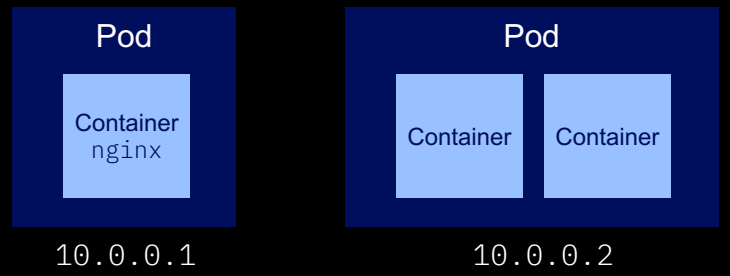
1. Create a Pod via the API Server and the API server writes it to etcd
2. The scheduler notices an “unbound” Pod and decides which node to run that Pod on. It writes that binding back to the API Server.
3. The Kubelet notices a change in the set of Pods that are bound to its node. It, in turn, runs the container via the container runtime (i.e. Docker).

The Kubelet monitors the status of the Pod via the container runtime. As things change, the Kubelet will reflect the current status back to the API Server

Pod

- A group of one or more containers is called a pod.
- Containers in a pod are deployed together, and are started, stopped, and replicated as a group.
- Containers in pod share the same network interface.
- Applications in the same pod
  - Share IP Address and port space
  - Share the same hostname
  - Can communicate using native IPC
  - Can share mounted storage
- Applications in different pods
  - Have different IP Addresses
  - Have different hostnames
  - Pods running on the same node might as well be on different servers
- **When designing pods ask, “Will these containers work correctly if they land on different machines?”**

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```



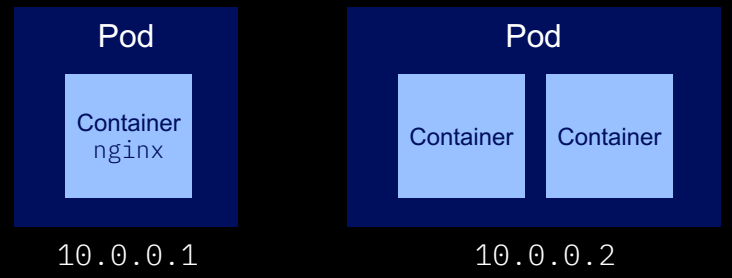
Creating a Pod

```
kubectl run nginx --image= nginx:1.7.9
or
kubectl create –f example.yaml
```

```
kubectl get pods
NAME                READY  STATUS   RESTARTS  AGE
nginx-5bd87f76c-vxc79  1/1   Running  0      29s
```

example.yaml

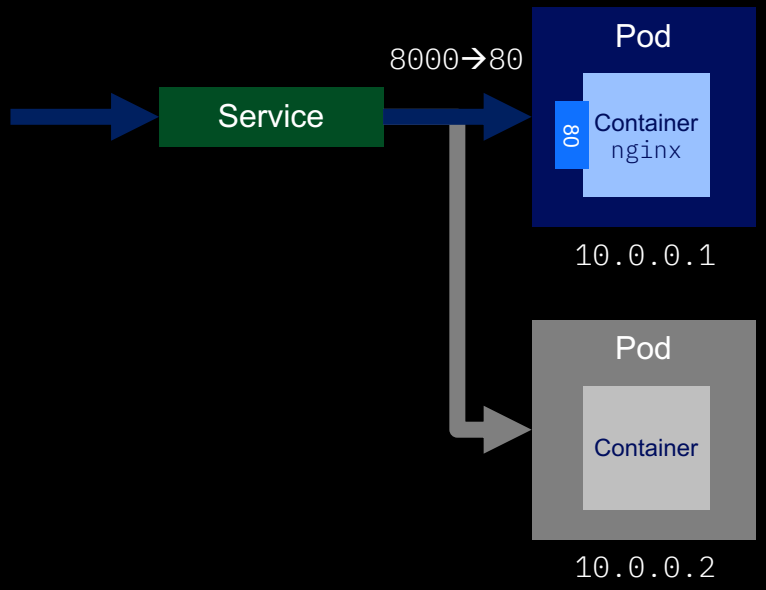
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```



Service

- A service provides a way to refer to a set of Pods (selected by labels) with a single static IP address.

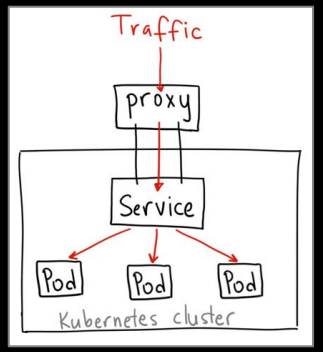
- Also provide load balancing



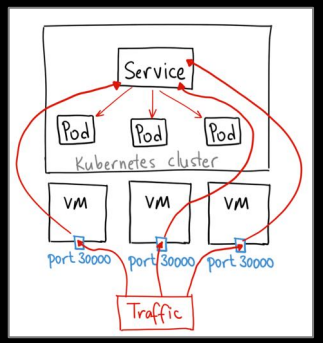

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
  selector:
    app: nginx
```

Service

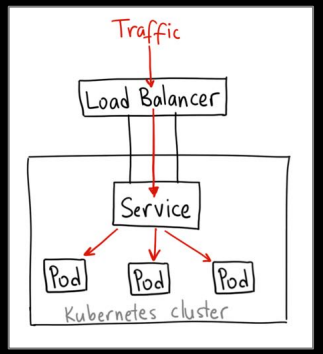
ClusterIP: This type exposes the service on the cluster internal IP. This means that the service is only reachable from within the cluster.



NodePort: This type exposes the service on each Node's static IP address.



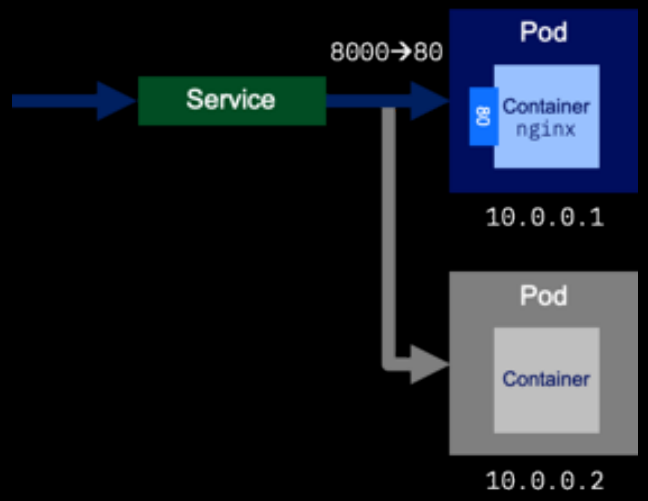
LoadBalancer: This service type exposes a service using the cloud provider load balancer.



Creating a Service

```
kubectl expose deployment nginx --type="NodePort" --port=8000 --target-port=80
or
kubectl create –f example.yaml
```

```
kubectl get services
NAME      TYPE       CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
nginx     NodePort   10.106.115.135  <none>       8000:32499/TCP   6s
```



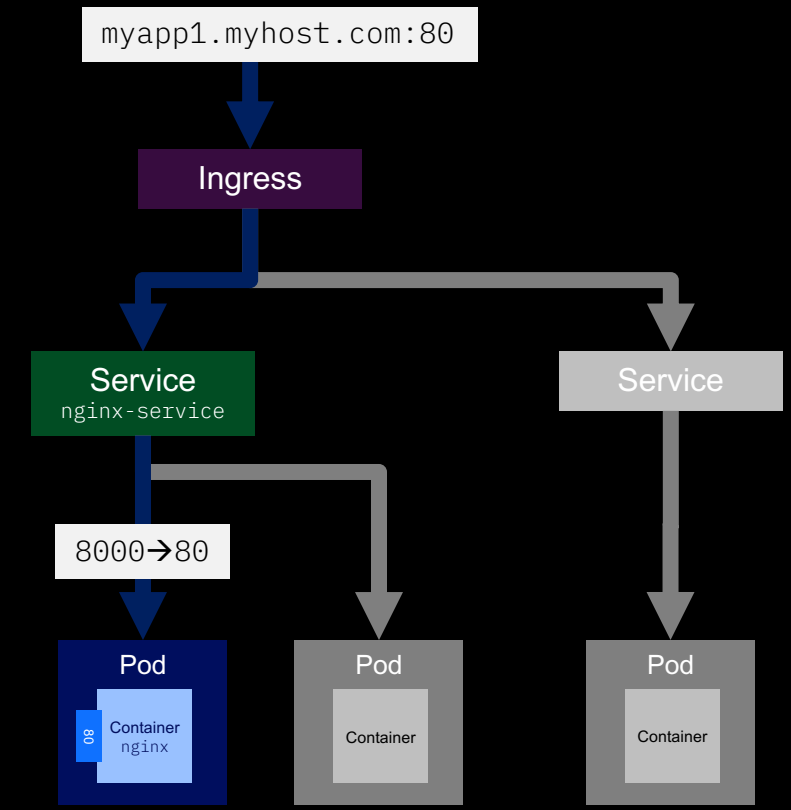
example.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
  selector:
    app: nginx
```

Ingress

An ingress can be configured to give services externally-reachable URLs, load balance traffic, terminate SSL, and offer name based virtual hosting. An ingress controller is responsible for fulfilling the ingress, usually with a loadbalancer.

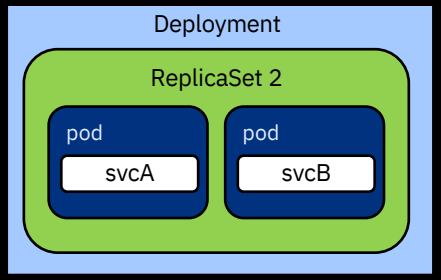
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: myapp1.myhost.com
    http:
      paths:
      - backend:
          serviceName: nginx-service
          servicePort: 80
```



Deployments & ReplicaSets

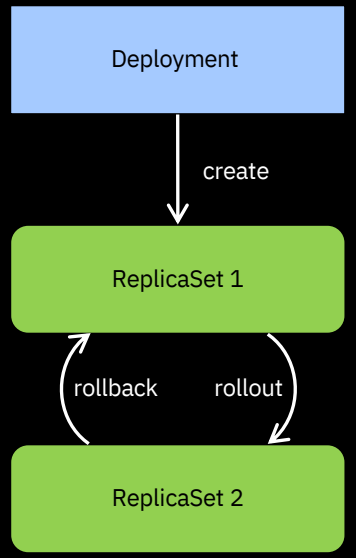
Deployment

- A set of **pods to be deployed together**, such as an application
- **Declarative**: Revising a Deployment **creates a ReplicaSet** describing the desired state
- **Rollout**: Deployment controller changes the actual state to the desired state at a controlled rate
- **Rollback**: Each Deployment revision can be rolled back
- **Scale** and autoscale: A Deployment can be scaled



ReplicaSet

- Cluster-wide pod manager that **ensures the proper number of pods are running** at all times.
- A set of pod templates that describe a set of pod replicas
- Uses a template that describes specifically what each pod should contain
- Ensures that a specified number of pod replicas are running at any given time



Deployment

- A Deployment object defines a Pod creation template and **desired replica count**.
- Create or delete Pods as needed to meet the replica count.
- Manage safely **rolling out changes** to your running Pods.

```
kubectl create –f example.yaml
```

example.yaml

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  template:
     metadata:
       labels:
         app: nginx
     spec:
       containers:
       - name: nginx
          image: nginx:1.7.9
          ports:
          - containerPort: 80
```

StatefulSet, DaemonSet, Job...

StatefulSet

- Intended to be used with stateful applications and distributed systems.
  - Pods are created sequentially
  - Ordinal index and stable network identity

DaemonSet

- Ensures that all (or some) nodes run a copy of a pod
  - running a cluster storage daemon
  - running a logs collection daemon
  - running a node monitoring daemon

Job

- Creates one or more pods and ensures that a specified number of them successfully terminate

Cron Job

- Manage time based jobs, once at a specified in time, or repeatedly at a specified time point

Naming

Name

- Each resource object by type has a unique name

Namespace

- **Resource isolation**: Each namespace is a virtual cluster within the physical cluster
  - Resource objects are scoped within namespaces
  - Low-level resources are not in namespaces: nodes, persistent volumes, and namespaces themselves
  - Names of resources need to be unique within a namespace, but not across namespaces
- **Resource quotas**: Namespaces can divide cluster resources
- Initial namespaces
  - `default` – The default namespace for objects with no other namespace
  - `kube-system` – The namespace for objects created by the Kubernetes system

Resource Quota

- Limits resource consumption per namespace
- Limit can be number of resource objects by type (pods, services, etc.)
- Limit can be total amount of compute resources (CPU, memory, etc.)
- Overcommit is allowed; contention is handled on a first-come, first-served basis

Kubernetes User Security

- **Authentication**
  - OIDC Tokens
  - ServiceAccount Tokens
- **Authorization**
  - RBAC
  - Role
  - RoleBinding
  - ClusterRole
  - ClusterRoleBinding
- **Namespaces**
  - Help to scope access control to a set of resources in a Kubernetes cluster

```
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: viewer
rules:
- apiGroups:
  - apps
  - extensions
  resources:
  - deployments
  - replicasets
  verbs:
  - get
  - list
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cicd-viewer
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: viewer
subjects:
- kind: User
  name: IAM#cicd@us.ibm.com
  apiGroup: rbac.authorization.k8s.io
```

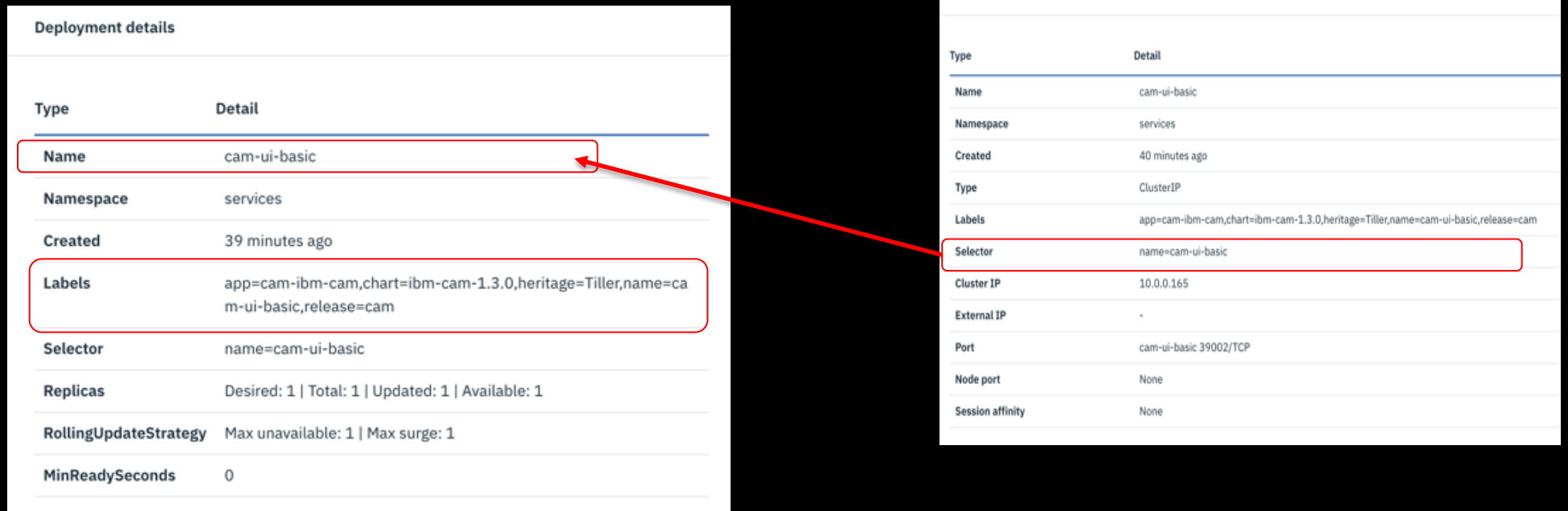
Configuring Resources and Containers

Label

- **Metadata** assigned to Kubernetes resources (pods, services, etc.)
- Key-value pairs for identification
- Critical to Kubernetes as it relies on querying the cluster for resources that have certain labels

Selector

- An expression that **matches labels** to identify related resources



Configuring Resources and Containers

ConfigMap

- Configuration values to be used by containers in a pod
- Stores configuration outside of the container image, making containers more reusable

Secret

- Sensitive info that containers need to read or consume
- Encrypted in special volumes mounted automatically

Configuring Resources and Containers

```
apiVersion: extensions/v1beta1
kind: ConfigMap
apiVersion: v1
metadata:
  name: example-config
data:
  allowed: '"true"'
  enemies: aliens
  lives: "3"
```

```
kind: Deployment
spec:
  containers:
    - name: test-container
      image: xxx
      ...
    env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: example-config
              key: enemies
```

Can be used as normal environment variable

Kubernetes Autoscaling

Horizontal Pod Autoscaling (HPA)

- Automatically scales the number of pods in a replication controller, deployment, or replica set
- Matches the observed average CPU utilization to the specified target
- Fetches metrics in two different ways: direct Heapster access and REST client access
- Kubernetes Heapster enables container cluster monitoring and performance analysis
- Default config: query every 30 sec, maintain 10% tolerance, wait 3 min after scale-up, wait 5 min after scale-down

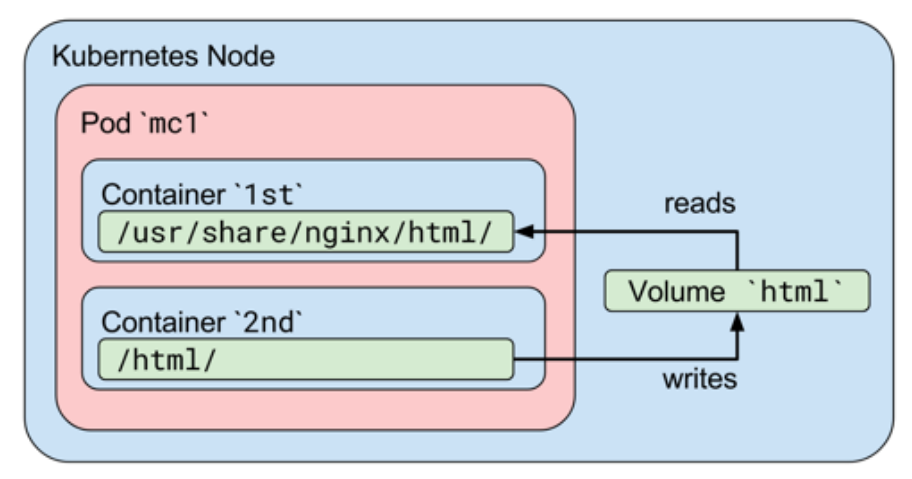
```
$ kubectl autoscale deployment <deployment-name> --cpu-percent=50
  --min=1 --max=10 deployment "<hpa-name>" autoscaled
```

Creates a horizontal pod autoscaler

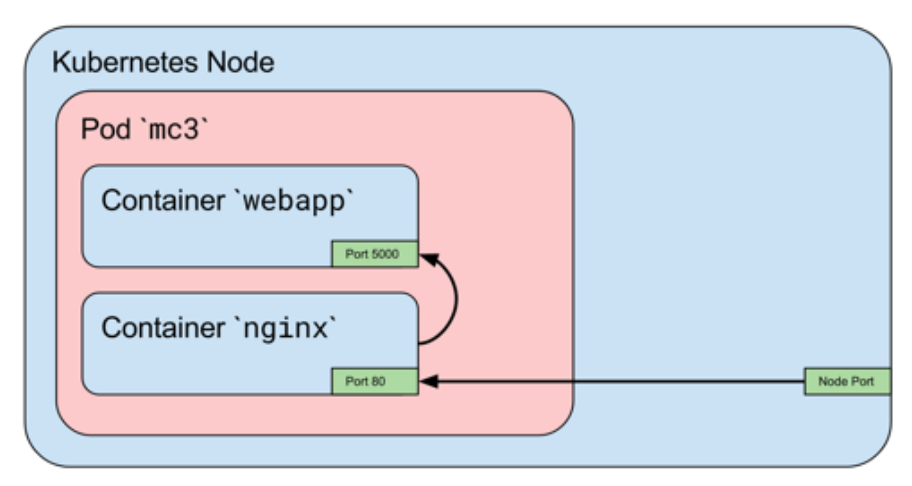
- An HPA instance
- Maintains between 1 and 10 replicas of the pods controlled by the deployment
- Maintains an average CPU utilization across all pods of 50%

Multi-Container Pod Design Patterns

Shared Volumes

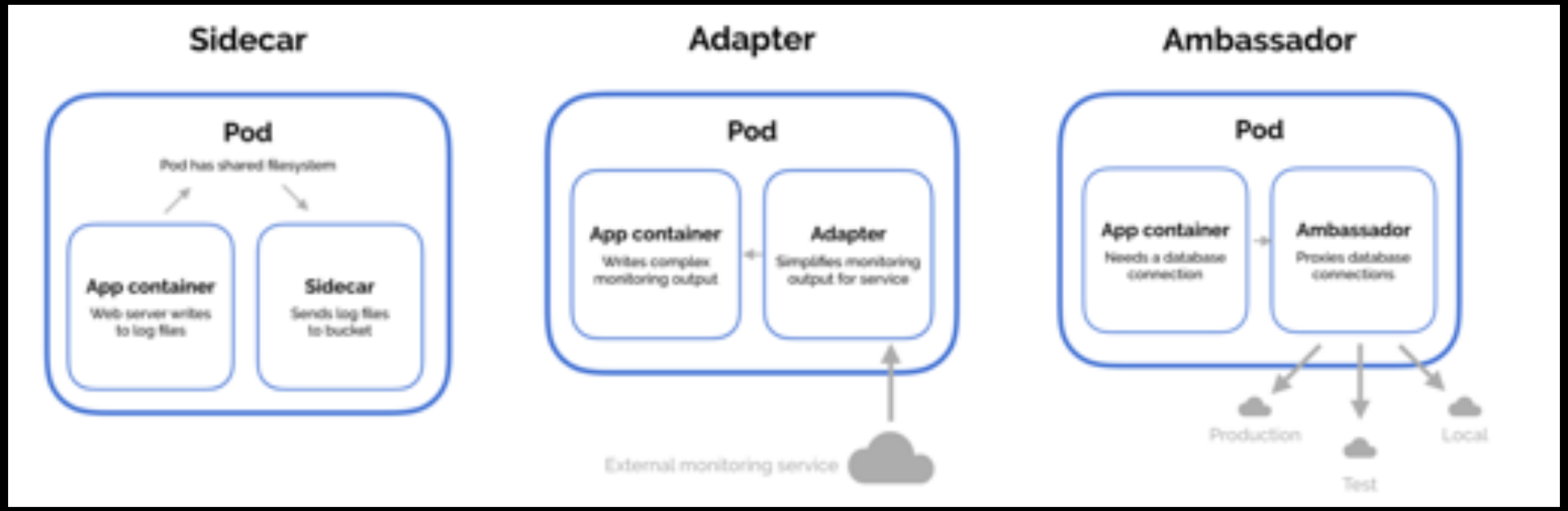


No Network isolation



In what order containers are being started in a Pod?

Multi-Container Pod Design Patterns



Sidecar pattern
The sidecar pattern consists of a main application—i.e. your web application—plus a helper container with a **responsibility that is essential to your application**, but is not necessarily part of the application itself.

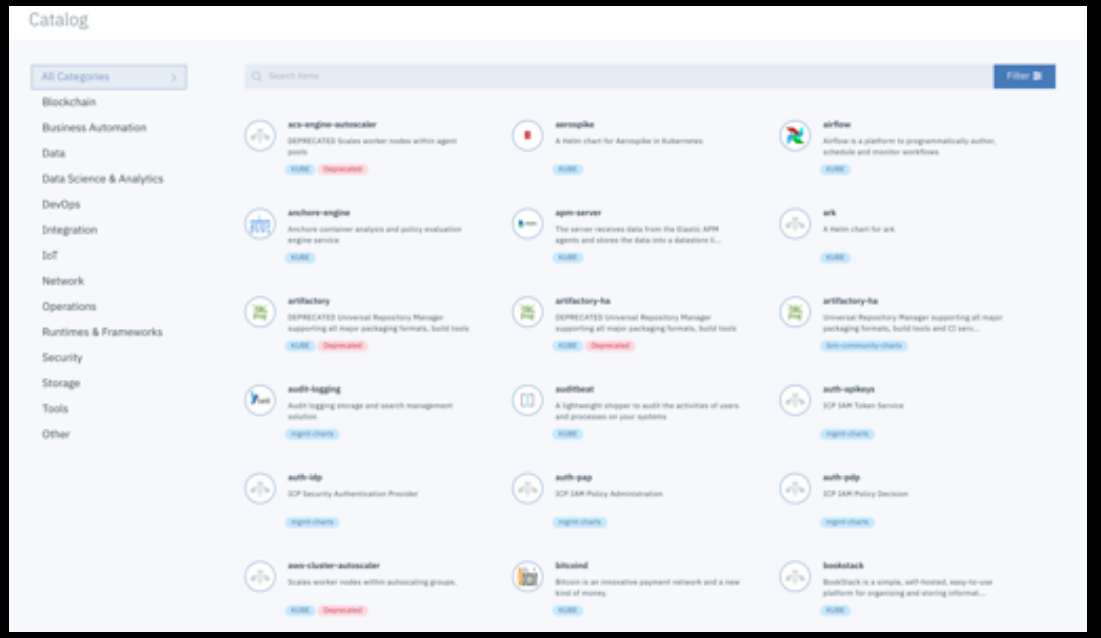
Adapter pattern
The adapter pattern is used to standardize and normalize application output or **monitoring data** for aggregation.

Ambassador pattern
The ambassador pattern is a useful way to connect containers with the outside world (**proxy**).

https://matthewpalmer.net/kubernetes-app-developer/articles/multi-container-pod-design-patterns.html

HELM Charts

- Helm is the open standard for **Application Packaging and Deployment** for Kubernetes (like npm, yum or apt-get)
- Helm charts **automate the deployment of resources** and prerequisites including locations of Docker images

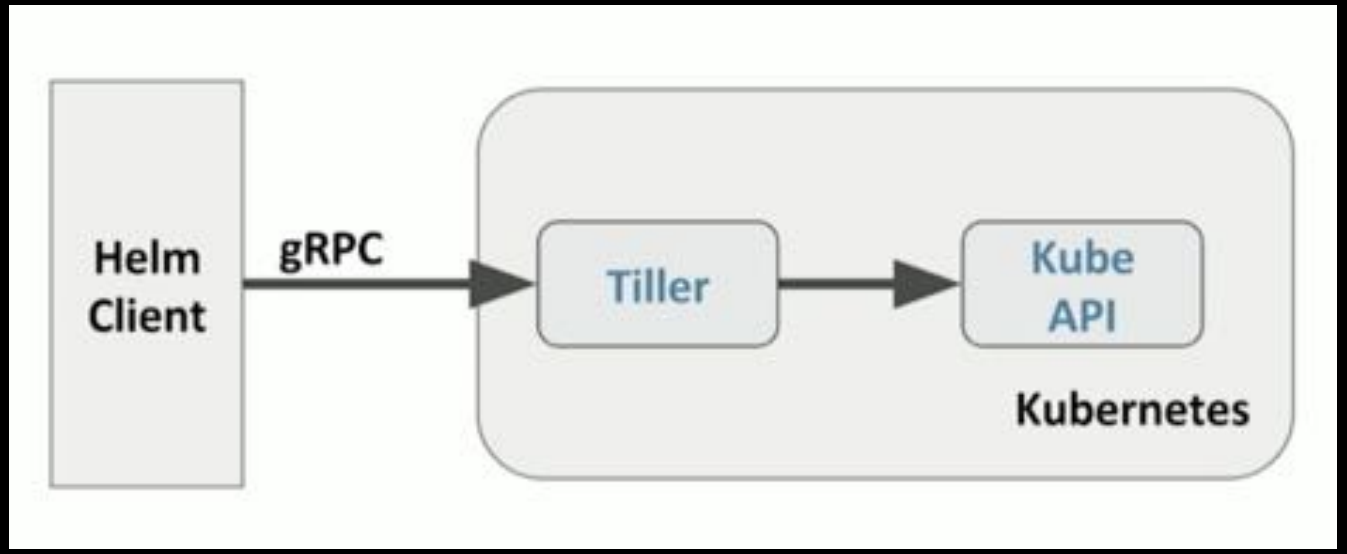


HELM Example

Helm has 2 Components :

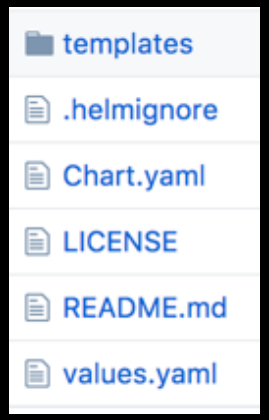
Tiller : Tiller is the in-cluster component of Helm.
It interacts directly with the Kubernetes API server to install, upgrade, query, and remove Kubernetes resources. It also stores the objects that represent releases.

Client : The Helm client interacts with the Tiller and stores information in a local directory.



HELM Example

- Chart :
  A Helm package that contains information sufficient for installing a set of Kubernetes resources into a Kubernetes cluster.
  - Chart.yaml
    Basic information about the chart (name, versions, links, icon, ...)
  - Templates (yaml)
    Kubernetes deployment manifests (yaml) with placeholders for external parameters.
  - values.yaml
    A set of variables that will show up in the UI (or customized at the helm command line) and be passed on at deploy time to the deployment manifests.



Deployment Chart

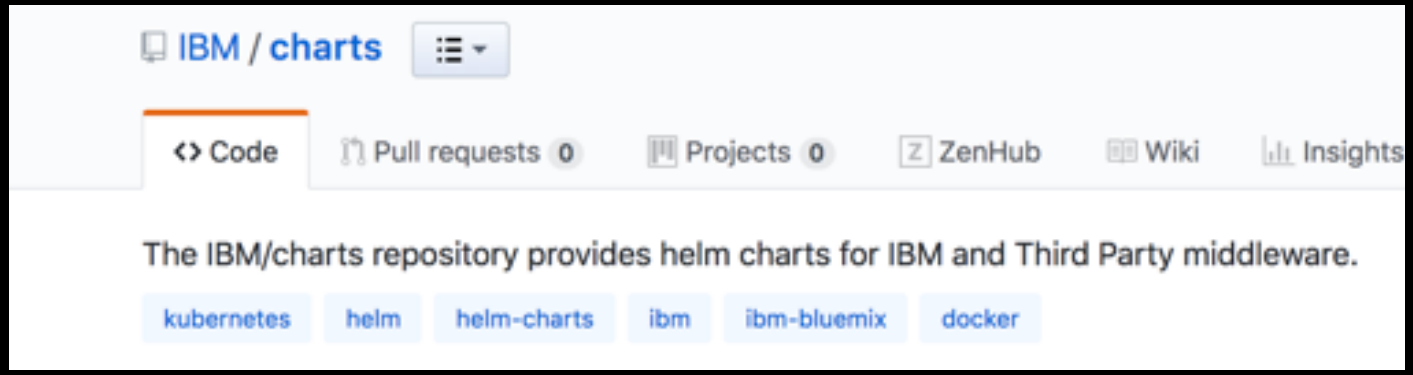
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: {{ template "fullname" . }}-pv-claim
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: {{ .Values.storage.size }}
```

Values.yaml file

```
storage:
  size: 2Gi
```

IBM Helm Charts

https://github.com/ibm/charts



Kubectl Commands

Get the state of your cluster

```
$ kubectl cluster-info
```

Get all the nodes of your cluster

```
$ kubectl get nodes -o wide
```

Get info about the pods of your cluster

```
$ kubectl get pods -o wide
```

Get info about the replication controllers of your cluster

```
$ kubectl get rc -o wide
```

Get info about the services of your cluster

```
$ kubectl get services
```

Get full config info about a Service

```
$ kubectl get service NAME_OF_SERVICE -
o json
```

Get the IP of a Pod

```
$ kubectl get pod NAME_OF_POD -
template={{.status.podIP}}
```

Delete a Pod

```
$ kubectl delete pod NAME
```

Delete a Service

```
$ kubectl delete service
NAME_OF_SERVICE
```

Resources

Linux Foundation – Introduction to Kubernetes

– https://courses.edx.org/courses/course-v1:LinuxFoundationX+LFS158x+2T2017/course/

Kubernetes tutorial

– https://kubernetes.io/docs/tutorials/kubernetes-basics/

Introduction to container orchestration

– https://www.exoscale.ch/syslog/2016/07/26/container-orch/

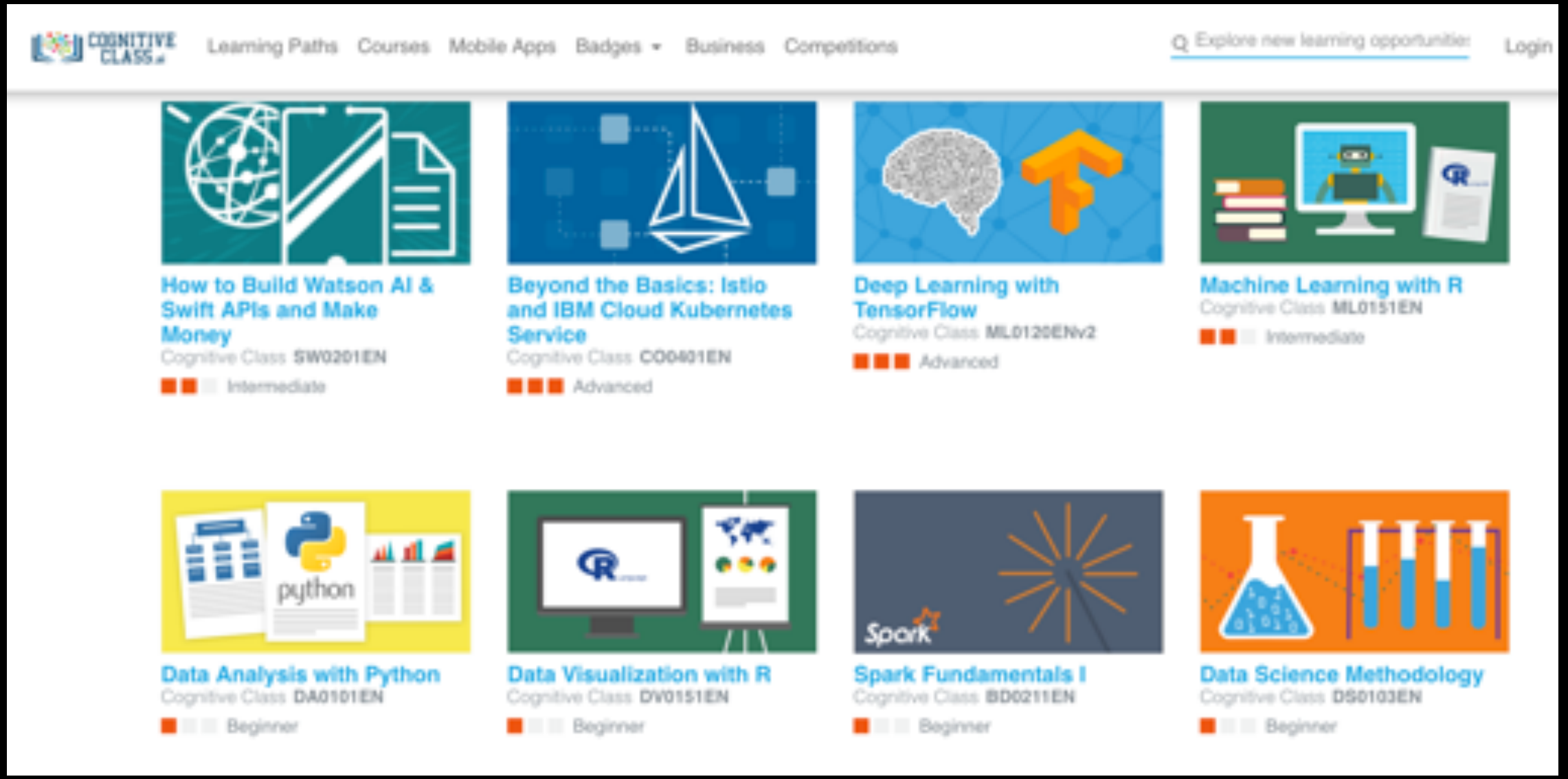
TNS Research: The Present State of Container Orchestration

– https://thenewstack.io/tns-research-present-state-container-orchestration/

Large-scale cluster management at Google with Borg

– https://research.google.com/pubs/pub43438.html

Cognitiveclass.ai – Free basic to advanced classes on AI, Cloud... etc



Benefits of using a Kubernetes Based Platform

Speed

- Fast boot-time
- Easy scalability
- Rapid deployment

13x

More software releases

Portability

- Between different environments
- Between private and public cloud

Eliminate

“works on my machine” syndrome

Efficiency

- Better usage of computer resources

~47%

Reduction of VMs, OS licensing and server cost

Automation

- Next level standardization and automation

Reduce

operation cost

IBM

QUESTIONS?

IBM **Cloud**

The Journey to Cloud

Let‘s get real

04

IBM **Cloud**

Microservice transformations – The good, the bad, the ugly

The good

IF implemented correctly Microservices improve scalability and reliability.

- Scale easily and very efficiently
- No single point of failure by isolation
- Be modified without compromising the whole system
- Use of languages and frameworks that best suit the purpose of each service
- Natively ready for CaaS and PaaS

Source: https://caylent.com/microservices-good-bad-ugly/

Microservice transformations – The good, the bad, the ugly

The bad

- Added management complexity
- Expertise required to maintain a microservice-based application
- No or reduced benefits if application doesn’t need scaling or cloud transformation
- Needs more robust testing in order to cover all APIs

Source: https://caylent.com/microservices-good-bad-ugly/

Microservice transformations – The good, the bad, the ugly

… and **the downright ugly**

- May require high initial investment to run for Dev, deployment and Ops overhead
- Higher initial operational and monitoring costs through shift in paradigms
- Non-uniform application design and architecture through free choice of technologies
- Overload of documentation for every individual component app

Source: https://caylent.com/microservices-good-bad-ugly/

Microservices, when implemented incorrectly, can make poorly written applications even more dysfunctional

Microservices – breaking down the ~~monolith~~ beast

How to identify candidates

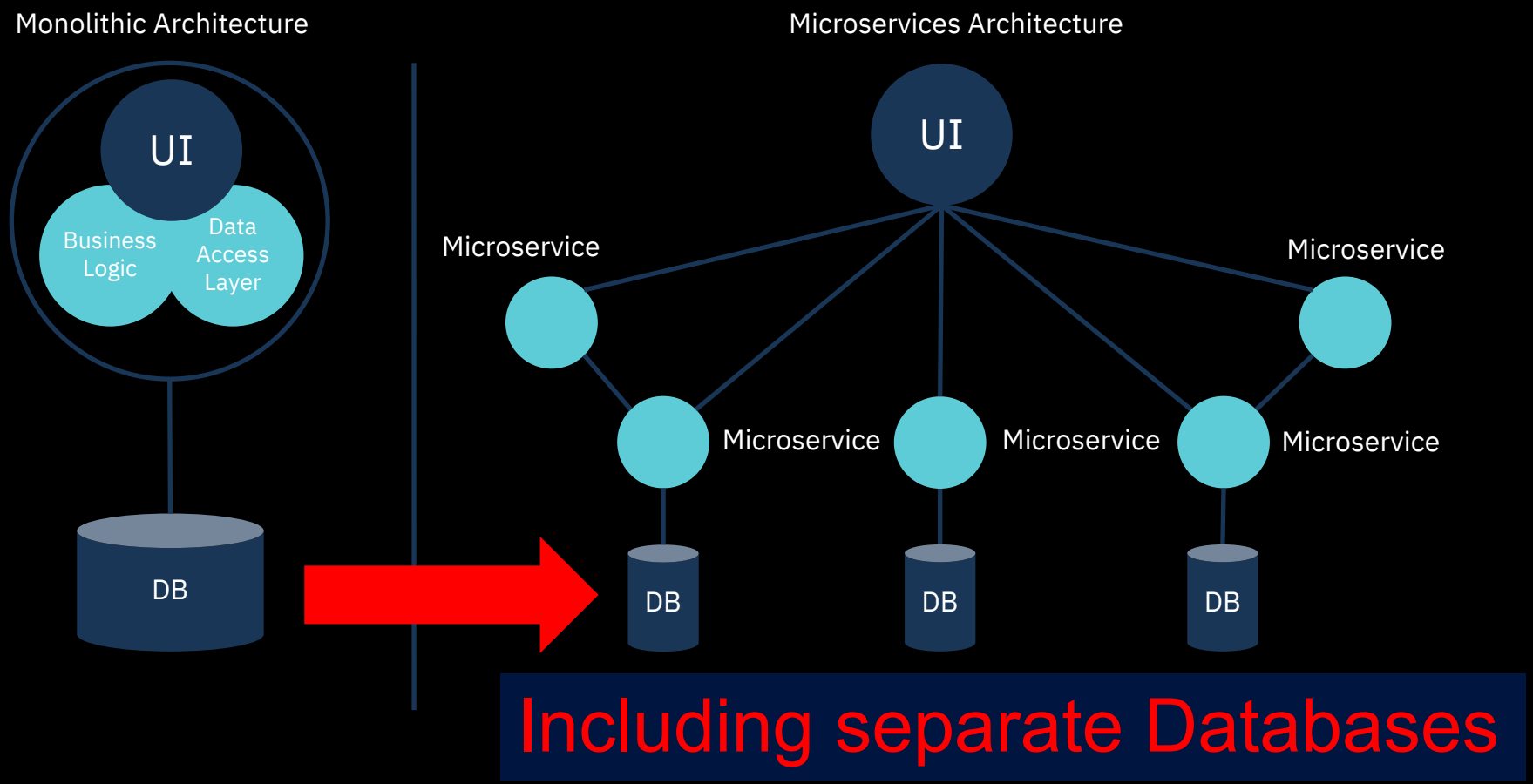
- Is there anything that is scaling at a differently than the rest of the system?
- Is there anything that feels “tacked-on”?
- Is there anything changing much faster than the rest of the system?
- Is there anything requiring more frequent deployments than the rest of the system?
- Is there a part of the system that a small team, operates independently?
- Is there a subset of tables in your datastore that isn’t connected to the rest of the system?

Source: https://www.simplethread.com/youre-not-actually-building-microservices/

Databases

Microservices – Database refactoring

Simplistically, microservices architecture is about breaking down large silo applications into more manageable, fully decoupled pieces



Microservices – Database refactoring

Containerizing your WAR file doesn’t mean you’re doing microservices.

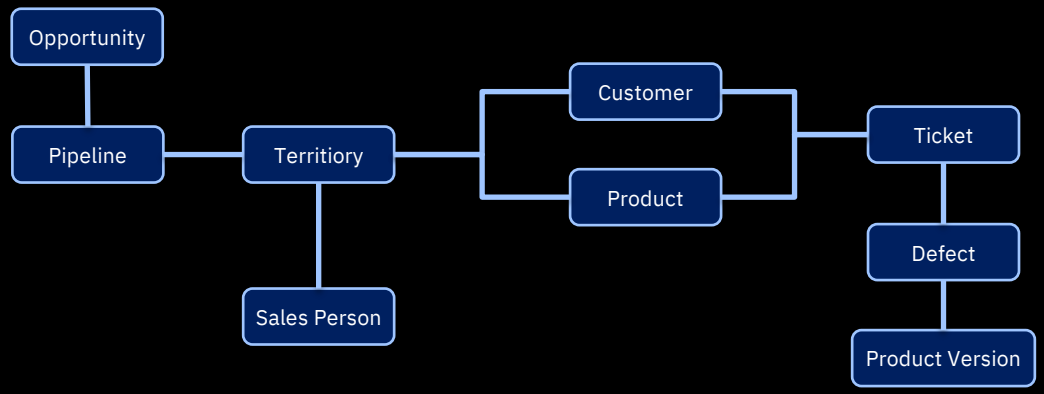
- What is the domain? What is reality?
- Where are the transactional boundaries?
- How should microservices communicate across boundaries?
- What if we just turn the database inside out?

Source: https://martinfowler.com/bliki/BoundedContext.html

Microservices – Bounded Context

Bounded Context is a central pattern in Domain-Driven Design

- DDD deals with large models by dividing them into different Bounded Contexts

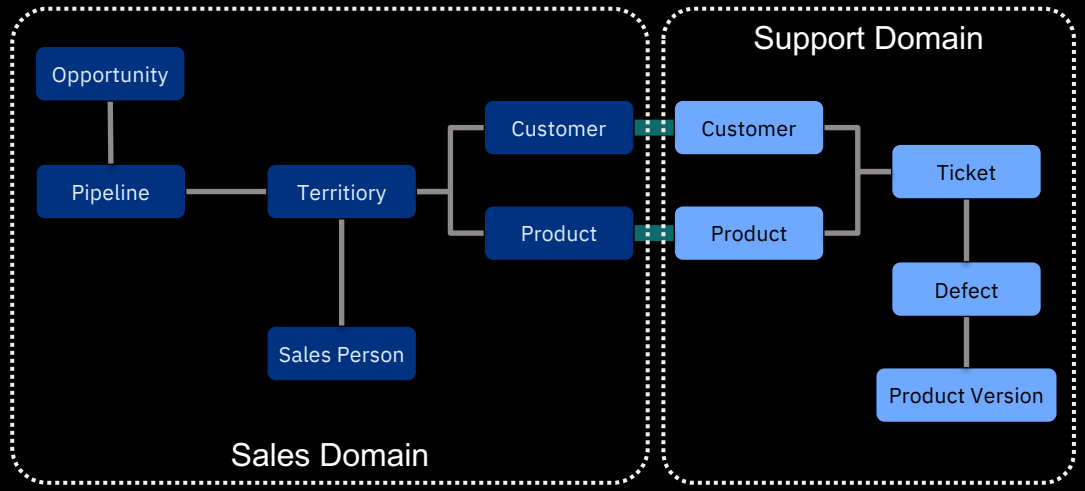


Source: https://martinfowler.com/bliki/BoundedContext.html

Microservices – Bounded Context

Bounded Context is a central pattern in Domain-Driven Design

- DDD deals with large models by dividing them into different Bounded Contexts
- Helps to build and refine a model that represents our domains, contained within a boundary that defines our context.

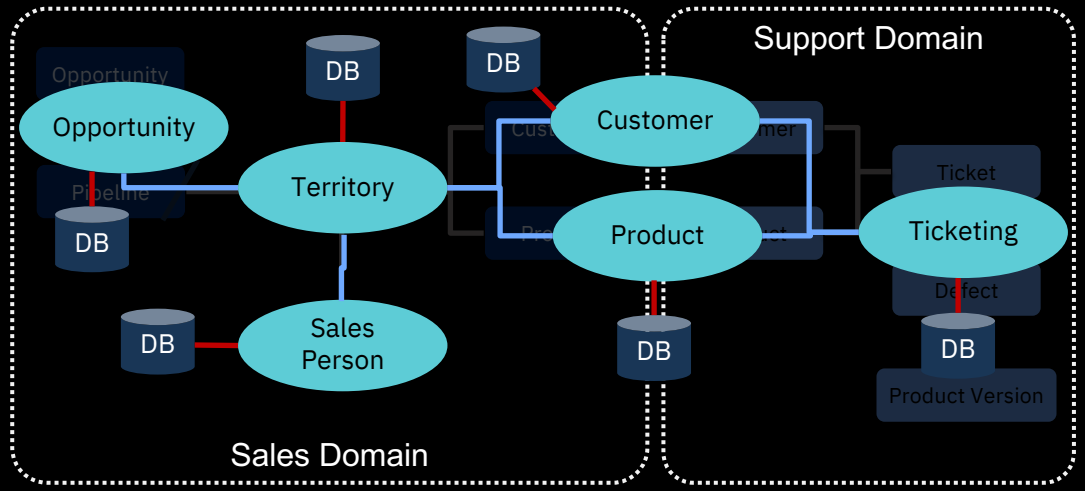


Source: https://martinfowler.com/bliki/BoundedContext.html

Microservices – Bounded Context

Bounded Context is a central pattern in Domain-Driven Design

- DDD deals with large models by dividing them into different Bounded Contexts
- Helps to build and refine a model that represents our domains, contained within a boundary that defines our context.
- These boundaries end up being our microservices

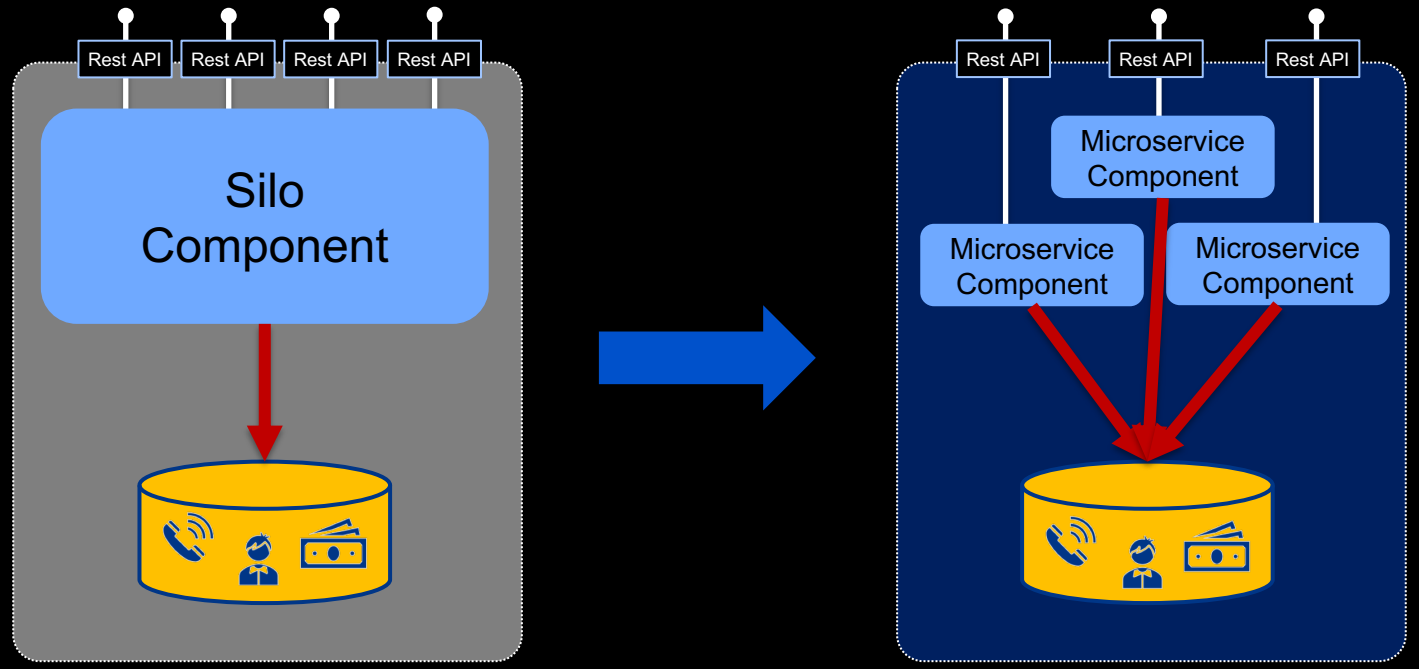


Source: https://martinfowler.com/bliki/BoundedContext.html

Microservices – Database refactoring

Data Isolation

Each microservice must keep its own data. One typical error that happens in the beginning of a microservice transformation is the use of a centralized database, the same way it was used in the monolithic application. This creates a single point of failure and dependency between the microservices.

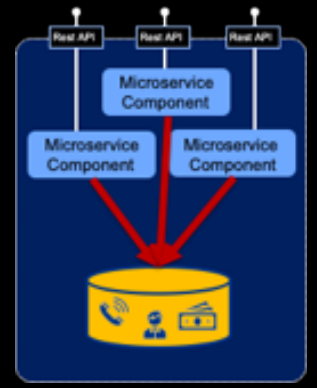


Source: IBM Architecture Center

Microservices – monolithic database

Challenges

- Inability to deploy your service changes independently through tight coupling.
- Schema changes need to be coordinated amongst all the services
- Difficult to scale individual services
- Difficult to improve application performance (DB/Table size)
- All your services have to use a relational database even when a no-SQL datastore would bring benefits

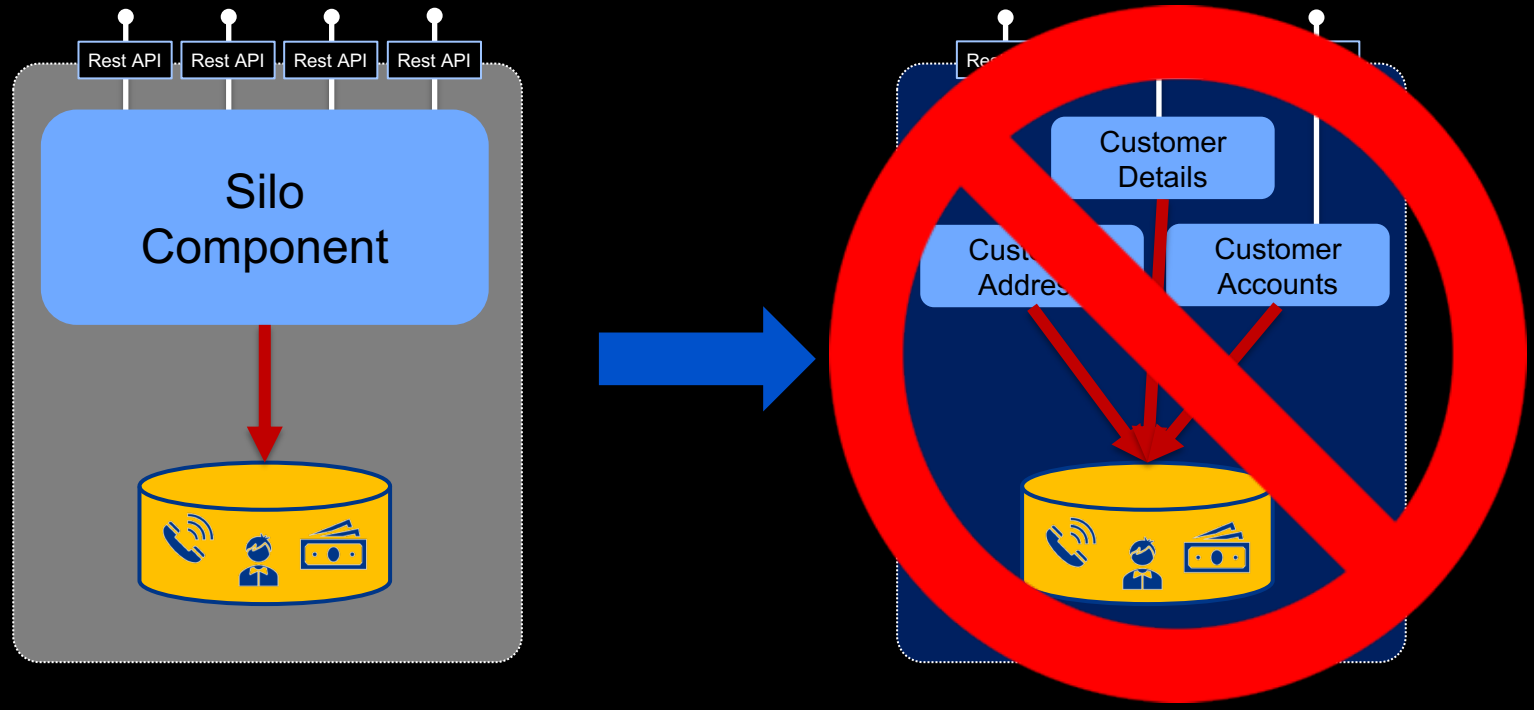


Source: https://dzone.com/articles/breaking-the-monolithic-database-in-your-microserv

Microservices – Database refactoring

Data Isolation

Each microservice must keep its own data. One typical error that happens in the beginning of a microservice transformation is the use of a centralized database, the same way it was used in the monolithic application. This creates a single point of failure and dependency between the microservices.

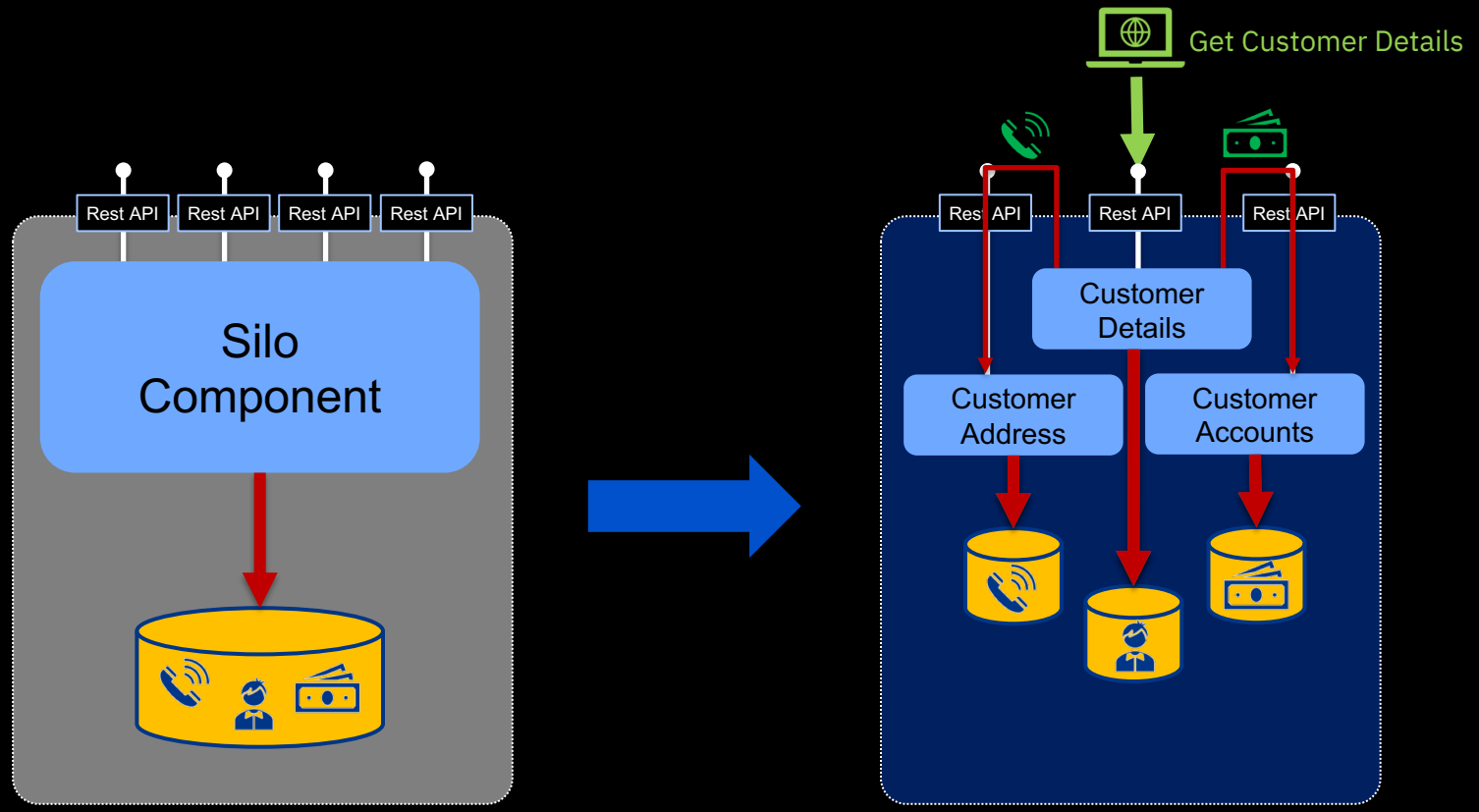


Source: IBM Architecture Center

Microservices – Database refactoring

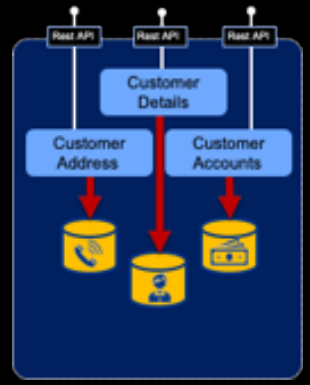
Data Isolation

Each microservice must keep its own data. One typical error that happens in the beginning of a microservice transformation is the use of a centralized database, the same way it was used in the monolithic application. This creates a single point of failure and dependency between the microservices.



Source: IBM Architecture Center

Microservices – per-service database

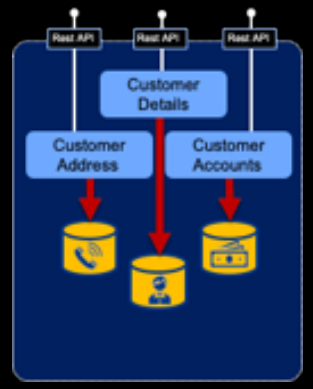


Benefits

- Avoid unexpected data modification – the API enforces consitency
- Make changes to our system without blocking progress of other parts of the system
- Store the data in a project-owned databases, using the appropriate technology
- Make changes to the schema/databases at our leisure
- Become much more scalable, fault tolerant, and flexible

Source: https://blog.christianposta.com/microservices/the-hardest-part-about-microservices-data/

Microservices – per-service database



Drawbacks

- Costly to refactor
- Needs robust transaction handling by the service (cost!)
- More difficult to debug
- More difficult to operationalize
- More difficult to test (needs well designed API tests)

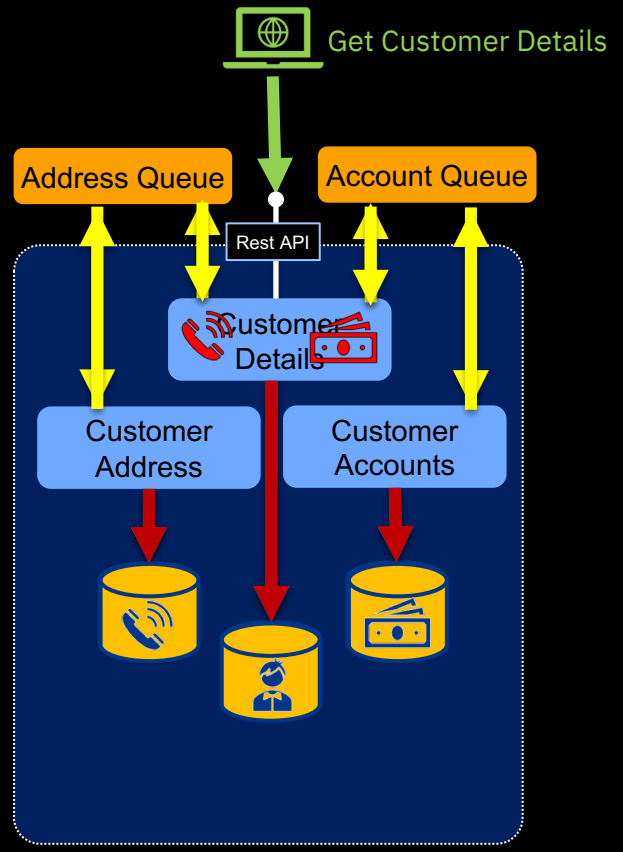
Source: https://blog.christianposta.com/microservices/the-hardest-part-about-microservices-data/

Event Driven Architecture

Microservices – one step further

Event-Driven Architecture

- Common pattern to maintain data consistency across different services.
- Avoid waiting for ACID transactions to complete
- Makes your application more available and performant
- Provides loose coupling between services



Source: https://dzone.com/articles/breaking-the-monolithic-database-in-your-microserv

Microservices

Good or bad idea?

Microservices – reasons not to use them

Some thoughts

- If speed is your goal, microservices aren't the solution (MVP, doesn’t have to scale)
- At what point is it in its lifecycle? Mission-critical?
- A moderately large, moderately complex application being maintained by a relatively small development and operations team
- Not all applications are large enough to break down into microservices
- Applications that require tight integration between individual components and services (real-time processing, ...)

Source: https://blog.christianposta.com/microservices/when-not-to-do-microservices/

Microservices – reasons not to use them

How to detect antipatterns

- A change to one microservice often requires changes to other microservices
- Deploying one microservice requires other microservices to be deployed at the same time
- Your microservices are overly chatty
- The same developers work across a large number of microservices
- Many of your microservices share a datastore
- Your microservices share a lot of the same code or models

Source: https://www.simplethread.com/youre-not-actually-building-microservices/

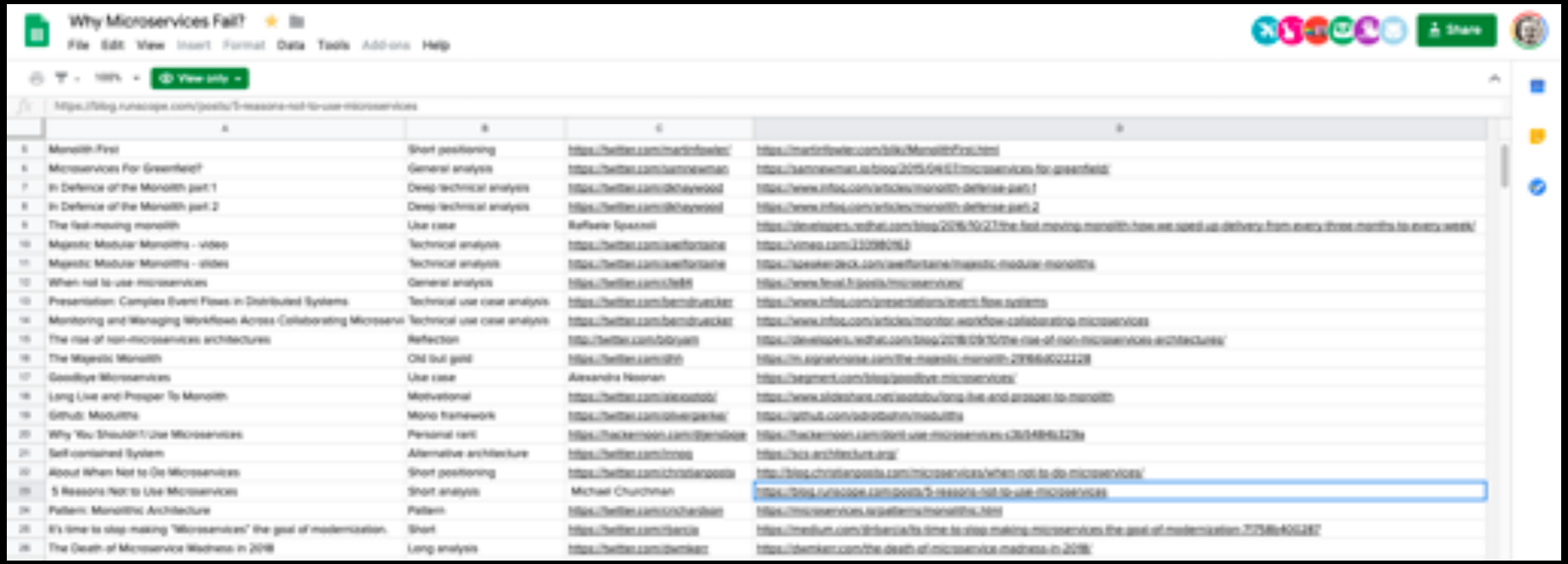
Microservices – reasons not to use them

What works for **large and complex** applications
does not always work at a **smaller scale**

What makes sense for a **new application**
does not always make sense when maintaining or
updating **existing applications**

Source: https://blog.runscope.com/posts/5-reasons-not-to-use-microservices

Microservices – reasons not to use them



https://docs.google.com/spreadsheets/d/1vjnjAII_8TZBv2XhFHra7kEQzQpOHSZpFIWDjynYYf0/edit?pli=1#gid=0

IBM

QUESTIONS?

IBM **Cloud**

The Journey to Cloud
Docker - *Hands-On*

05

IBM **Cloud**

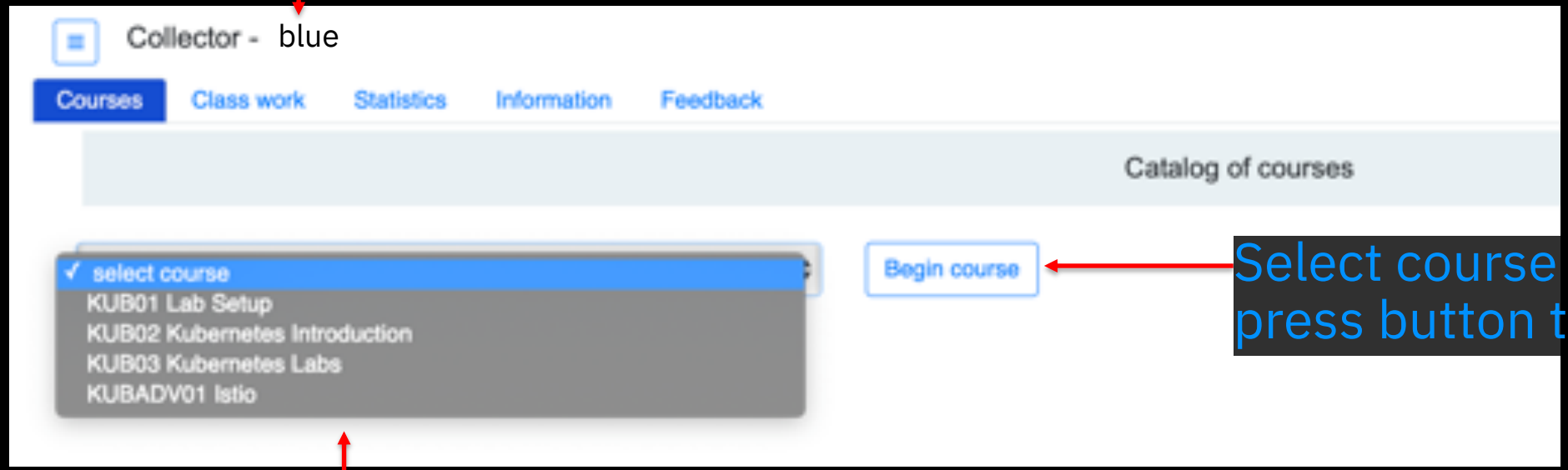
Remember your Team Color

Collector - Accessing team web site

http://158.177.137.195:{port#}

blue 31704

Team name / color will be shown



JTC01 Docker Labs

Lab 1:

- Creating images
- Run the Images
- Use Portainer tool
- Use private registries

JTC01
Docker
Labs

READY
SET
GO!!!!

http://158.177.137.195:{port#}

Duration: 60 mins

IBM

QUESTIONS?

IBM **Cloud**

Sources and documentation will be available here:

https://github.com/niklaushirt/training

The Journey to Cloud
Kubernetes - *Hands-On*

06

IBM **Cloud**

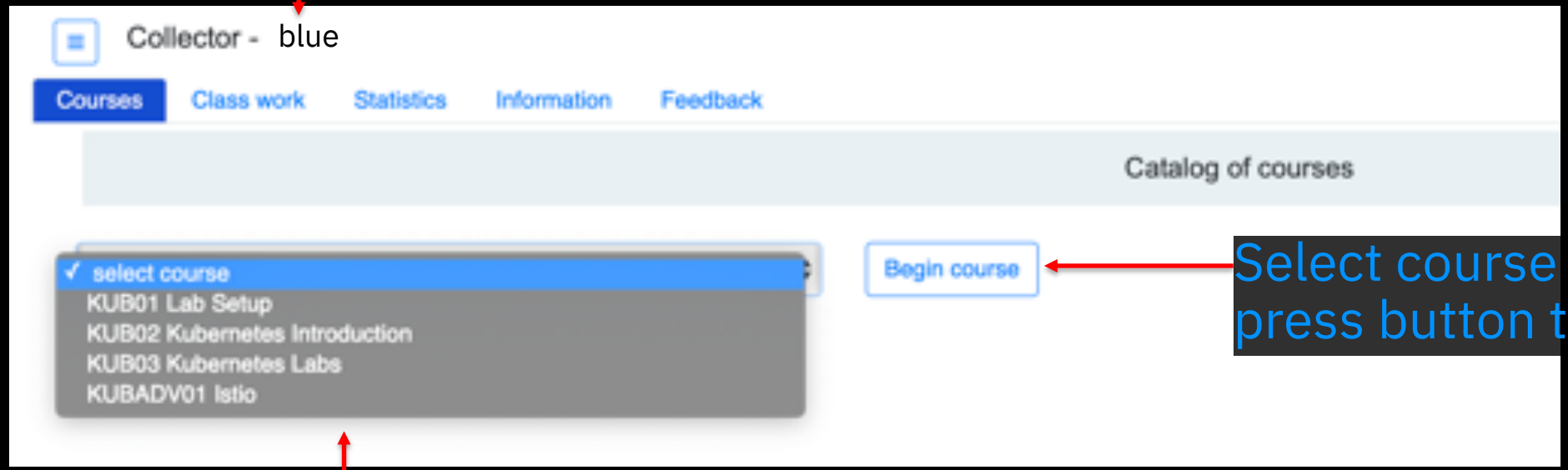
Remember your Team Color

Collector - Accessing team web site

http://158.177.137.195:{port#}

blue 31704

Team name / color will be shown



JTC02 Kubernetes Labs

Lab 1: Overview Kubernetes

Lab 2: Running your first Pod on Kubernetes

Lab 3: Deploy a Web Application on Kubernetes

Lab 4: Scale an application on Kubernetes.

Lab 5: Persisting data in Kubernetes with Volumes

JTC02
Kubernetes
Labs

READY SET GO!!!!

http://158.177.137.195:{port#}

Duration: 90 mins

IBM

QUESTIONS?

IBM **Cloud**

Sources and documentation will be available here:

https://github.com/niklaushirt/training

IBM

The Journey to Cloud

Kubernetes Security

07

IBM Cloud

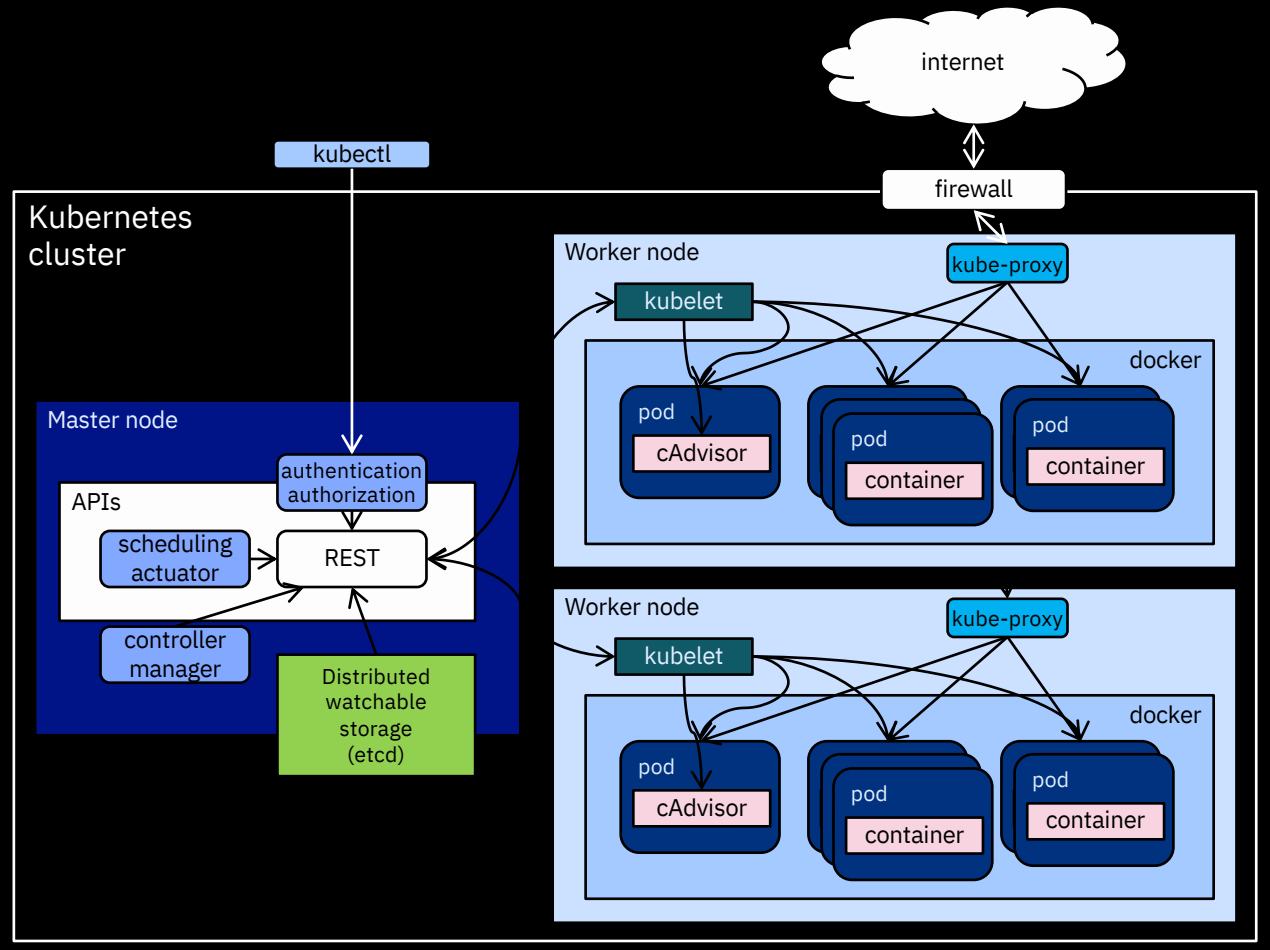
Kubernetes – **Security – Back to basics**

Master node

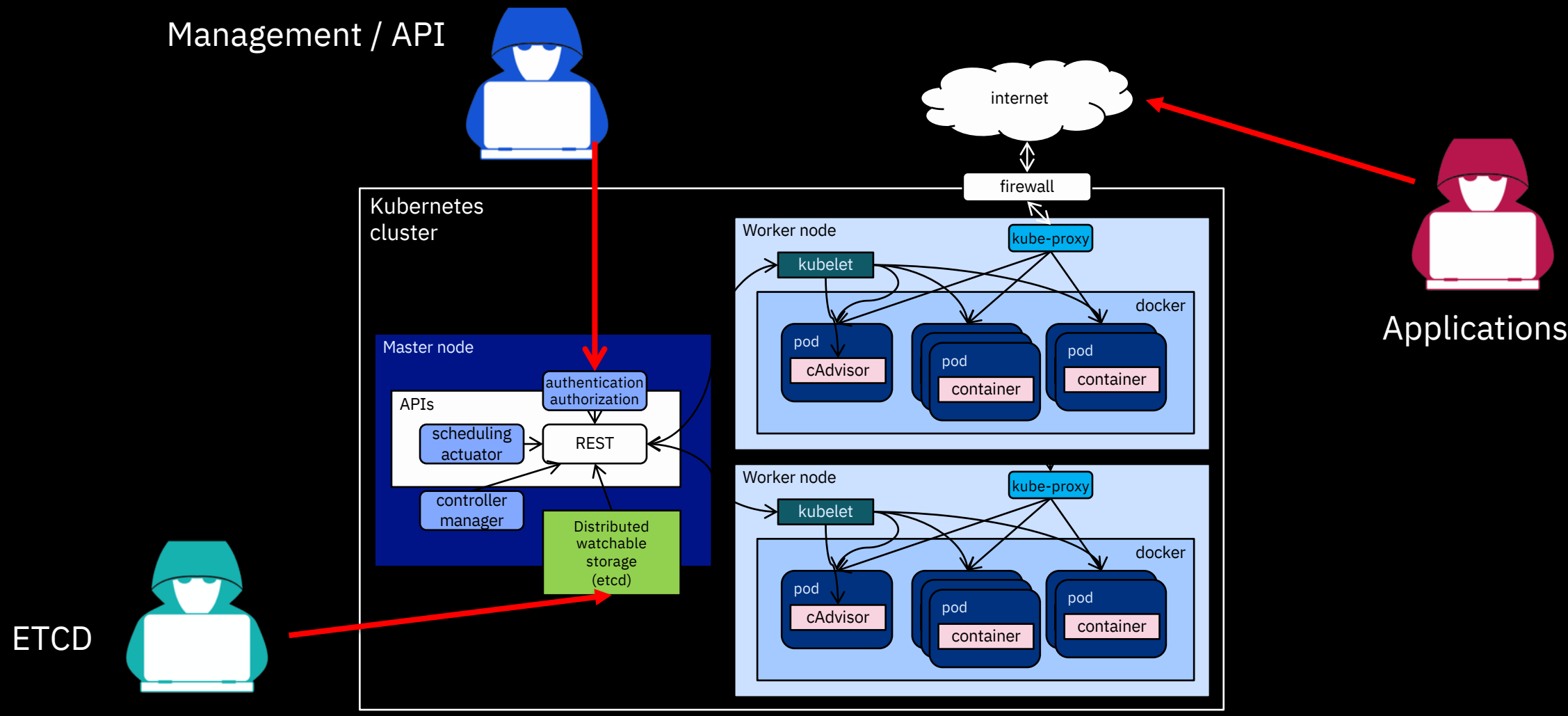
- Node that manages the cluster
- Scheduling, replication & control
- Multiple nodes for HA

Worker nodes

- Node where pods are run
- Docker engine
- kubelet agent accepts & executes commands from the master to manage pods
- cAdvisor – Container Advisor provides resource usage and performance statistics
- kube-proxy – routes inbound or ingress traffic



Kubernetes – **Security – Attack surface**



Kubernetes – **Security Basic Topics**

The **Billion Laughs** attack is a particular type of denial of service (DoS) attack which is aimed specifically at XML document parsers. This attack is also referred to as an XML bomb or an exponential entity expansion attack.

It exploits the fact that nested references to nodes can grow very large when expanded. Because the **kube-apiserver** doesn't perform validation on the manifest, it doesn't detect if those nested references will cause a problem. If the nesting references grow too large, excessive CPU and RAM usage can render the apiserver unresponsive to connections … hence the Denial of Service.

Kubernetes – **Security Basic Topics**

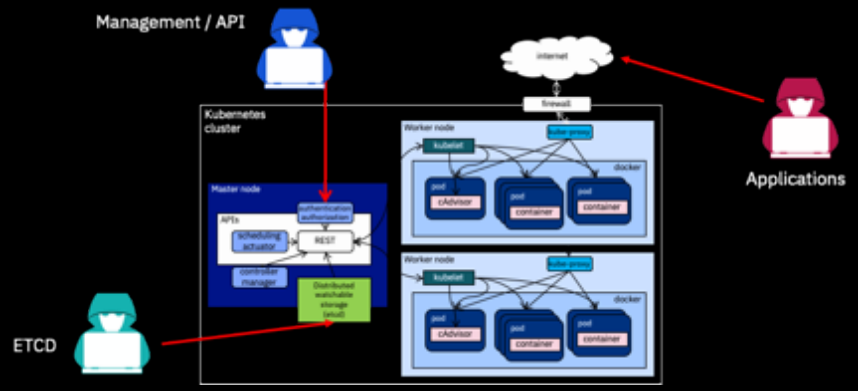
Reduce Kubernetes Attack Surfaces

etcd

Management/API

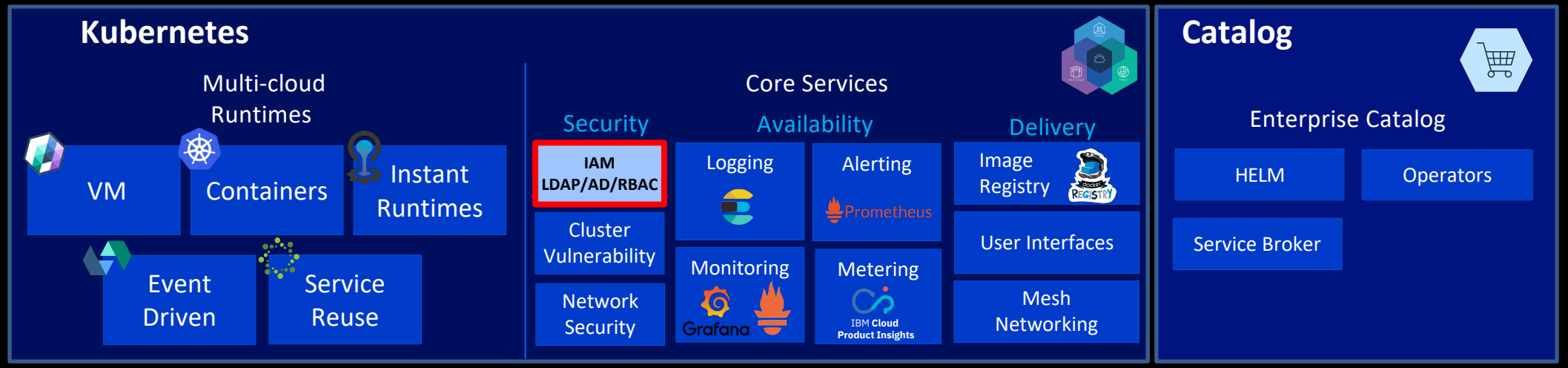
Applications

- Secure access to etcd
- Controlling access to the Kubernetes API
- Controlling access to the Kubelet
- Enforce resource usage limits for workloads
- Rotate infrastructure credentials frequently
- Enable audit logging
- Use Linux security features
- Controlling what privileges containers run with
- Enforcing Network Policies
- Image Scanning of your containers



Source: https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster
https://kubernetes.io/blog/2018/07/18/11-ways-not-to-get-hacked/

Kubernetes – Core Services - Controlling K8s Access



Authentication – WHO am I - (token, certs, OpenID Connect Provider (OIDC)...)

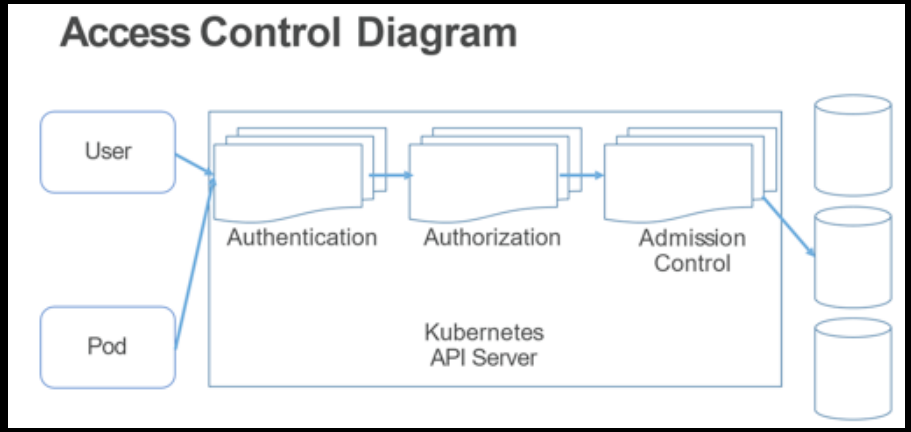
Developer — create, view, get permission.
Tester — create, delete, update, get permission.
Administrator — All permission

Authorization – CAN I - (RBAC)

Is the user or application authorized or have permissions to access a K8s object?

Admission Control

Intercepts requests to the Kubernetes API server prior to persistence of the object, but after the request is authenticated and authorized

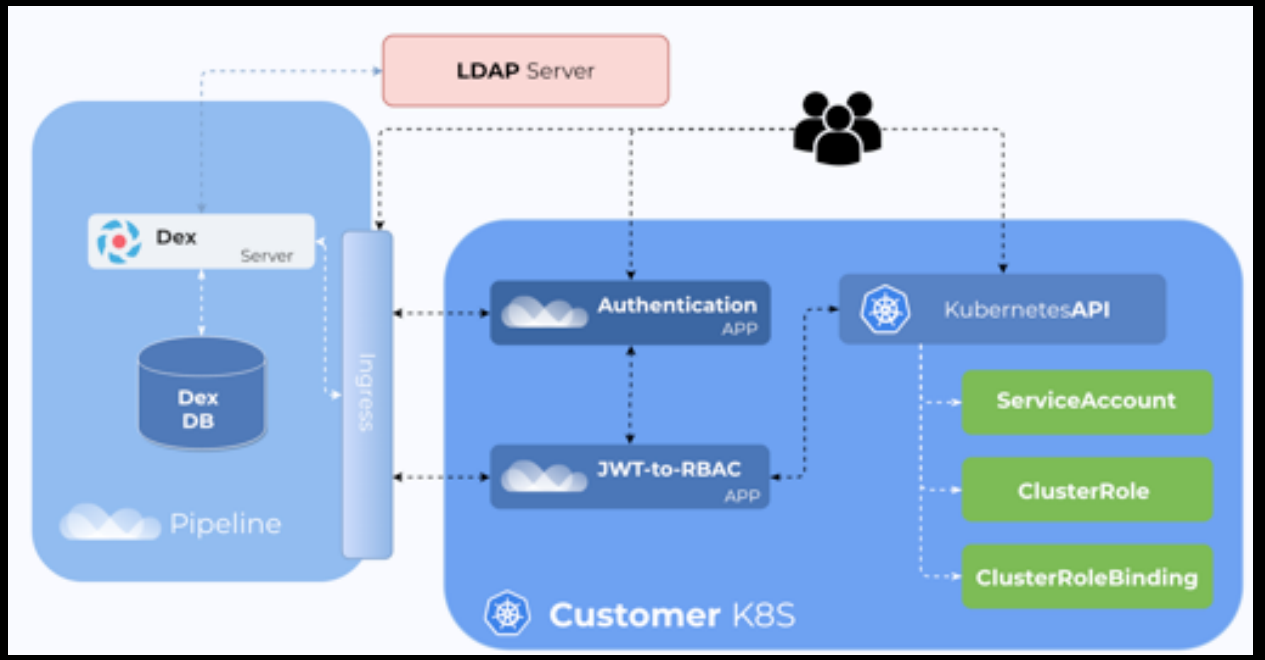


Source: https://kubernetes.io/docs/reference/access-authn-authz/controlling-access/

Integrating with LDAP/AD

OpenID Connect
ID Tokens are JSON Web Tokens (JWTs)

dex
Dex acts as a portal to other identity providers through connectors
dex defer authentication to LDAP servers, SAML providers, or established identity providers like GitHub, Google, and Active Directory.

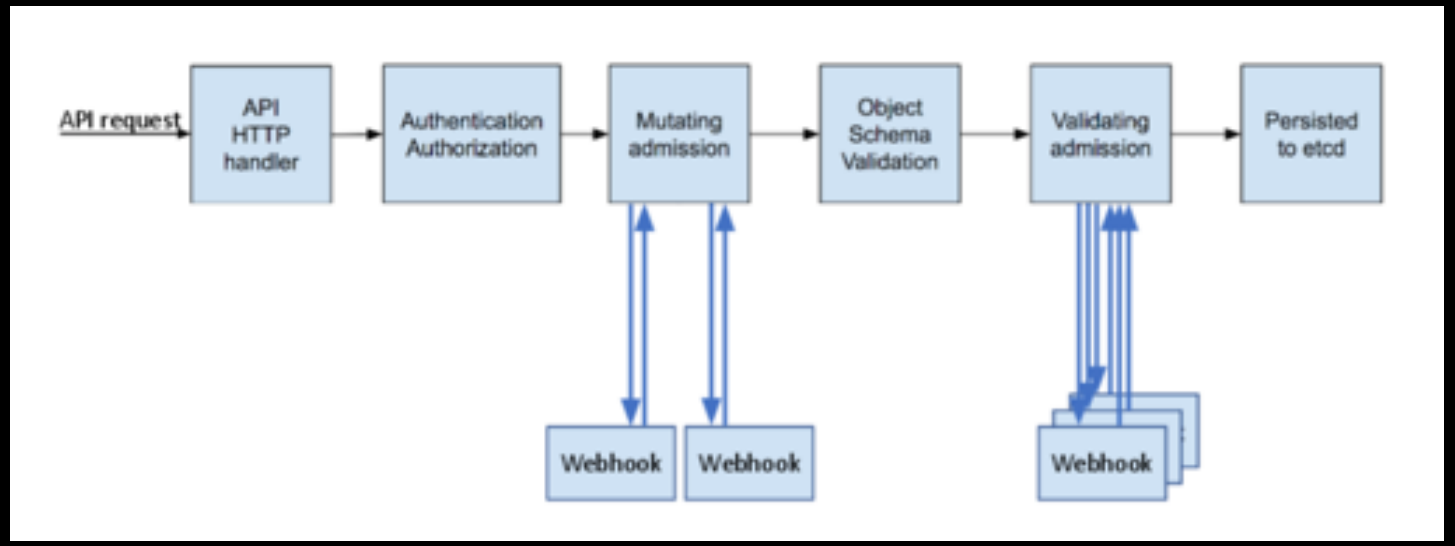


Source: https://github.com/krishnapmv/k8s-ldap and https://github.com/dexidp/dex

Kubernetes WebHooks

WebHooks Type

- **mutating**: to dynamically change incoming deployments on the fly (think automatic Istio sidecar injection), and
- **validating**: to accept or reject those same deployments based on the rules in your callback.
- → Image Scanner



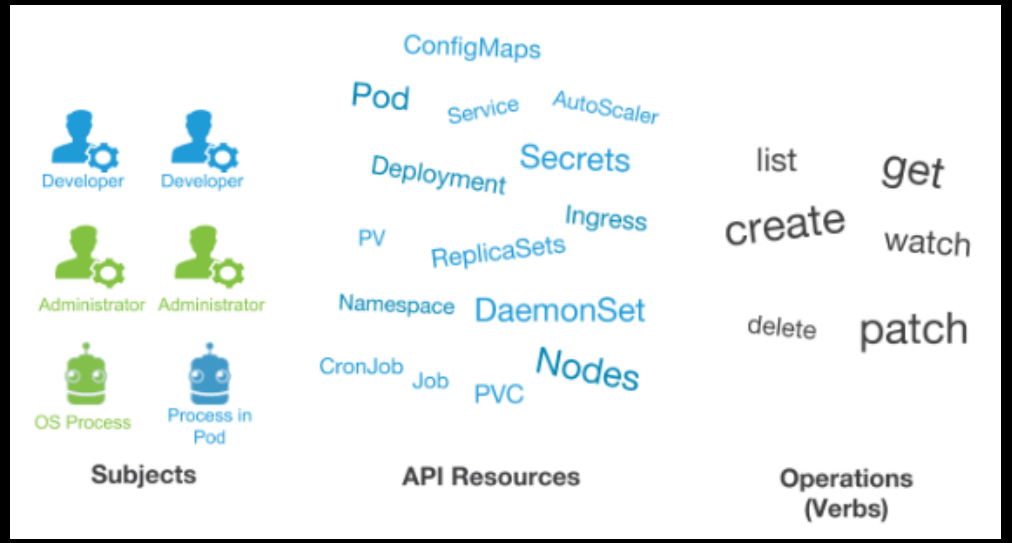
Source: https://github.com/krishnapmv/k8s-ldap and https://github.com/dexidp/dex

RBAC Basic Elements

RBAC Building Blocks

Objects
- Pods
- PersistentVolumes
- ConfigMaps
- Deployments
- Nodes
- Secrets
- Namespaces

Verbs
- create
- get
- delete
- list
- update
- edit
- watch
- exec



Source: https://kubernetes.io/docs/reference/access-authn-authz/rbac/

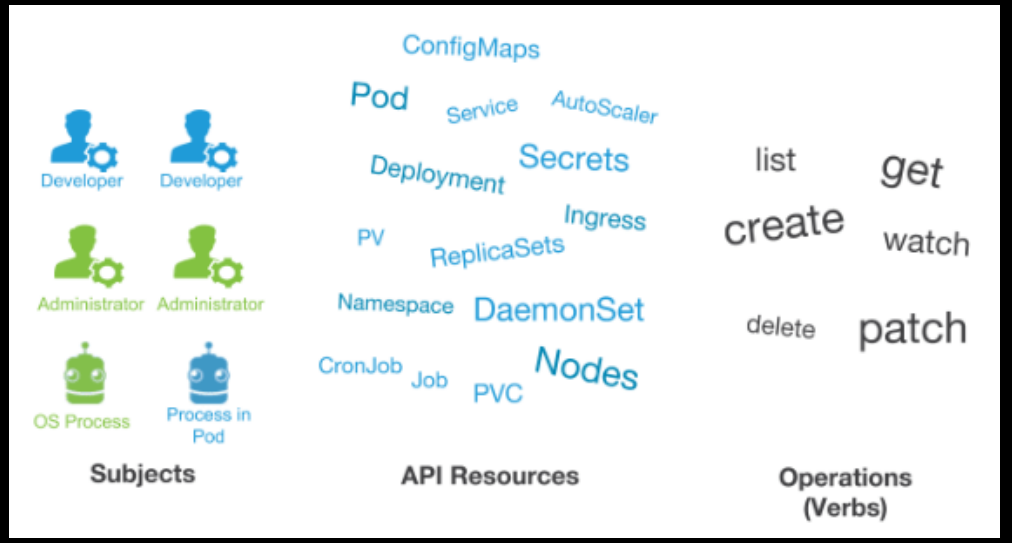
RBAC Basic Elements

Rules
Operations (verbs) that can be carried out on a group of resources which belong to different API Groups.

Roles and **ClusterRoles**
Both consist of rules.
- Role: applicable to a single namespace
- ClusterRole: is cluster-wide

Subjects
Entity that attempts an operation in the cluster
- User Accounts: Humans or processes living outside the cluster.
- Service Accounts: Namespaced account.
- Groups: Reference multiple accounts. (default groups like cluster-admin)

RoleBindings and **ClusterRoleBindings**
Bind subjects to roles



Source: https://kubernetes.io/docs/reference/access-authn-authz/rbac/

RBAC Example


```
kind: Role
metadata:
  name: pod-reader
rules:
- apiGroups: [“”]
  resources: [“pods”]
  verbs: [”get”,”list”]
```

```
kind: RoleBinding
metadata:
  name: pod-reader
subjects:
- kind: User
  name: John
roleRef:
  kind: Role
  name: pod-reader
```

Source: https://kubernetes.io/docs/reference/access-authn-authz/rbac/

Common RBAC Pitfalls

- **Cluster Administrator Role Granted Unnecessarily**
  The built-in **cluster-admin** role grants effectively unlimited access to the cluster.
  Should be granted only to the specific users that need it.

- **Duplicated Role Grant**
  Role definitions may overlap with each other, making access revocation more difficult.

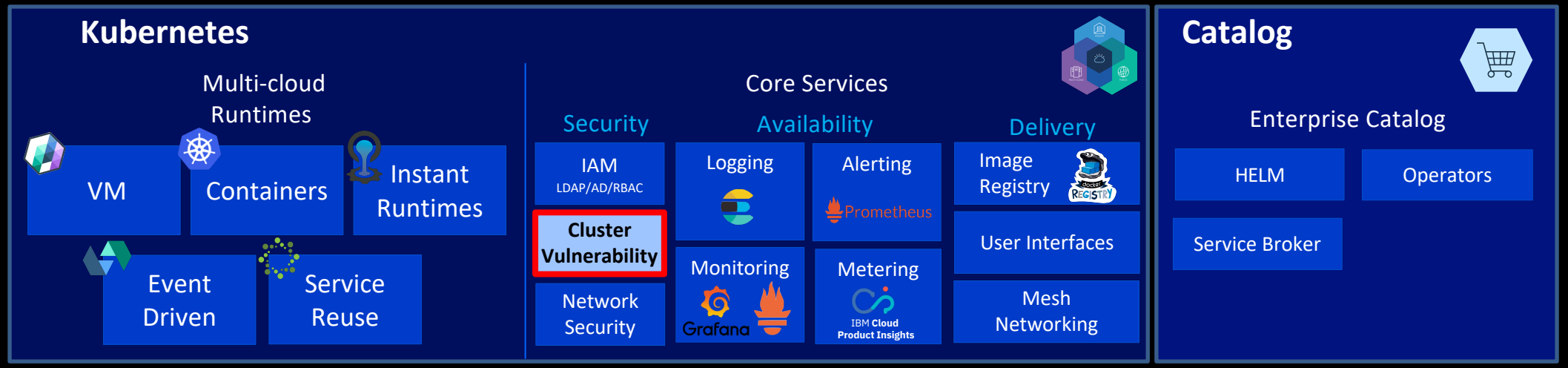
- **Unused Role**
  Roles that are created but not granted to any subject can increase the complexity of RBAC management.

- **Grant of Missing Roles**
  Role bindings can reference roles that do not exist.
  If the role name is reused those role bindings can unexpectedly become active.

Source: https://www.cncf.io/blog/2019/09/16/5-kubernetes-rbac-mistakes-you-must-avoid/

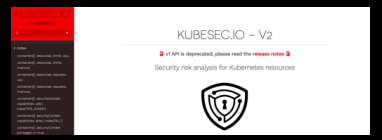
Always adapt least privileged access practices

Kubernetes – Core Services – Cluster Security



Kubernetes – Cluster Security – Secure Deployments

Scan for K8s Best Practices



kubesec – Validation of security best practices

Performs security risk analysis for Kubernetes resources and tells you what you should change in order to improve the security of those pods. It also gives you a score that you can use to create a minimum standard. The score incorporates a great number of Kubernetes best practices.


```
{
  "object": "Deployment/k8sdemo.default",
  "valid": true,
  "message": "Passed with a score of 4 points",
  "score": 4,
  "scoring": {
    "advise": [
```

```
{
  "selector": ".spec .serviceAccountName",
  "reason": "Service accounts restrict Kubernetes API access and should be configured with least privilege",
  "points": 3
},
```


Source: https://kubesec.io/

Kubernetes – Cluster Security – Secure Images

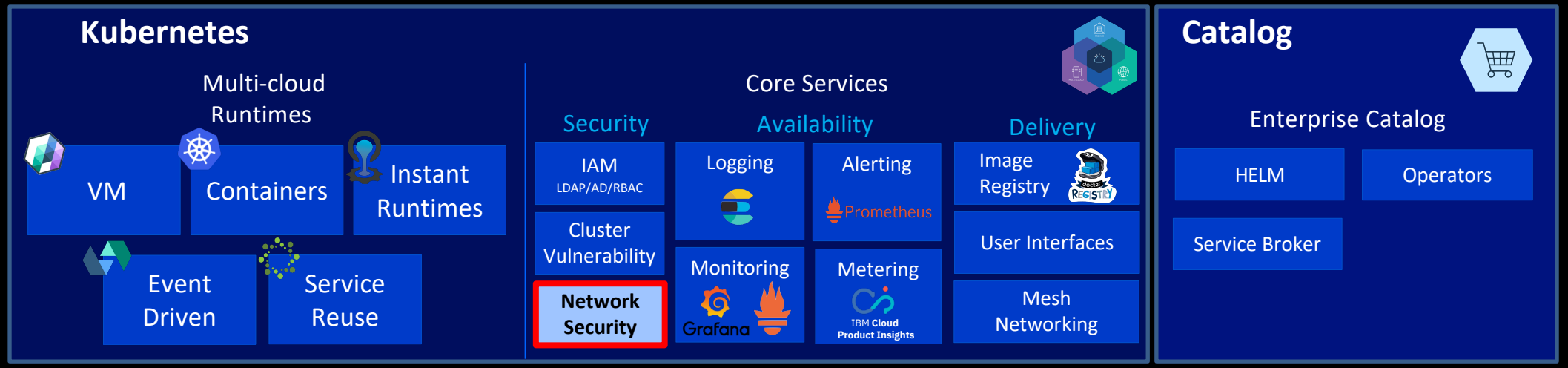
Scan images for vulerabilities

One of the best-known Image scanning tools is Clair.
It is an open source project for the static analysis of vulnerabilities in application containers (currently including appc and docker).

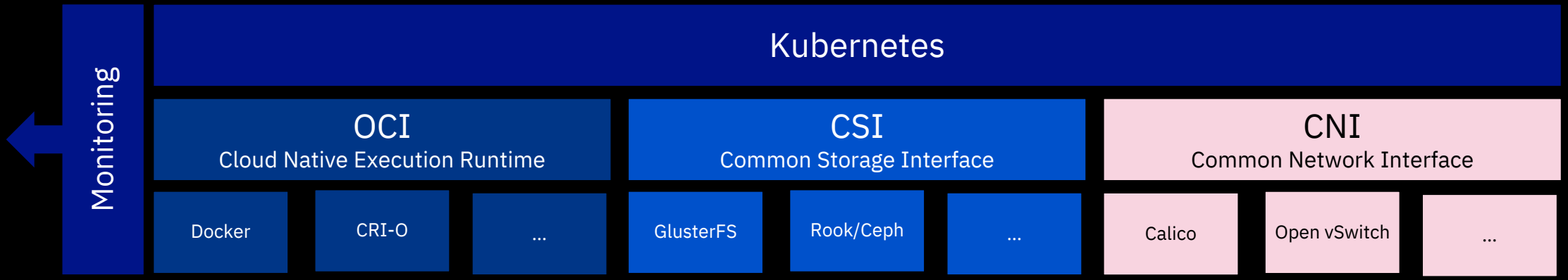
```
clair timeout 1m0s
docker timeout: 1m0s
no whitelist file
Analysing 14 layers
Got results from Clair API v1
Found 734 vulnerabilities
Unknown: 223
Negligible: 345
Low: 164
Medium: 2

+----------+----------------+-------------+-----------------+---------+-------------------------------+-------------------------------------------------+
| SEVERITY |      NAME      | FEATURENAME | FEATUREVERSION | FIXEDBY |          DESCRIPTION          |                      LINK                       |
+----------+----------------+-------------+-----------------+---------+-------------------------------+-------------------------------------------------+
| Medium   | CVE-2009-3546  | libwmf      | 0.2.8.4-10.6    |         | The _gdGetColors function     | https://security-tracker.debian.org/tracker/CVE-2009-3546 |
|          |                |             |                 |         | in gd_gd.c in PHP 5.2.11 and  |                                                 |
|          |                |             |                 |         | 5.3.x before 5.3.1, and the   |                                                 |
|          |                |             |                 |         | GD Graphics Library 2.x, does |                                                 |
|          |                |             |                 |         | not properly verify a certain |                                                 |
|          |                |             |                 |         | colorsTotal structure member, |                                                 |
|          |                |             |                 |         | which might allow remote      |                                                 |
|          |                |             |                 |         | attackers to conduct buffer   |                                                 |
|          |                |             |                 |         | overflow or buffer over-read  |                                                 |
|          |                |             |                 |         | attacks via a crafted GD file,|                                                 |
|          |                |             |                 |         | a different vulnerability than|                                                 |
|          |                |             |                 |         | CVE-2009-3293. NOTE: some of  |                                                 |
|          |                |             |                 |         | these details are obtained    |                                                 |
|          |                |             |                 |         | from third party information. |                                                 |
+----------+----------------+-------------+-----------------+---------+-------------------------------+-------------------------------------------------+
| Medium   | CVE-2007-3996  | libwmf      | 0.2.8.4-10.6    |         | Multiple integer overflows    | https://security-tracker.debian.org/tracker/CVE-2007-3996 |
|          |                |             |                 |         | in libgd in PHP before 5.2.4  |                                                 |
|          |                |             |                 |         | allow remote attackers to     |                                                 |
|          |                |             |                 |         | cause a denial of service     |                                                 |
|          |                |             |                 |         | (application crash) and       |                                                 |
|          |                |             |                 |         | possibly execute arbitrary    |                                                 |
|          |                |             |                 |         | code via a large (1) srcW     |                                                 |
|          |                |             |                 |         | or (2) srcH value to the (a)  |                                                 |
|          |                |             |                 |         | gdImageCopyResized function,  |                                                 |
|          |                |             |                 |         | or a large (3) sy (height)    |                                                 |
|          |                |             |                 |         | or (4) sx (width) value to    |                                                 |
|          |                |             |                 |         | the (b) gdImageCreate or the  |                                                 |
|          |                |             |                 |         | (c) gdImageCreateTrueColor    |                                                 |
|          |                |             |                 |         | function.                     |                                                 |
+----------+----------------+-------------+-----------------+---------+-------------------------------+-------------------------------------------------+
```

Source: https://github.com/quay/clair

Kubernetes – Core Services – Network Security



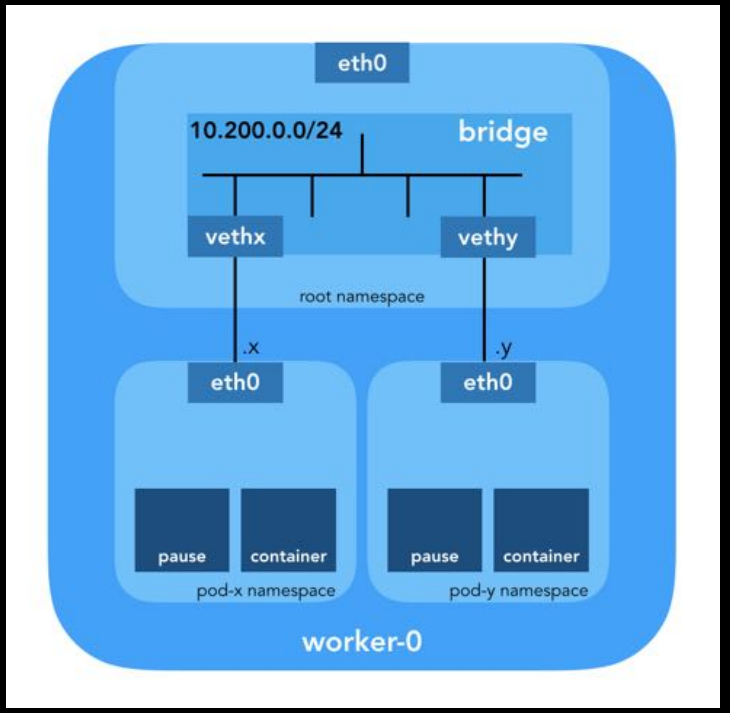
Kubernetes – Core Services – Network Security



CNI – Common Network Interface

Provides a rich set of security enforcement capabilities running on top of a highly scalable and efficient virtual network

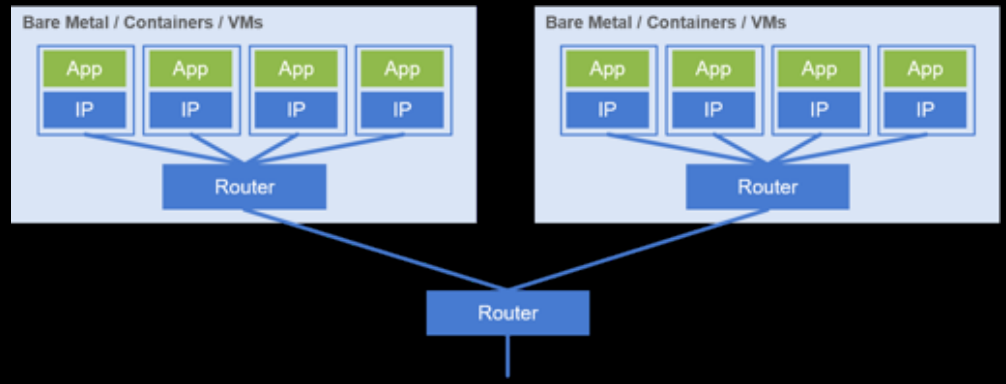
- **Calico**
  Virtual networking and network security for containers, VMs, and bare metal services.
- **Open vSwitch**
  Production quality, multilayer virtual switch
- ...



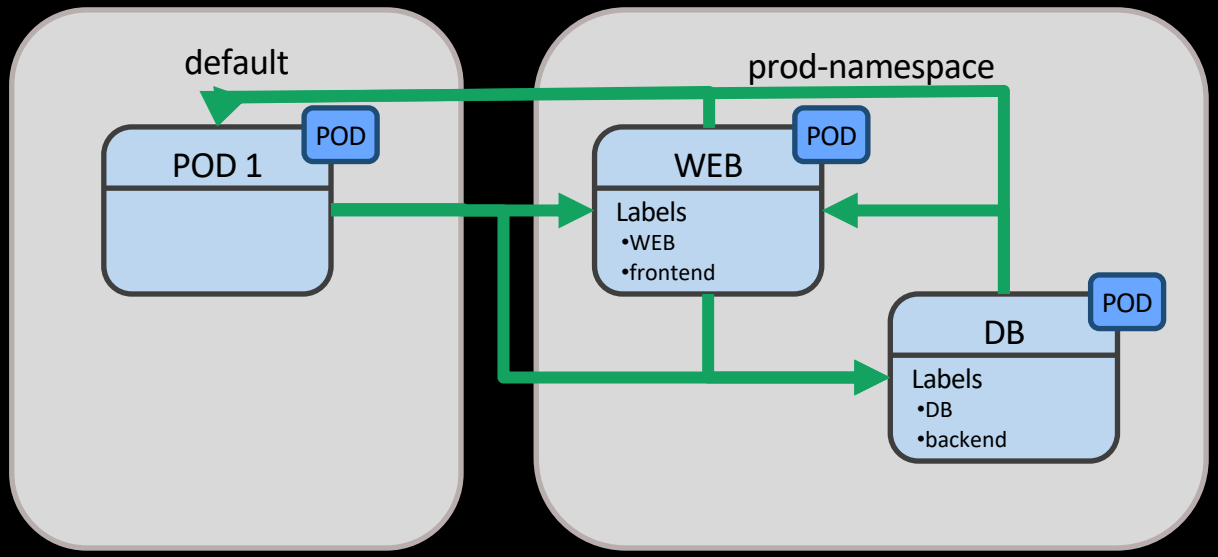
Kubernetes – Core Services – Network Security

Operates at **Layer 3**, which is the network layer

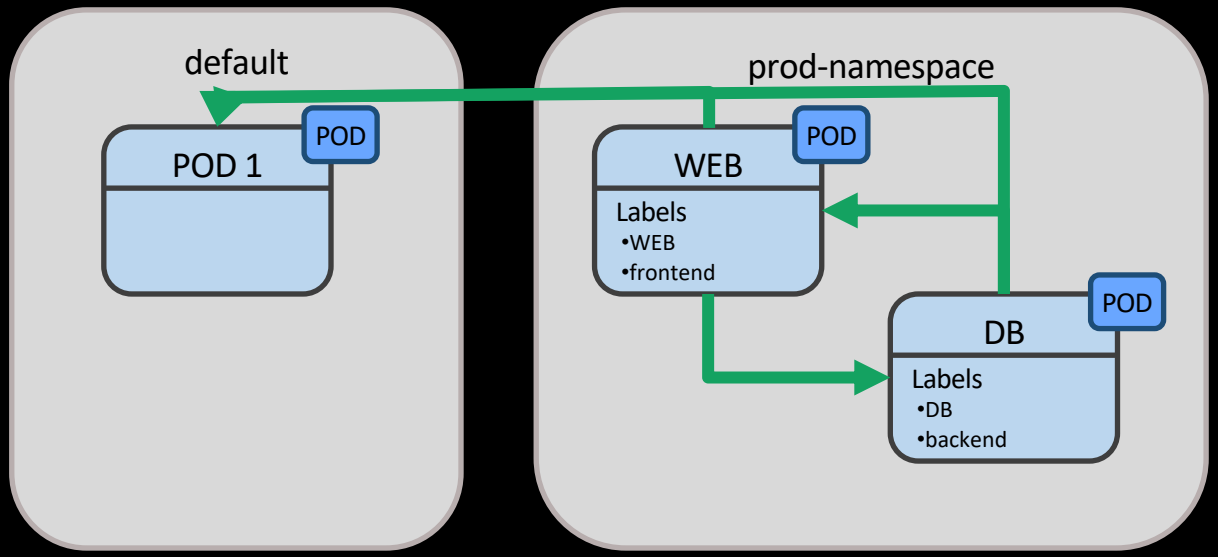
- Has the advantage of being **universal** (DNS, SQL, real-time streaming, ...)
- Can extend beyond the service mesh (including to **bare metal or VM** endpoints not under the control of Kubernetes).
- Calico's policy is enforced at the host node, outside the network namespace of the guest pods.
- Based on **iptables**, which are packet filters implemented in the standard Linux kernel, it is extremely fast.



Kubernetes – Core Services – Network Security

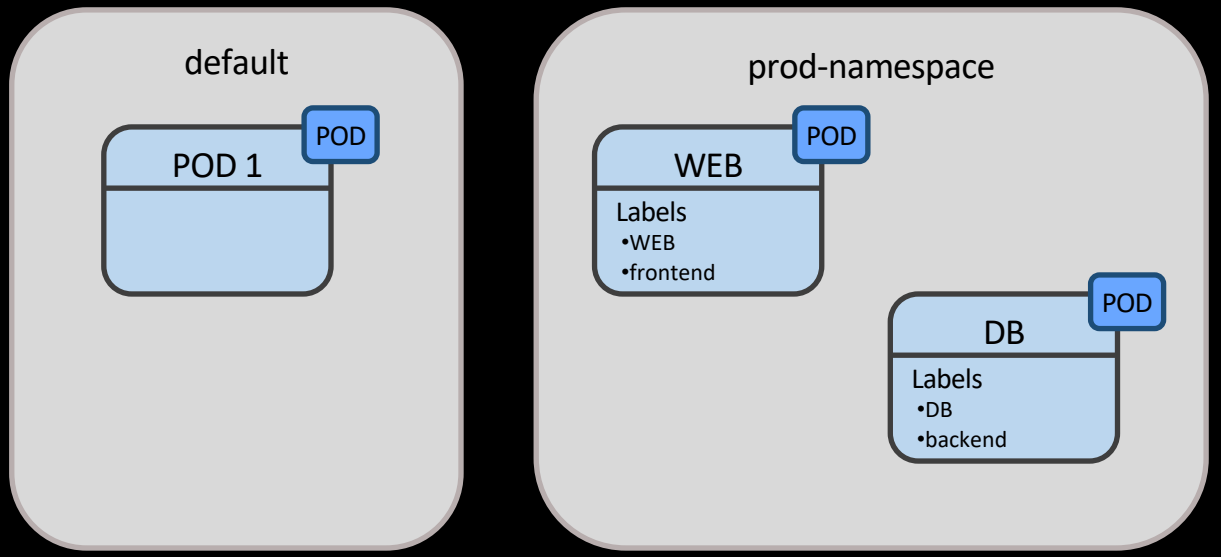


Kubernetes – Core Services – Network Security



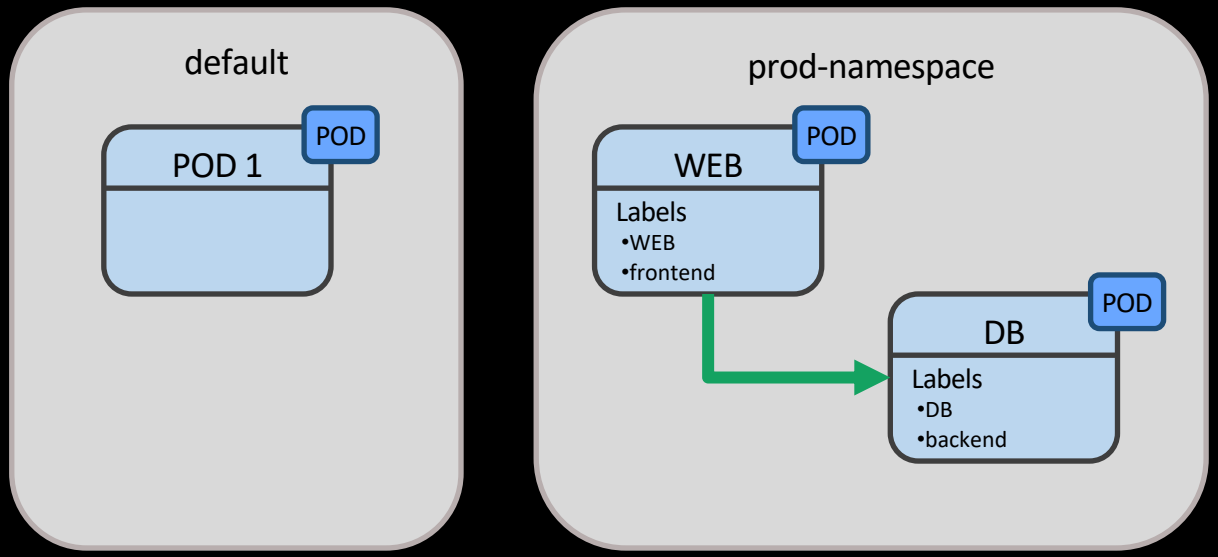

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
  namespace: demo-namespace
spec:
  podSelector:
    matchLabels: {}
```

Kubernetes – Core Services – Network Security



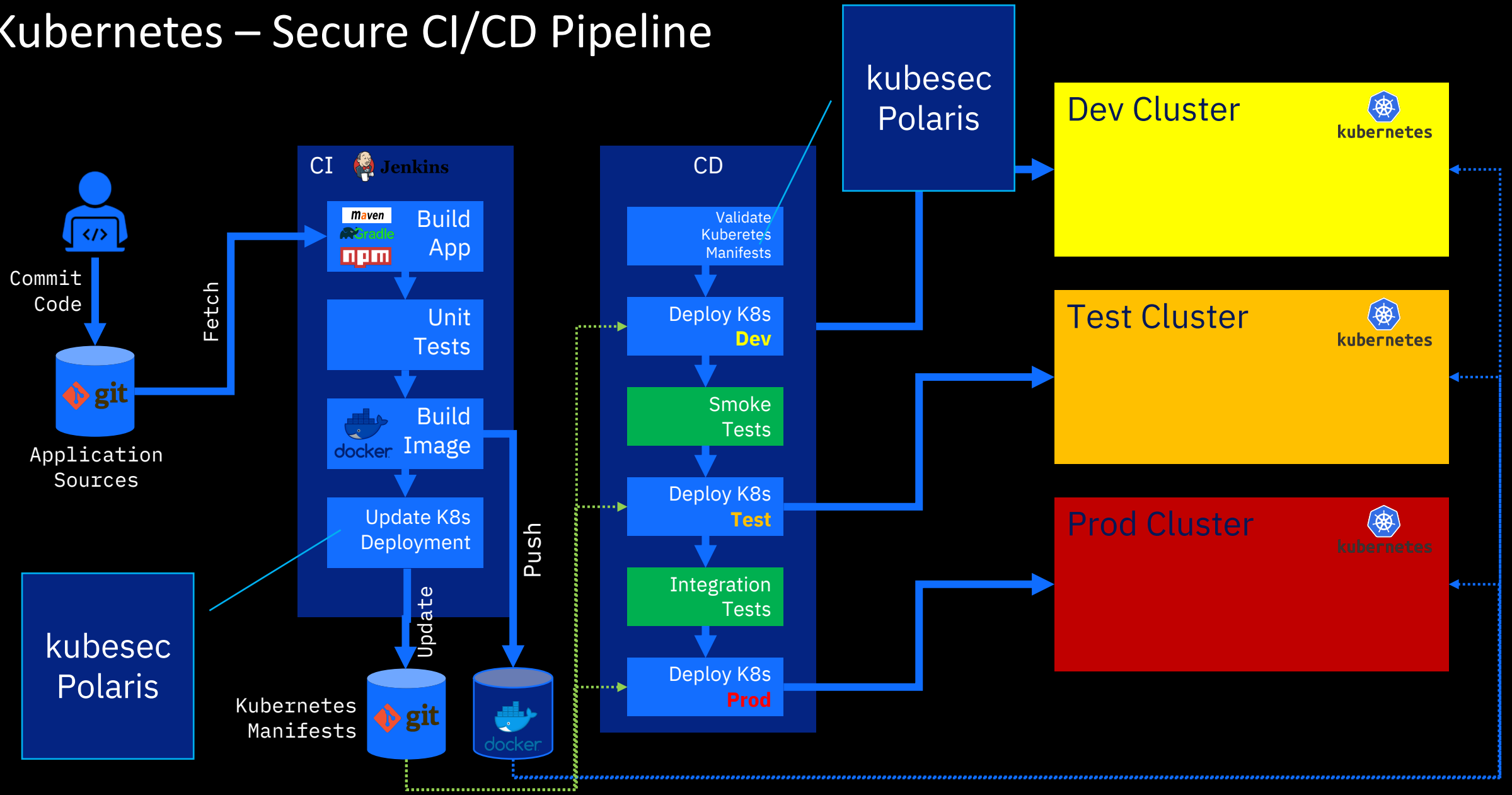

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny-all
spec:
  podSelector:
    matchLabels: {}
```

Kubernetes – Core Services – Network Security

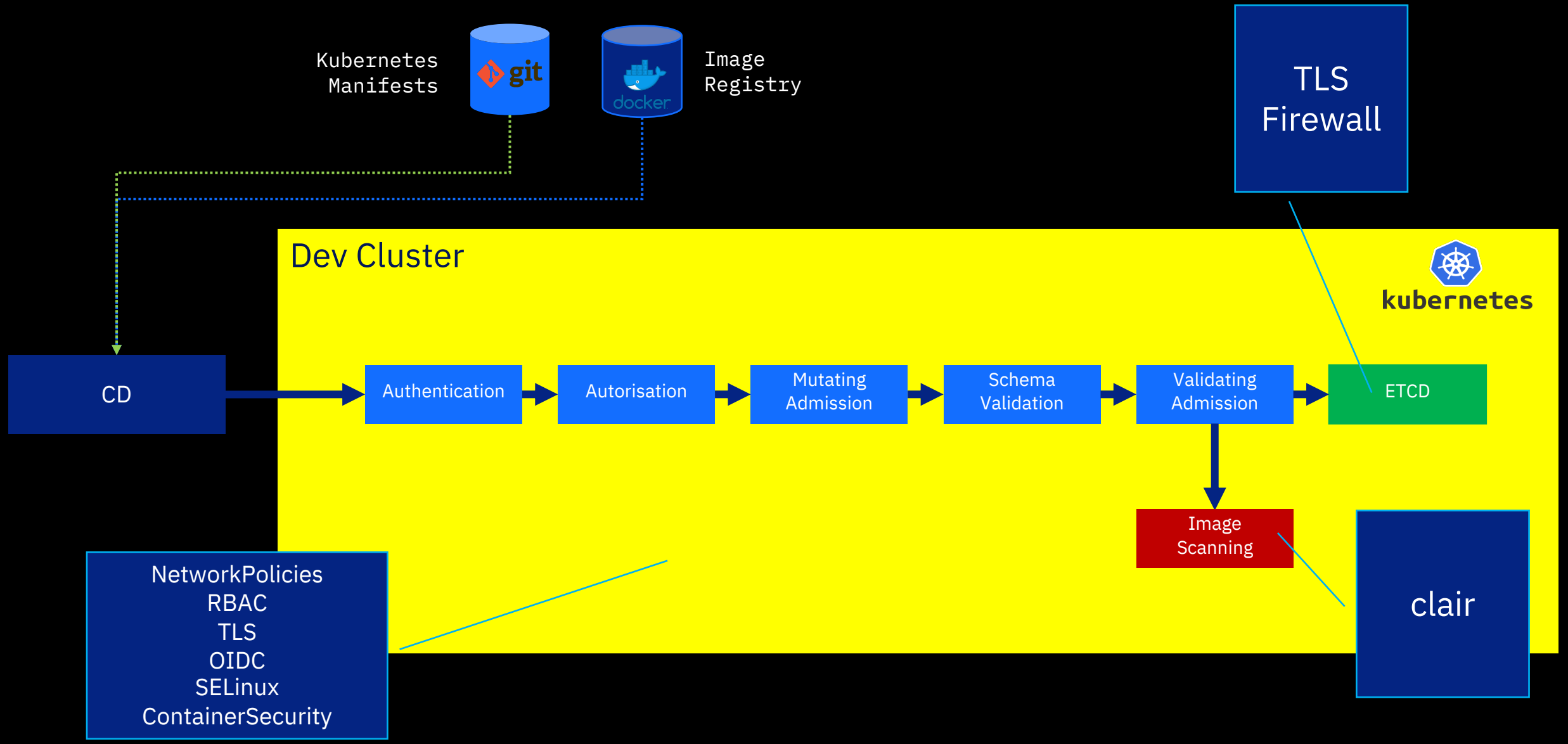



```
kind: NetworkPolicy
metadata:
  name: access-frontend-backend
  namespace: prod-namespace
spec:
  podSelector:
    matchLabels:
      run: DB
  ingress:
    - from:
      - podSelector:
          matchLabels:
            run: WEB
```

Kubernetes – Secure CI/CD Pipeline



Kubernetes – Secure CI/CD Pipeline



Kubernetes – Tip 1

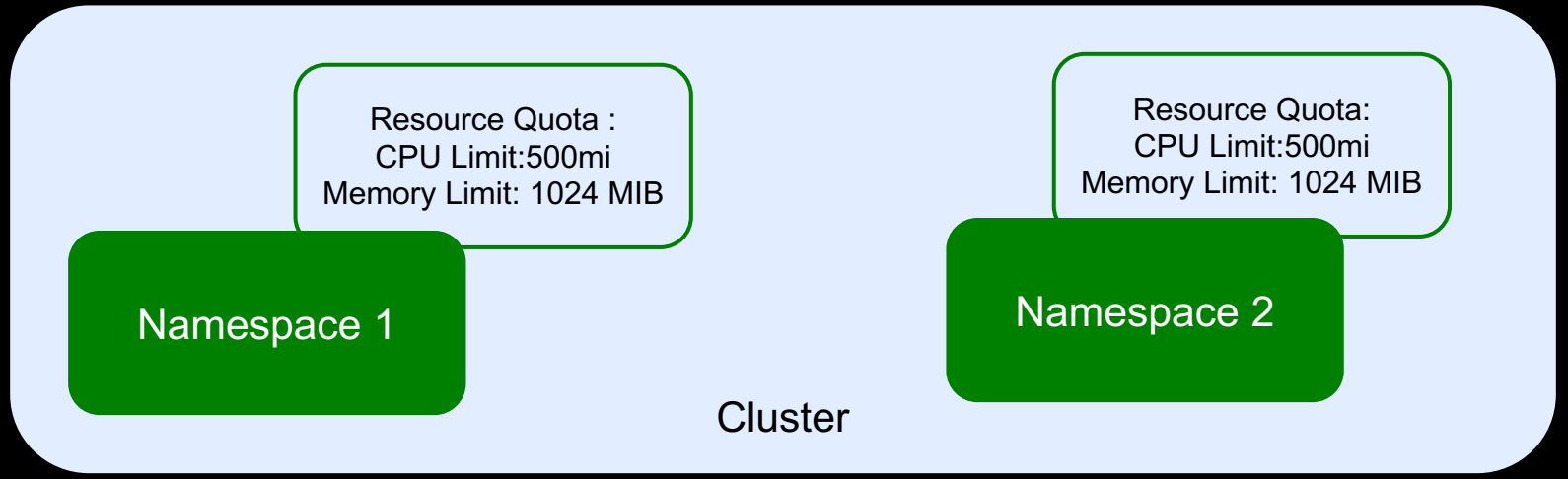
Use Namespaces Liberally

“Namespaces are cheap.”
Use them to separate things like infrastructure tooling and applications.
This allows you to restrict access easily using **RBAC** and to limit the scope of applications. It will also make your **network policy** creation easier when you decide to do it.

Kubernetes – Tip 2

Set resource quota

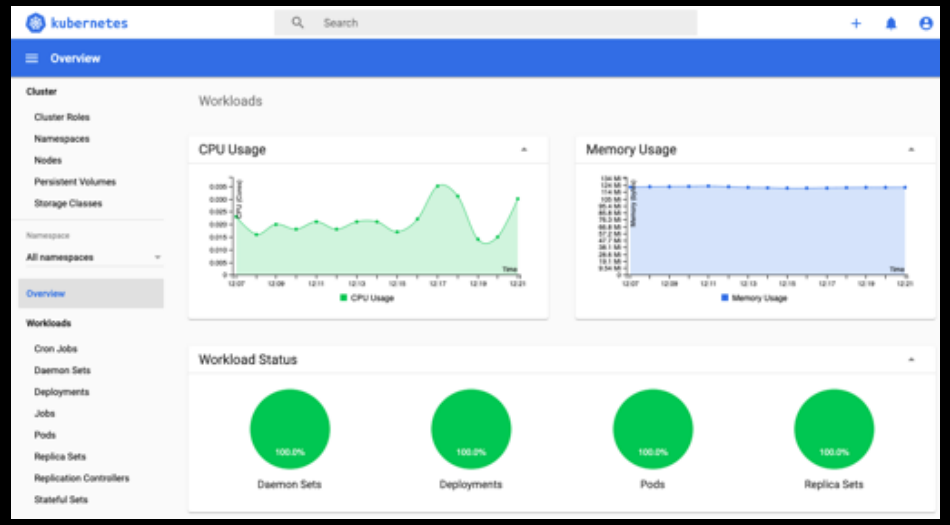
Once quota in a namespace for compute resources set, the users are forced to set requests or limits for those values. Avoids starving of Cluster.



Kubernetes – Tip 3

Secure Kubernetes Dashboard

The Kubernetes Dashboard has often been used for attacks.
Easy to deploy with high privileges.



Best practice is to:

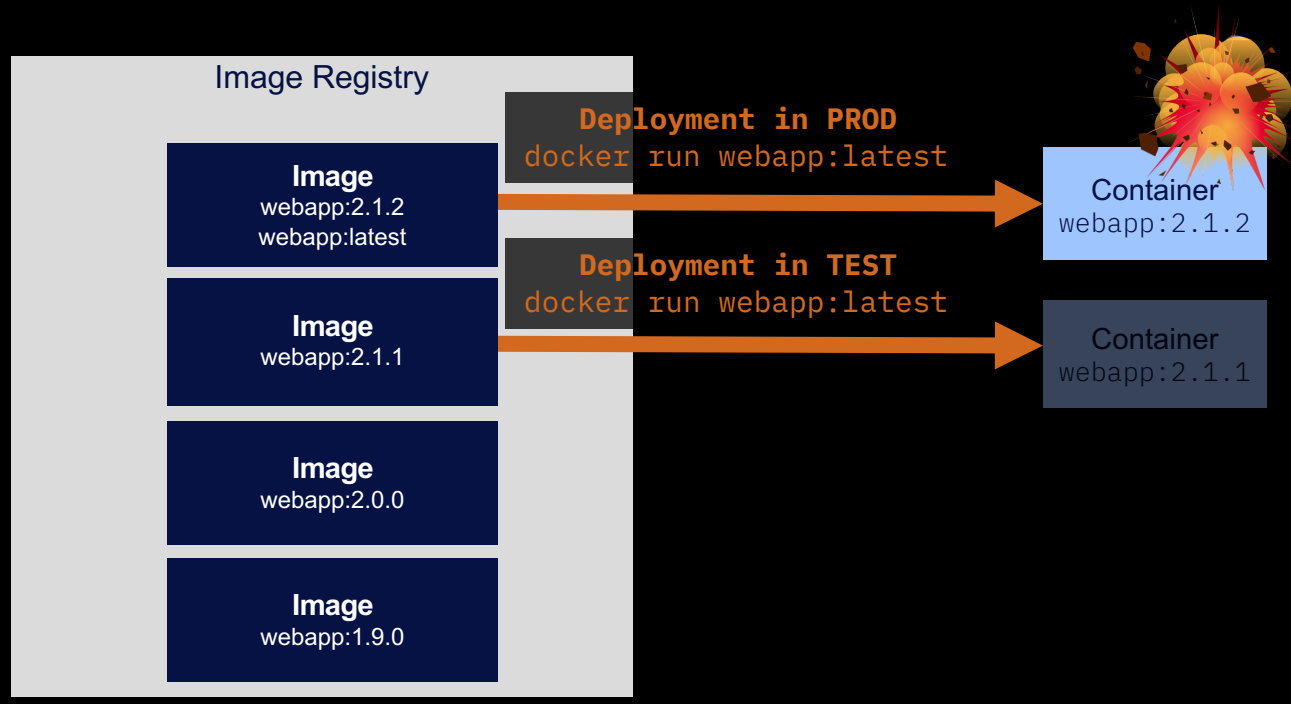
- Allow only authenticated access
- Use RBAC
- Ensure Dashboard service account has limited access
- Don’t expose to the internet (use kubectl proxy for local access)

Kubernetes – Tip 4

Avoid Rolling Tags at Any Cost

Avoid commands like *docker pull myregistry/myimage:latest.*
This “latest” is an example of a rolling tag (i.e. a tag that will point to different images over time).

If you want your deployments to be secure and maintainable, make sure that you use immutable images (for example: “*myregistry/myimage:1.1.2 "*).
With this approach, every time you deploy or scale, you know what image you are using and you will have the guarantee that the deployed image has been tested and validated.



Kubernetes – Tip 5

Don’t run containers as the root user

By default, containers run as the root user inside the container. This is easy to avoid using the pod specification to set a high-numbered UID.

```
spec:
  securityContext:
    runAsUser: 10324
```

Source https://medium.com/uptime-99/kubernetes-203-security-basics-6af5aee0367d

Kubernetes – Tip 6

Don’t run containers as privileged

Running a container or pod as privileged gives it the ability to make modifications to the host. This is a large security issue that is easy to mitigate by just not doing it.

Source https://medium.com/uptime-99/kubernetes-203-security-basics-6af5aee0367d

Kubernetes – Tip 7

Don’t use the default list of capabilities

Docker runs containers with a significant set of Linux capabilities by default, many of which your app might not require.

The following config will drop all Linux capabilities by default, allowing you to add only the specific capabilities your app actually needs:

```
spec:
  containers:
  - name: foo
    securityContext:
      capabilities:
        drop:
        - ALL
```

Source https://medium.com/uptime-99/kubernetes-203-security-basics-6af5aee0367d

IBM

QUESTIONS?

IBM **Cloud**

IBM

The Journey to Cloud
Mesh Networking

08

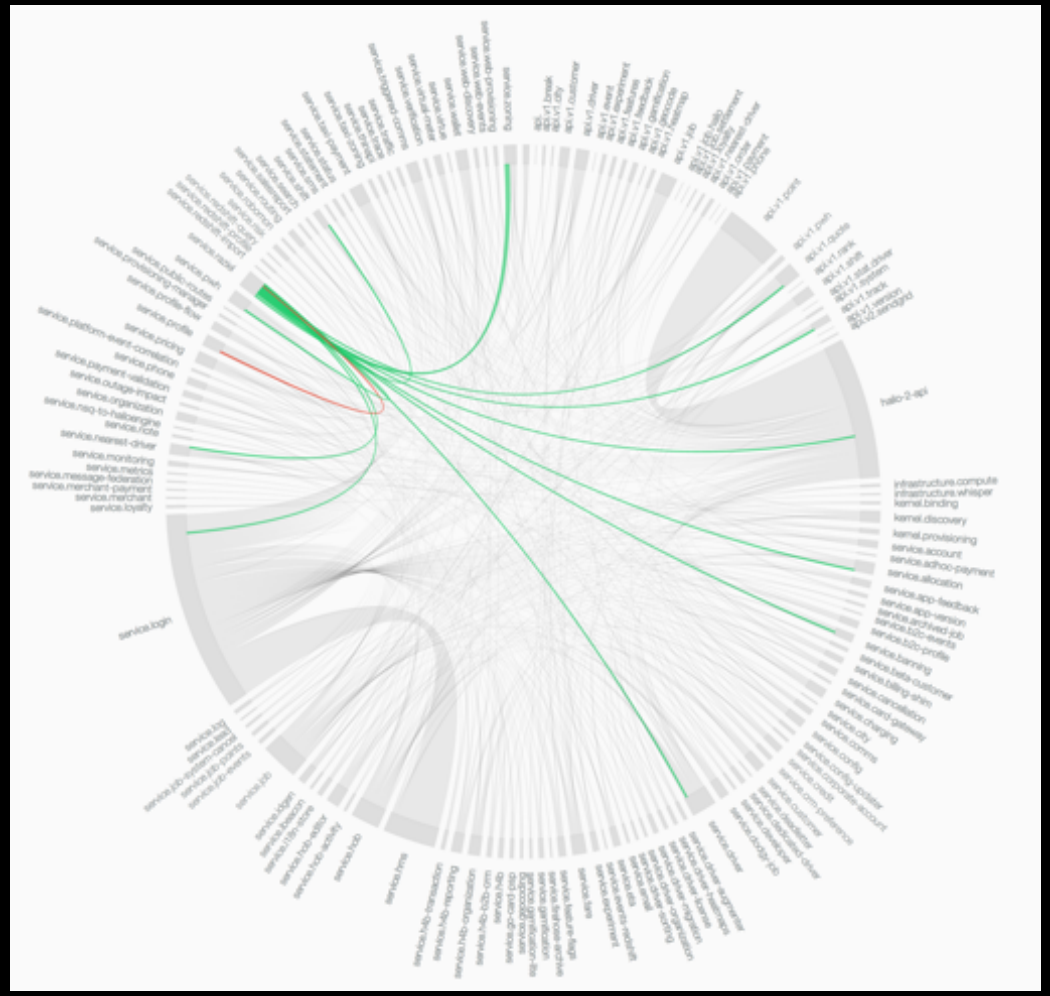
IBM **Cloud**

The trade off

Improved delivery velocity

in exchange for

increased operational complexity



Common DevOps Challenge 1

How do I **roll out** a newer version of my microservice **without down time**?

How do I **ensure traffic** continue to go to the current version before the newer version is tested and ready?

Common DevOps Challenge 2

How do I do **canary testing**?

How do I proceed to a **full rollout** after satisfactory testing of the new version?

Common DevOps Challenge 3

How do I do **A/B testing**?

- Release a new version to a subset of users in a precise way

I want to leverage crowdsourced testing. How do I **test** the new version **with a subset of users**?

I have **launched B in the dark**, but how can I keep B to myself or a small testing group?

Other common DevOps Challenges

4. Things don’t always go correctly in production… How do I **inject fault** to my microservices to prepare myself?

5. My services can only **handle certain rate**, how can I limit rate for some of my services?

Other common DevOps Challenges

6. I need to **view and monitor** what is going on with each of my services when crisis arises.

7. How can I **secure my services**.

Service Mesh

Dedicated infrastructure layer
to make
service-to-service communication
fast, safe and reliable

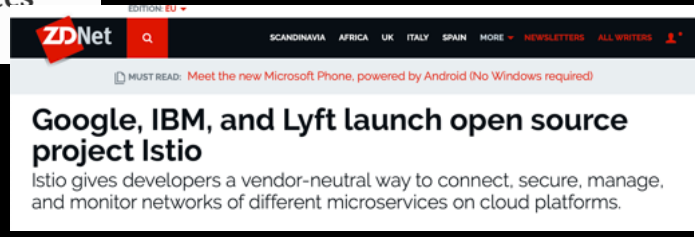
Istio

A service mesh designed to connect, manage and secure micro services


Forbes

3,859 views | May 25, 2017, 01:39am

Google, IBM And Lyft Want To Simplify Microservices Management With Istio

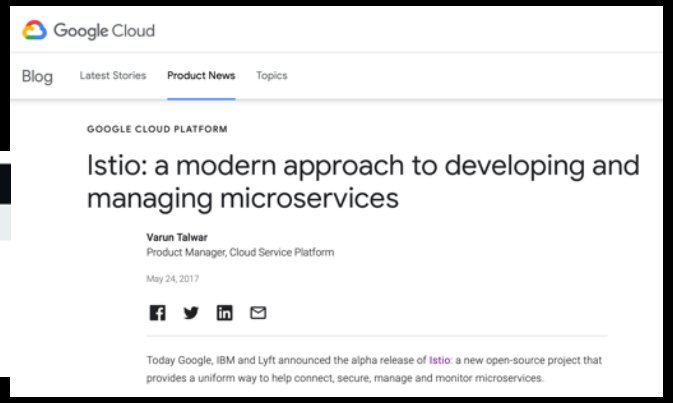

ZDNet

SCANDINAVIA AFRICA UK ITALY SPAIN MORE NEWSLETTERS ALL WRITERS

MUST READ: Meet the new Microsoft Phone, powered by Android (No Windows required)

Google, IBM, and Lyft launch open source project Istio

Istio gives developers a vendor-neutral way to connect, secure, manage, and monitor networks of different microservices on cloud platforms.


Google Cloud

Blog Latest Stories Product News Topics

GOOGLE CLOUD PLATFORM

Istio: a modern approach to developing and managing microservices

Varun Talwar
Product Manager, Cloud Service Platform

May 24, 2017

Today Google, IBM and Lyft announced the alpha release of Istio: a new open-source project that provides a uniform way to help connect, secure, manage and monitor microservices.

Launched in May 2017 by Google, Lyft and IBM

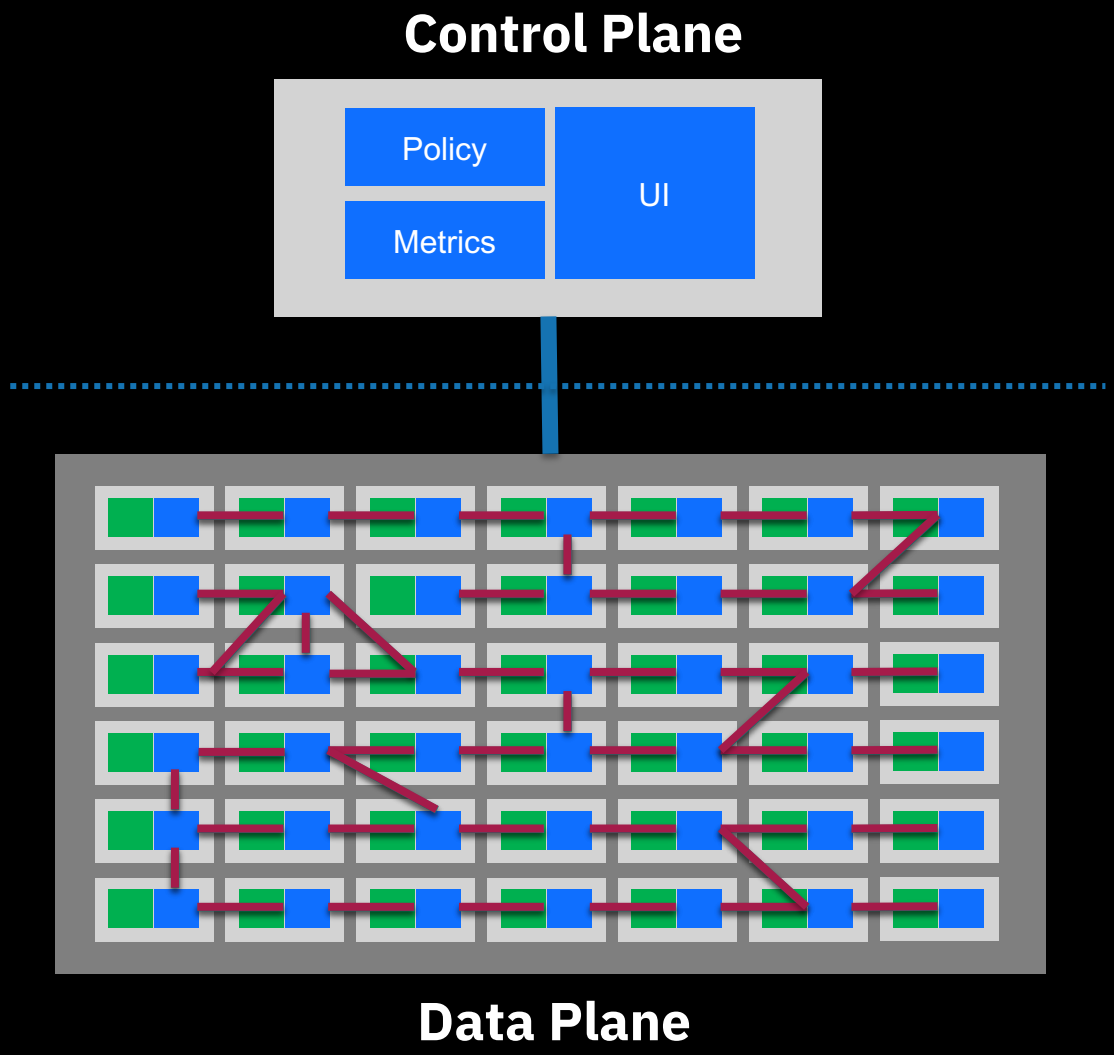
Open Source

Zero Code Changes

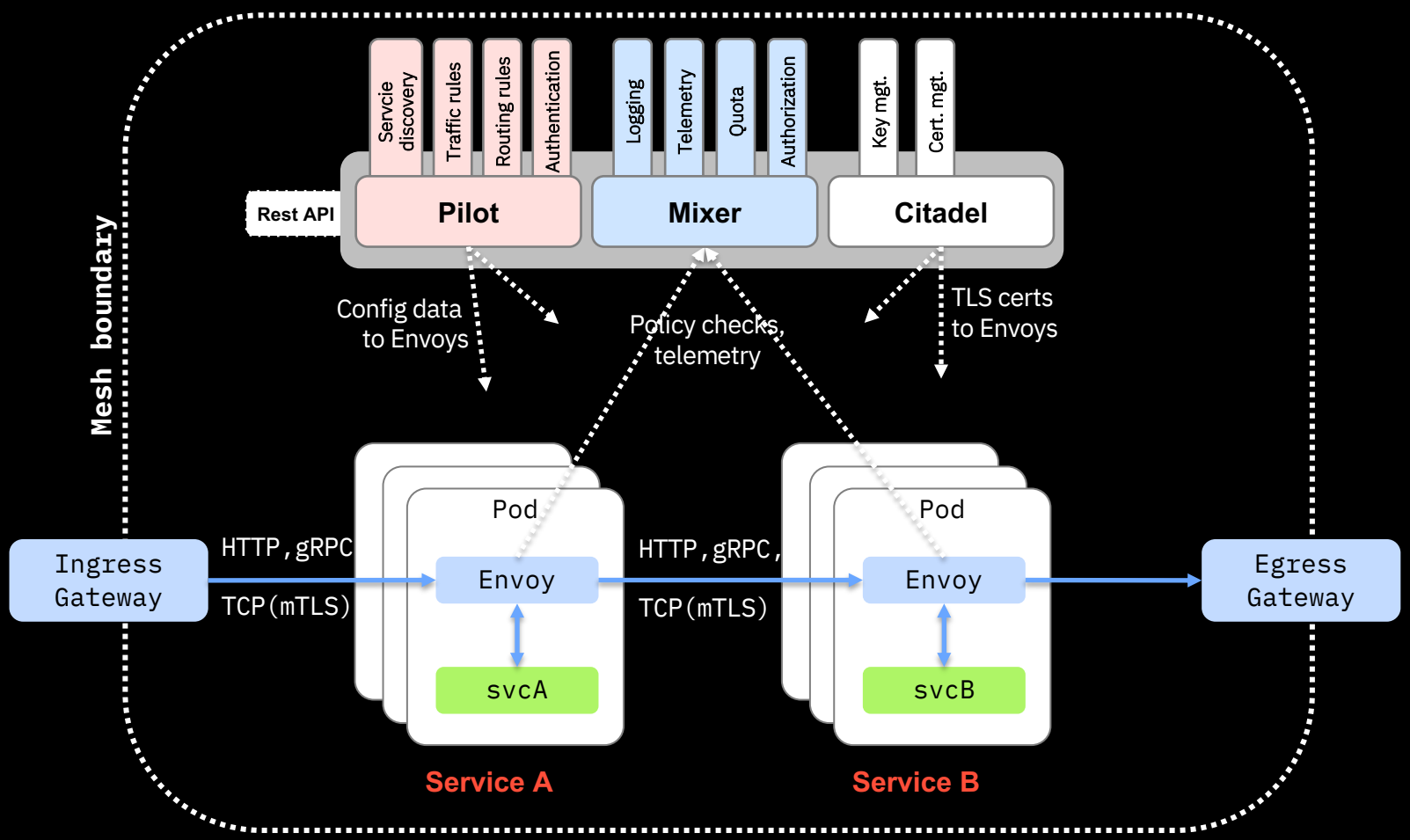
Service Mesh

Tent poles:

- Control over request routing (CI/CD release patterns)
- Resiliency (retries, timeouts, etc)
- Cascading failure prevention (circuit breaking)
- Load balancing
- Provide and manage TLS termination between endpoints
- Metrics to provide instrumentation at the service-to-service layer



ISTIO - Architecture



Istio control plane traffic. Request routing rules, resilience configuration (circuit breakers, timeouts, retries), policies (ACLs, rate limits, auth), and metrics/reports from proxies.

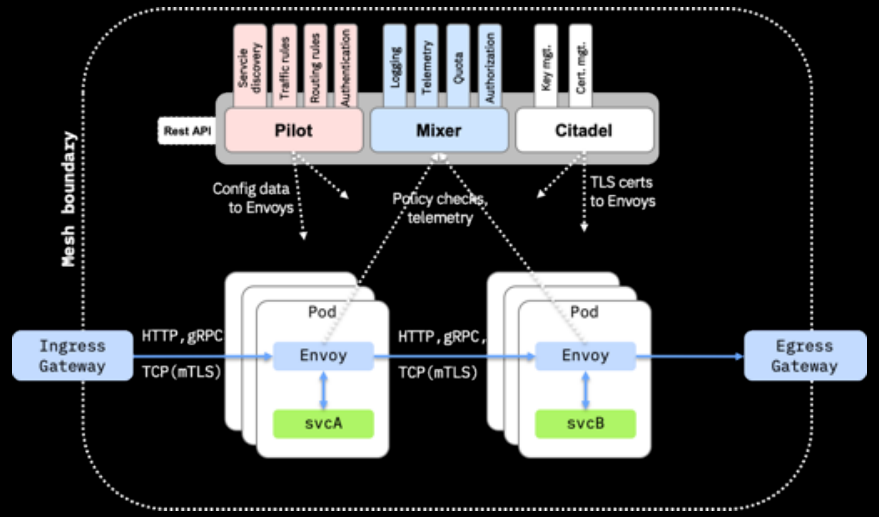
User/application traffic. HTTP/1.1, HTTP/2, gRPC, TCP with or without TLS

ISTIO - Architecture

Istio

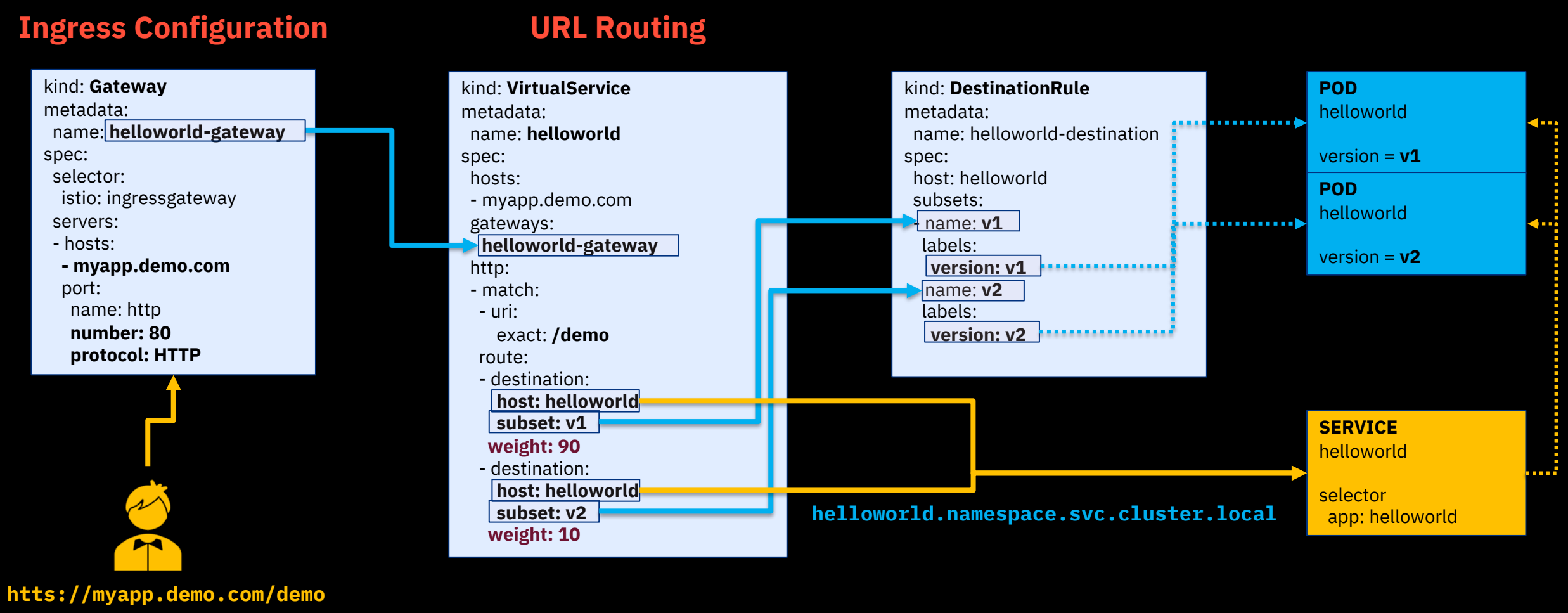
Operates at **Layer 7**

- policies can be applied based on virtual host, URL, or other HTTP headers.
- Flexibility in processing.
- Allows it to be distributed



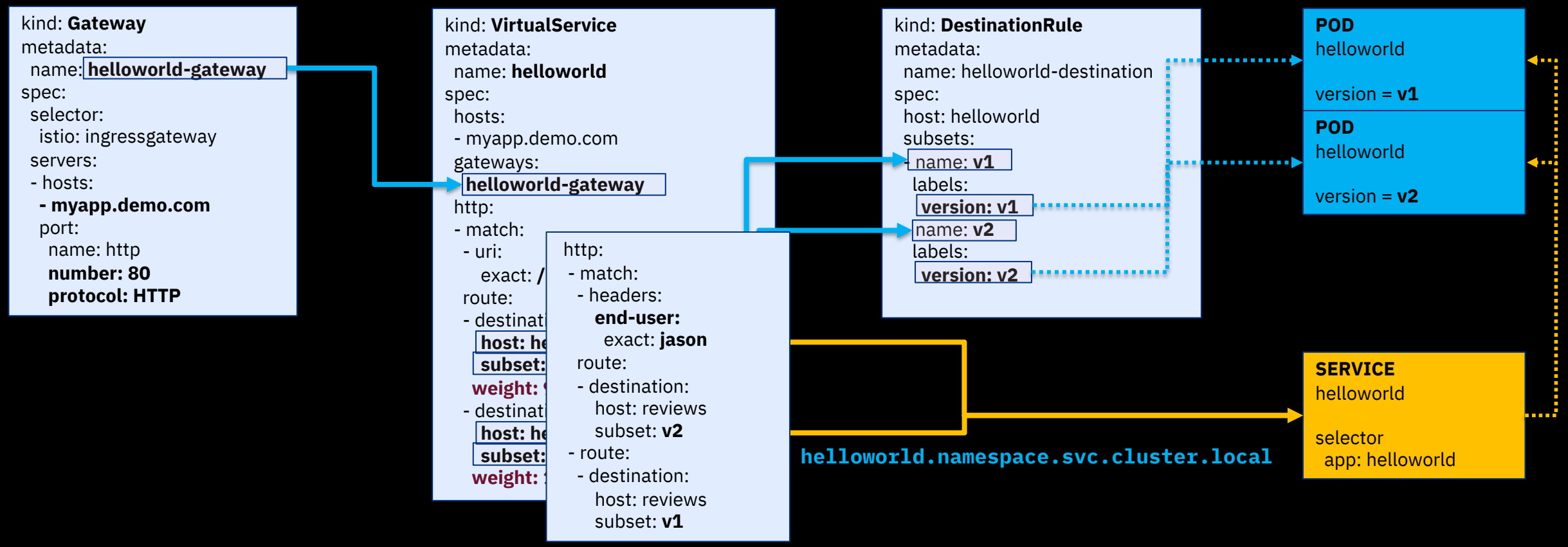
ISTIO

Custom resource definitions



ISTIO

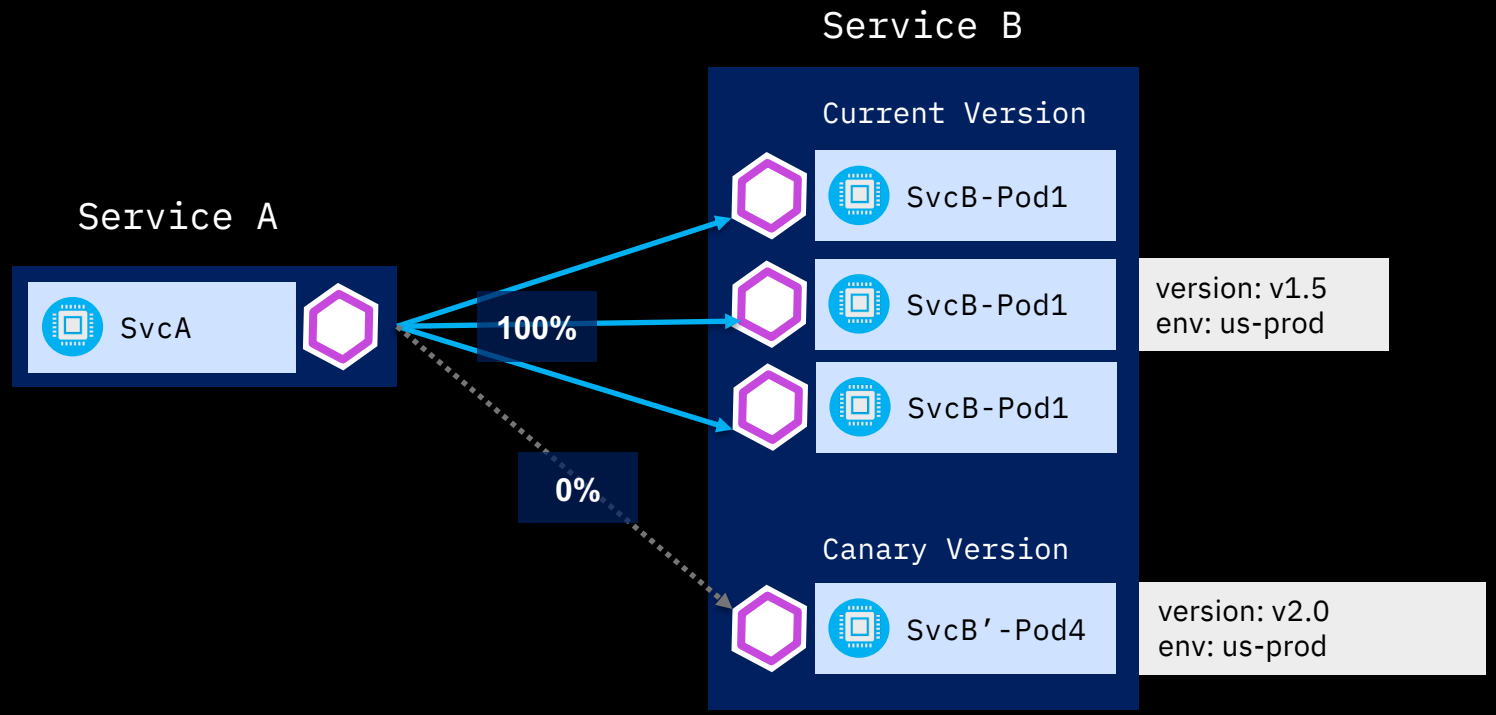
Custom resource definitions



Addressing DevOps Challenges

#	CHALLENGE	ISTIO SOLUTION
CHALLENGE 1	ROLL OUT NEW VERSION WITHOUT DOWNTIME OR CHANGING CODE	TRAFFIC CONTROL
CHALLENGE 2	HOW TO DO CANARY TESTING	TRAFFIC SPLITTING
CHALLENGE 3	HOW TO DO A/B TESTING	TRAFFIC STEERING
CHALLENGE 4	THINGS DON'T ALWAYS GO CORRECTLY IN PRODUCTION…	TRAFFIC MIRRORING RESILIENCY RESILIENCY TESTING
CHALLENGE5	HOW CAN I LIMIT RATE FOR SOME OF MY SERVICES?	RATE LIMITING
CHALLENGE 6	I NEED TO VIEW AND MONITOR WHAT IS GOING ON WHEN CRISIS ARISES	TELEMETRY
CHALLENGE 7	HOW CAN I SECURE MY SERVICES?	AUTHENTICATION AUTHORIZATION CALICO

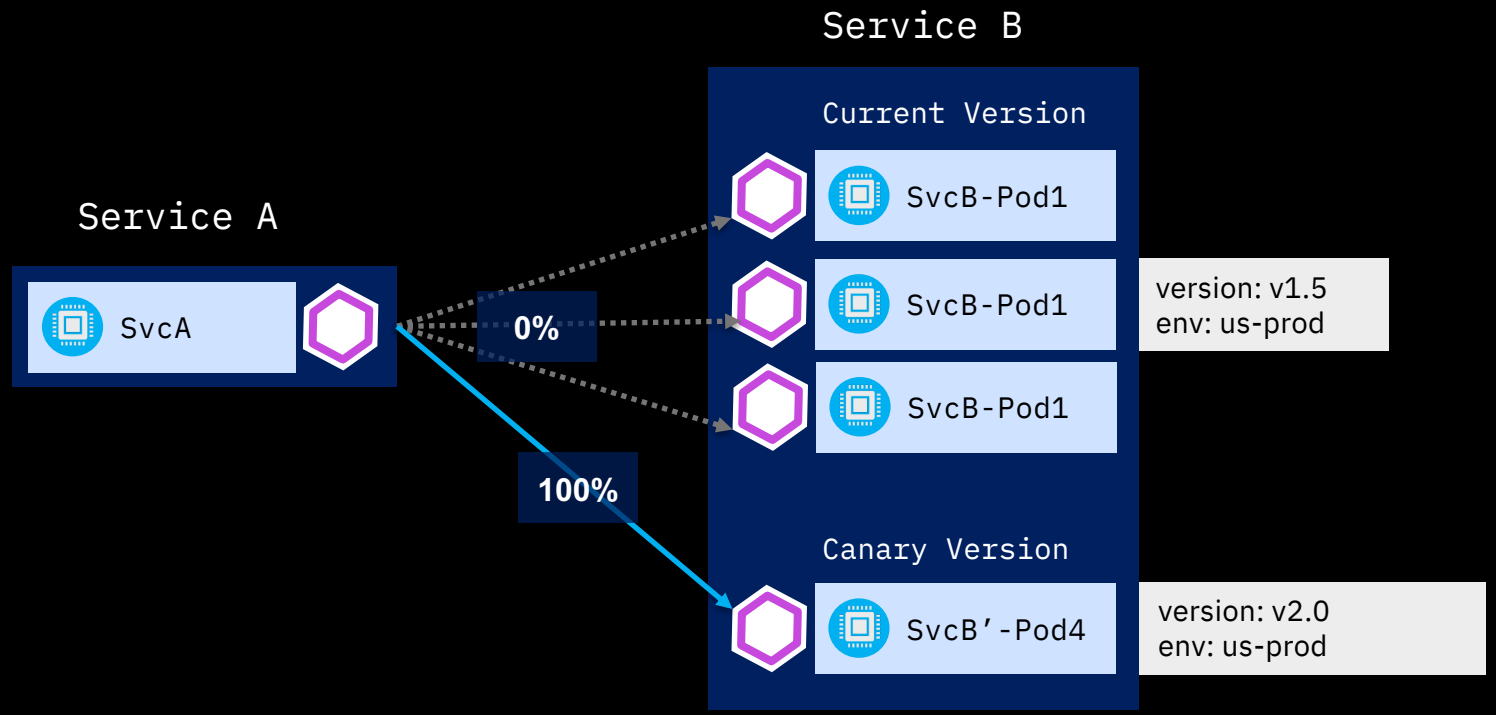
Traffic Control




```
kind: VirtualService
destination:
  serviceB.cluster.local
match:
  source:
    serviceA.cluster.local
route:
  - labels:
    version: v1.5
    env: us-prod
  weight: 100
```

CHALLENGE 1
ROLL OUT NEW VERSION WITHOUT DOWNTIME OR CHANGING CODE

Traffic Control

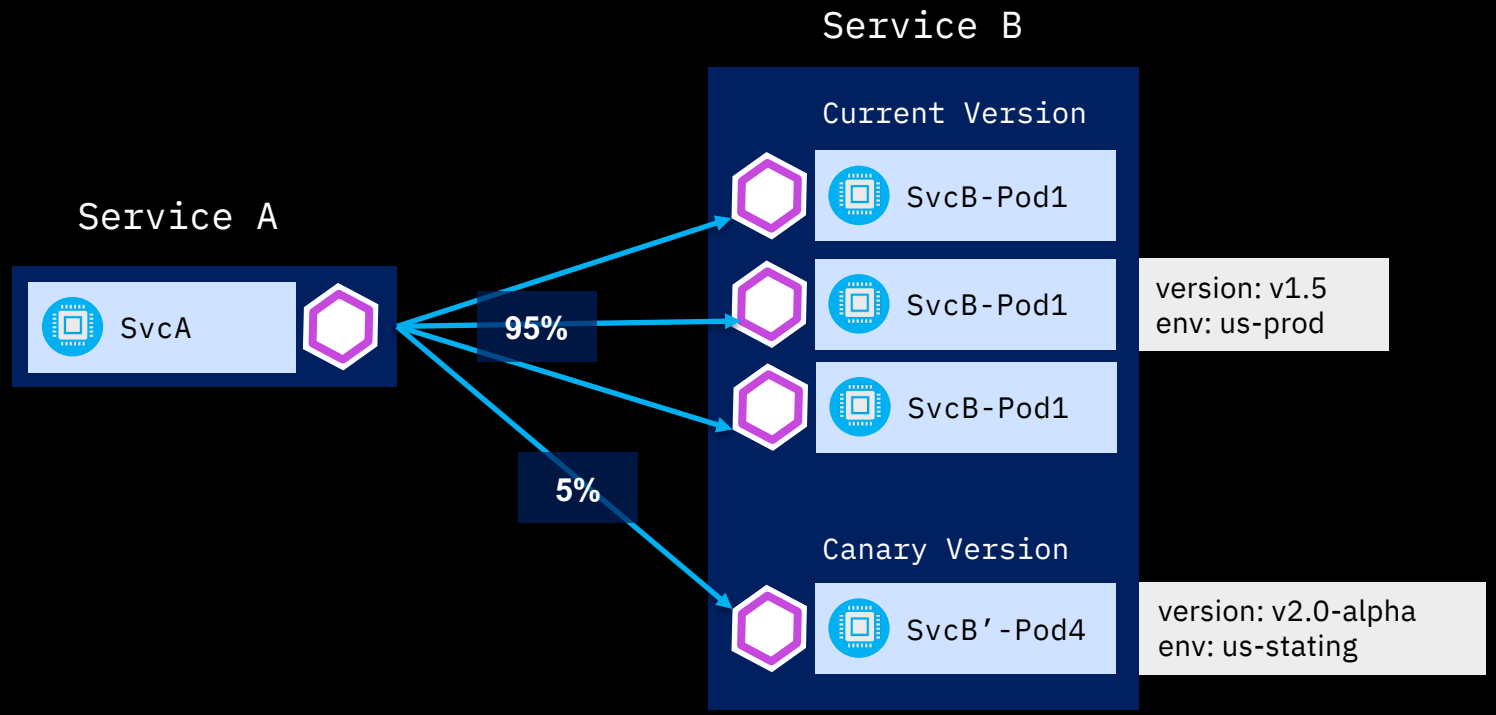



```
kind: VirtualService
destination:
  serviceB.cluster.local
match:
  source:
    serviceA.cluster.local
route:
  - labels:
    version: v2.0
    env: us-prod
  weight: 100
```

Update

CHALLENGE 1
ROLL OUT NEW VERSION WITHOUT DOWNTIME OR CHANGING CODE

Traffic Splitting

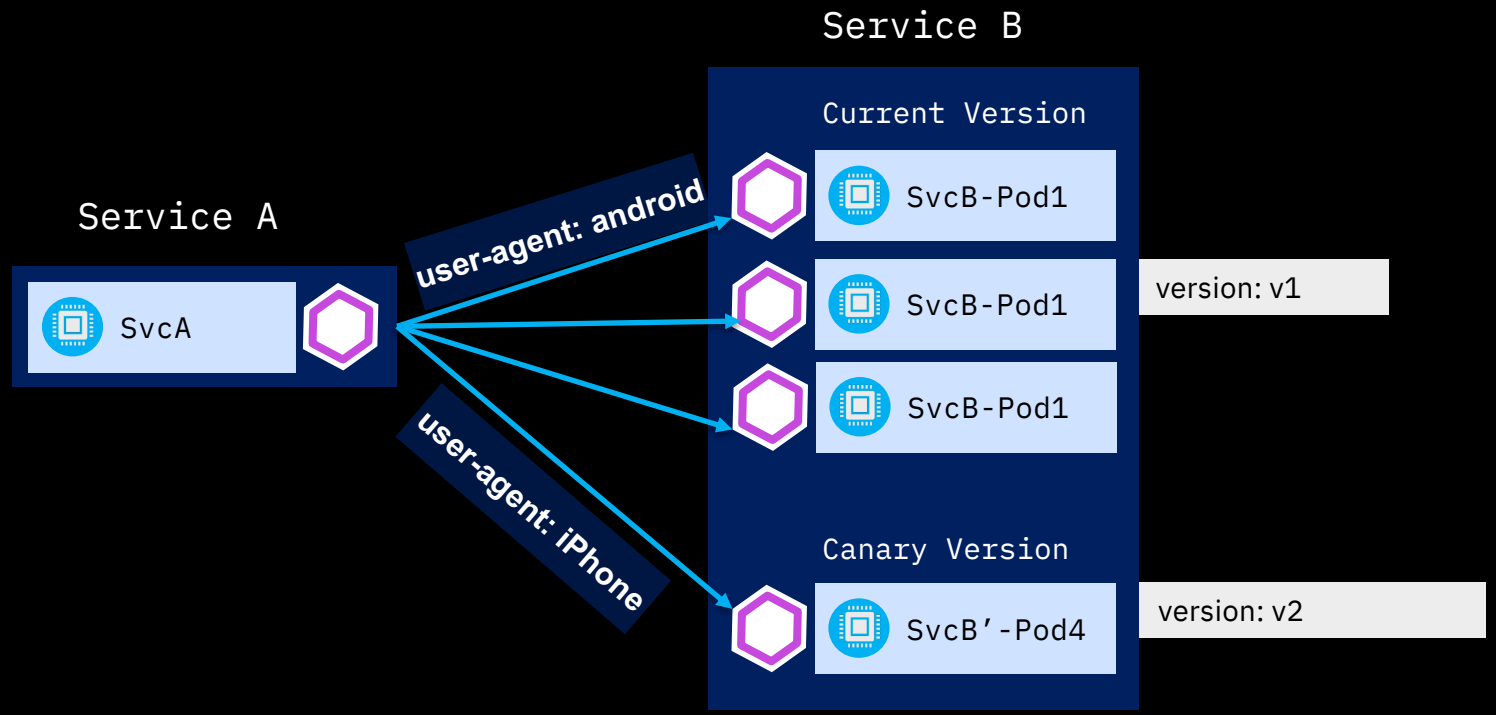



```
kind: VirtualService
destination:
  serviceB.cluster.local
match:
  source:
    serviceA.cluster.local
route:
  - labels:
    version: v1.5
    env: us-prod
  weight: 95
  - labels:
    version: v2.0-alpha
    env: us-staging
  weight: 5
```

CHALLENGE 2
HOW TO DO CANARY TESTING

Routing not based on the request content.
Staged rollouts with %-based traffic splits.

Traffic Steering

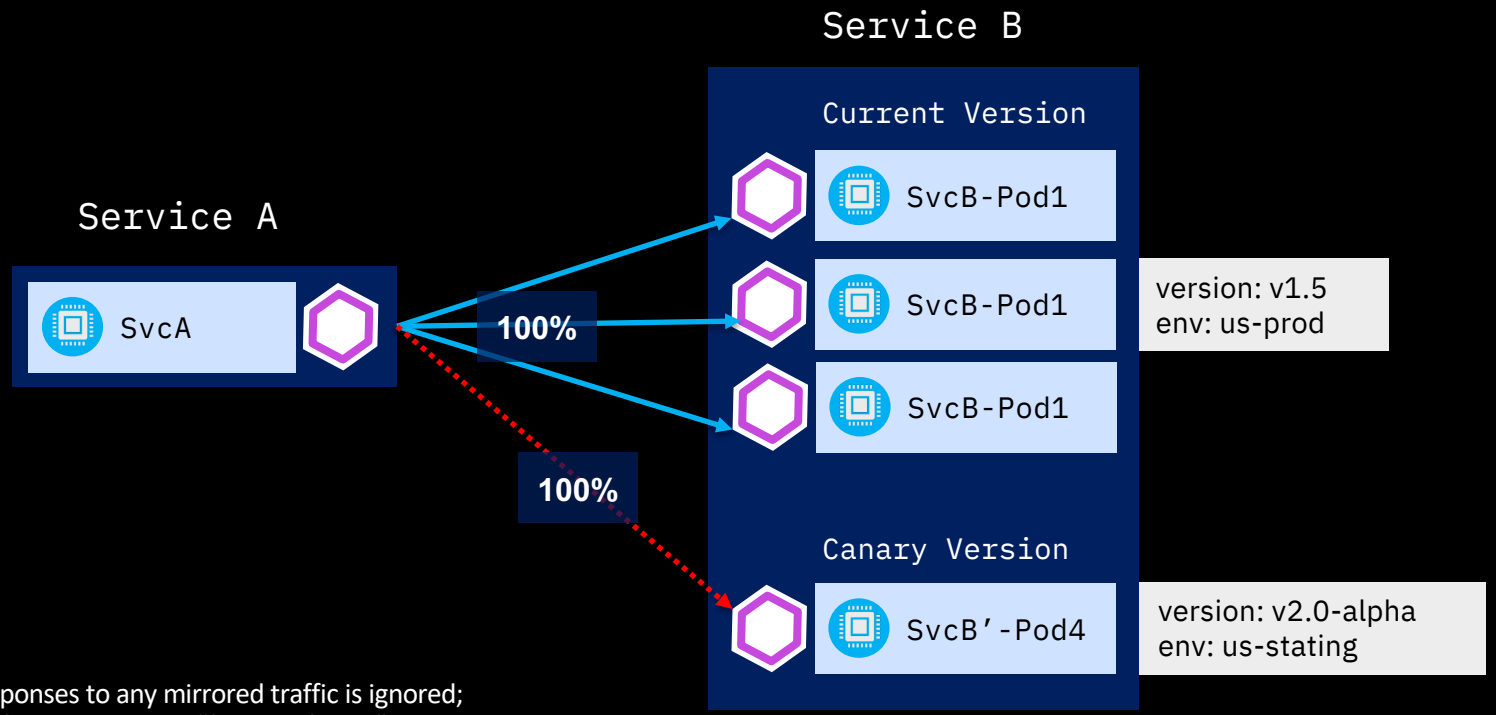



```
kind: VirtualService
destination:
 serviceB.cluster.local
match:
  httpHeaders:
    user-agent:
      regex:
        ^(.*?;)?(iPhone)(;.*)?$
 route:
- labels:
  version: v2
```

Routing based on the request content

CHALLENGE 3
HOW TO DO A/B TESTING

Traffic Mirroring



Responses to any mirrored traffic is ignored;
traffic is mirrored as "fire-and-forget"
You'll need to have the 0-weighted route to hint to
Istio to create the proper Envoy

```
kind: VirtualService
destination:
  serviceB.cluster.local
match:
  source:
    serviceA.cluster.local
route:
  - labels:
    version: v1.5
    env: us-prod
  weight: 100
  - labels:
    version: v2.0-alpha
    env: us-staging
  weight: 0
  mirror:
    name: httpbin
    labels:
      version: v2.0-alpha
      env: us-staging
```

CHALLENGE 4
THINGS DON'T ALWAYS GO CORRECTLY IN PRODUCTION...

Resiliency

Istio adds fault tolerance to your application without any changes to code

```
// Circuit breakers
destination: serviceB.example.cluster.local
policy:
- labels:
    version: v1
  circuitBreaker:
    simpleCb:
      maxConnections: 100
      httpMaxRequests: 1000
      httpMaxRequestsPerConnection: 10
      httpConsecutiveErrors: 7
      sleepWindow: 15m
      httpDetectionInterval: 5m
```

Resilience features

- Timeouts
- Retries with timeout budget
- Circuit breakers
- Health checks
- AZ-aware load balancing w/ automatic failover
- Control connection pool size and request load

CHALLENGE 4
THINGS DON'T ALWAYS GO CORRECTLY IN PRODUCTION...

Rate limiting

Istio protects your application from rogue actors by imposing ratelimits

```
Quotas:
- name: requestcount.quota.istio-system
  maxAmount: 5000
  validDuration: 1s
  overrides:
  - dimensions:
      destination: ratings
      source: reviews
      sourceVersion: v3
    maxAmount: 1
    validDuration: 1s
  - dimensions:
      destination: ratings
    maxAmount: 100
    validDuration: 1s
```

Rate limit

- Configurable limits with overrides
- Multiple rate limiting backends
- Conditional rate limiting

CHALLENGE 5
HOW CAN I LIMIT RATE FOR SOME OF MY SERVICES?

Telemetry

Monitoring & tracing should not be an afterthought in the infrastructure

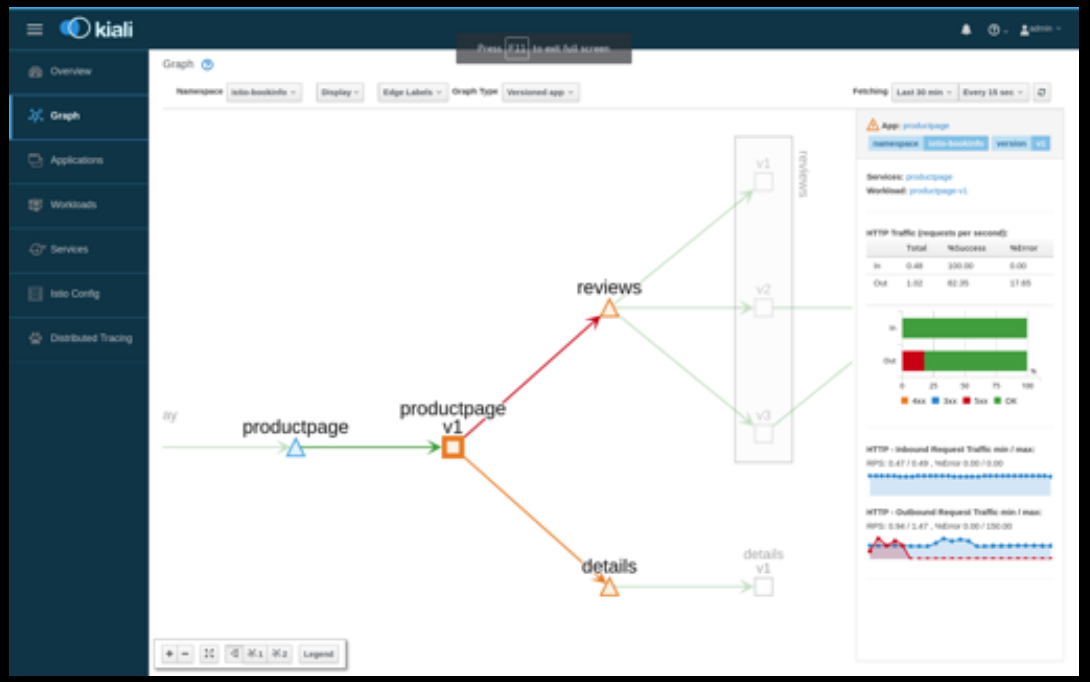
Goals

- Metrics without instrumenting apps
- Consistent metrics across fleet
- Trace flow of requests across services
- Portable across metric backend providers

CHALLENGE 6
I NEED TO VIEW WHAT IS GOING ON WHEN CRISIS ARISES

Kiali

Kiali (greek κιάλι)
monocular or *spyglass*
Visualise the service mesh topology, features like circuit breakers or request rates

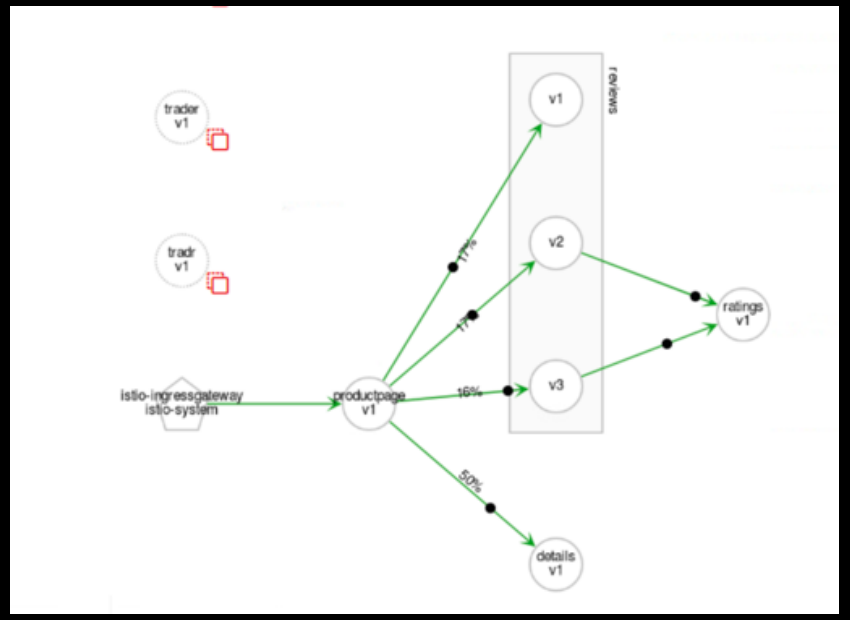


Features

Graph
- Health
- Types
- Side Panel
- Traffic Animation

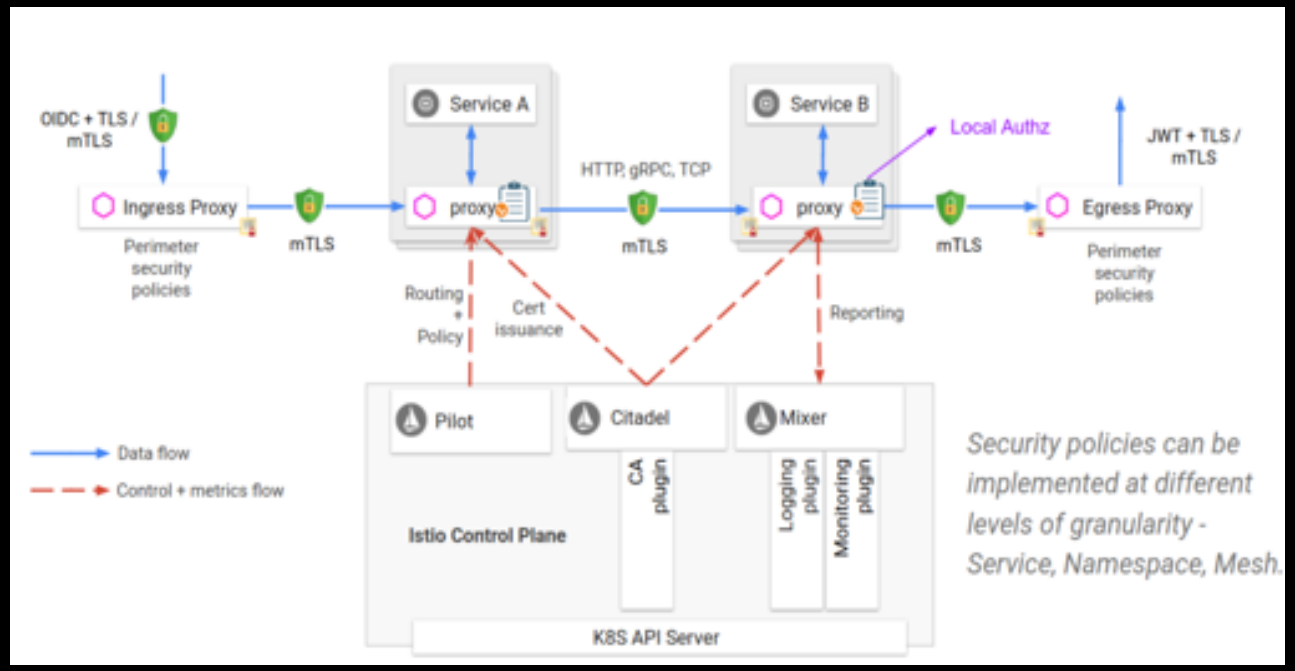
Applications, Workloads and Services
- Detailed Metrics

Traffic Routing
Istio compliance
Istio Configuration



CHALLENGE 6
I NEED TO VIEW WHAT IS GOING ON WHEN CRISIS ARISES

Security



Authentication

- Transport authentication, also known as service-to-service authentication
- Origin authentication, also known as end-user authentication

Authorization

- Based on RBAC
- Namespace-level, service-level and method-level access control for services

CHALLENGE 7
HOW CAN I SECURE MY SERVICES?

Service Mesh - Bad Idea ?

A Service Mesh is not always the right solution...

- **Service Meshes are Opinionated**
  They are a *platform* solution. “Work their way”
- **Service Meshes are Complex**
  Adds considerable complexity with sidecars and control plane
- **Service Meshes can be Slow**
  Routing traffic through a series of proxies can get painfully slow (about 700 nodes → reflector)
- **Service Meshes are for Developers**
  Focused primarily on Developer view.

https://glasnostic.com/blog/comparing-service-meshes-linkerd-vs-istio

IBM

QUESTIONS?

IBM **Cloud**

Thank you

Niklaus Hirt
Cloud / DevOps Architect
—
nikh@ch.ibm.com
+41 79 948 72 46
ibm.com


IBM