The Journey to Cloud

DevSecOps *and* From Microservices to Kubernetes

Niklaus Hirt
DevOps Architect / Cloud Architect
nikh@ch.ibm.com

IBM

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 – DevSecOps & Kubernetes Basic Concepts

Module 0: Prepare the Labs

Module 1: DevSecOps

BREAK

Module 2: Microservices

Module 3: Containers with Docker

Module 4: Kubernetes

Module 5: Let’s get real

Module 6: Kubernetes Security

LUNCH

Agenda – DevSecOps & Kubernetes Basic Concepts

Module 7: Docker Hands-On

Module 8: Kubernetes Hands-On

Module 9: Kubernetes Security Hands-On (Optional)

Wrap-up

END of course – 16:00

Documentation

https://github.com/niklaushirt/k8s_training_public

Sources

https://github.com/niklaushirt/training

Always-on training environment

http://158.177.137.195:31999/

The Journey to Cloud
Prepare the Labs

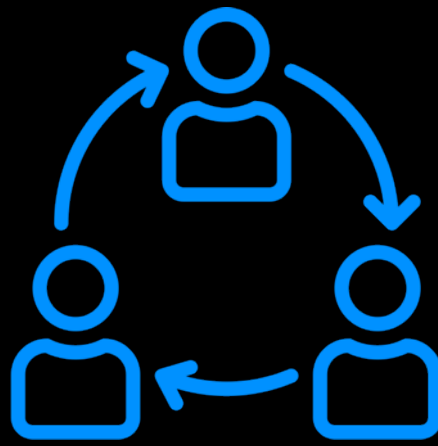
00

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

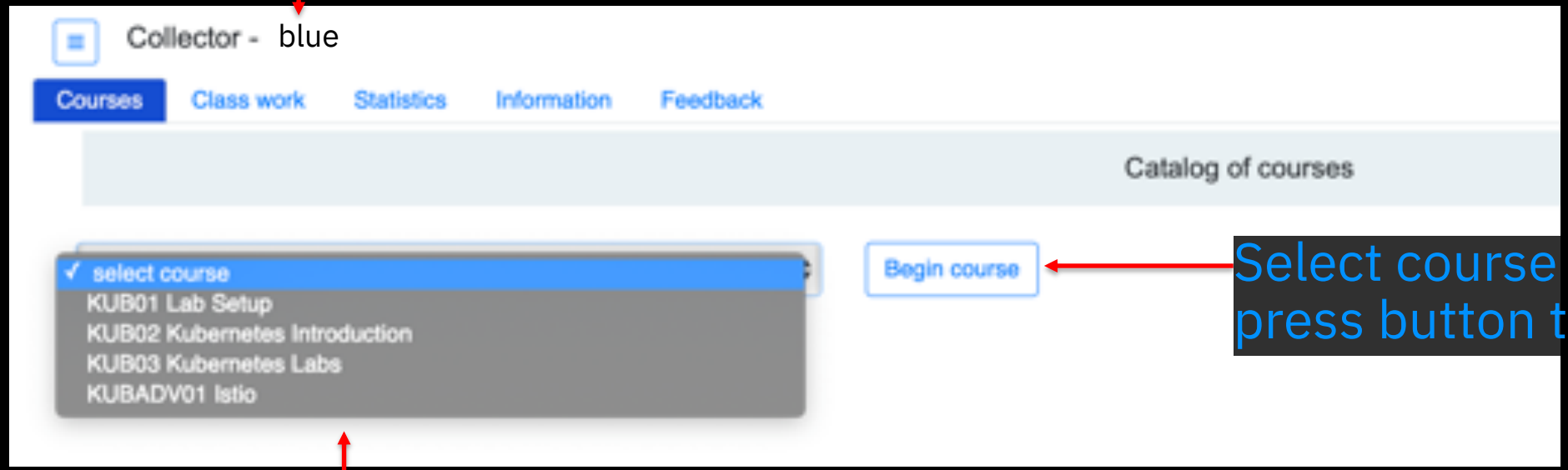
black 31701	olive 31711	peru 31715
white 31702	brown 31712	chocolate 31716
red 31703	lightblue 31713	orchid 31717
blue 31704	orange 31708	gold 31718
yellow 31705	purple 31709	pink 31719
lime 31706	maroon 31710	violet 31720
cyan 31707	firebrick 31714	

Collector - Accessing team web site

http://158.177.137.195:{port#}

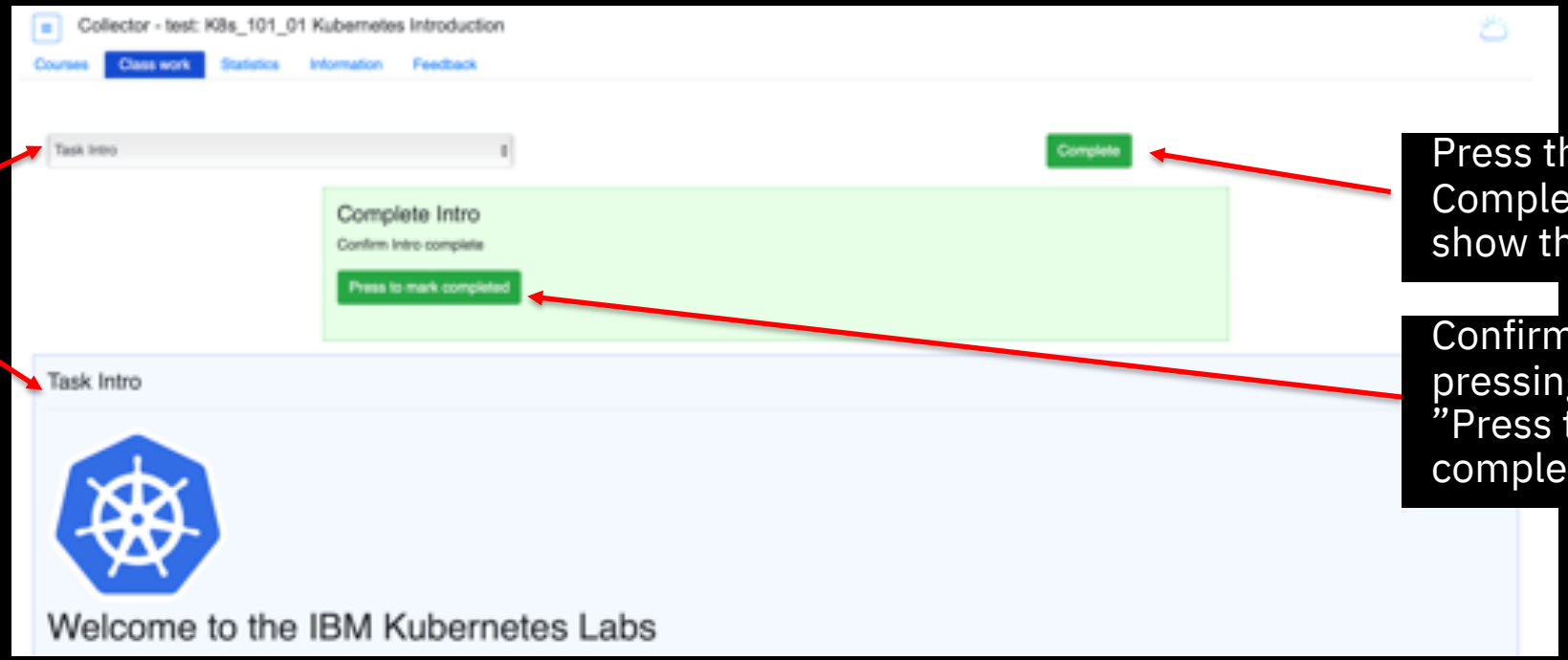
blue 31704

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

EVERYBODY OK ?

Task 1: Setup Minikube

Task 2: Setup kubectl

Task 3: Setup git

Task 4: Final Check

IBM

The Journey to Cloud
DevSecOps

01

IBM **Cloud**

Kodak

disruption

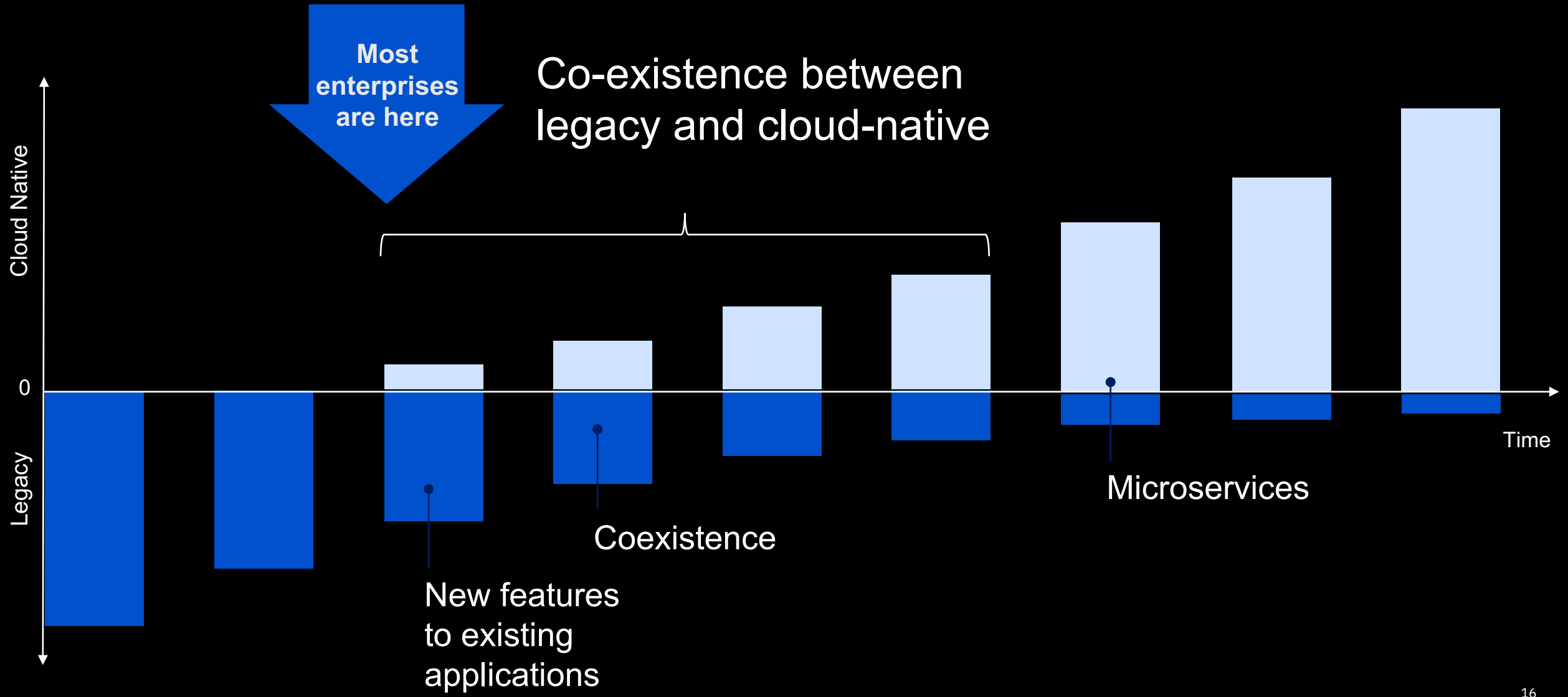
dɪsˈrʌpʃn/

noun

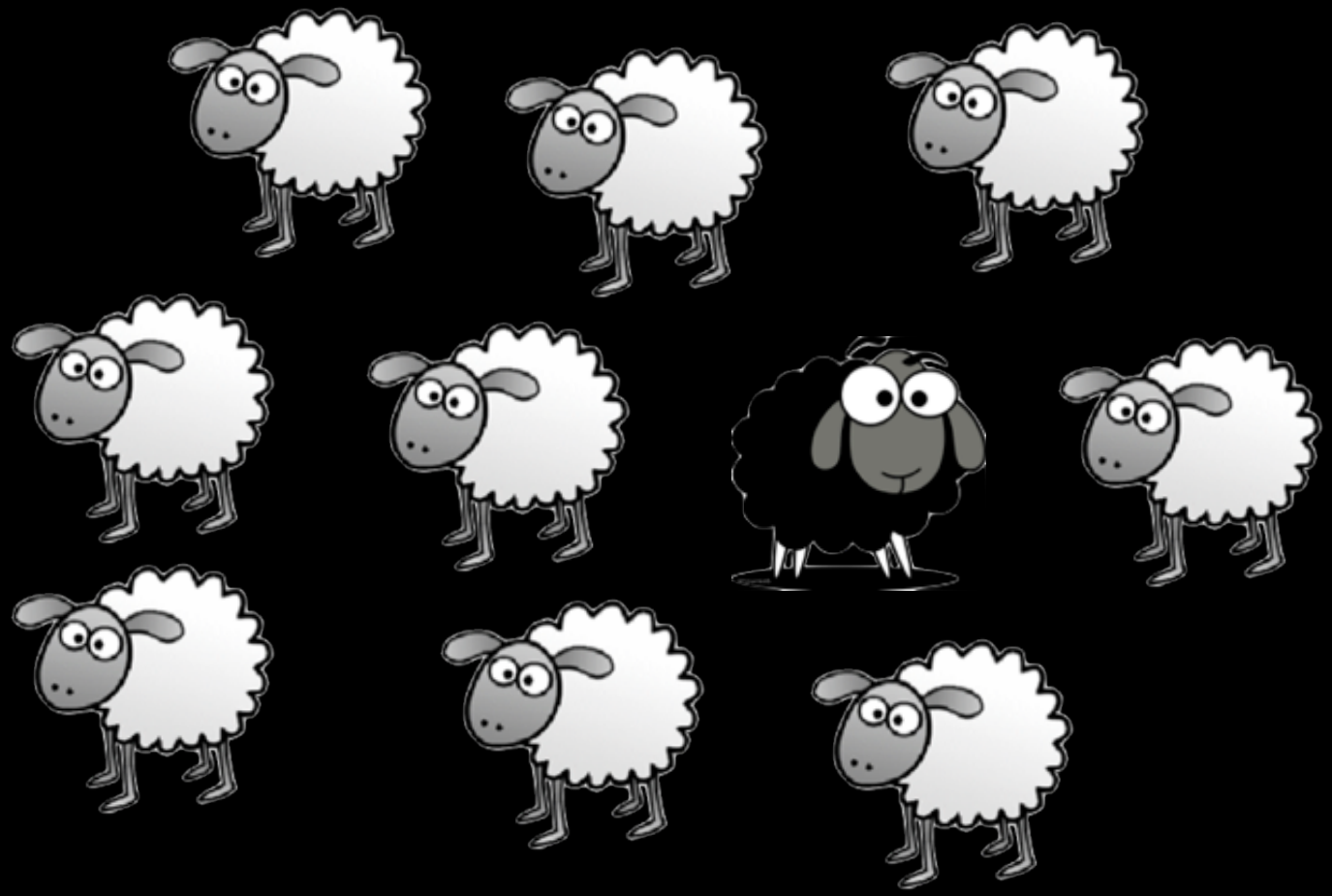
Business. a **radical change** in an industry, business strategy, etc., especially involving the introduction of a **new product** **or service** that creates a **new market**

Digital Transformation - Way to go

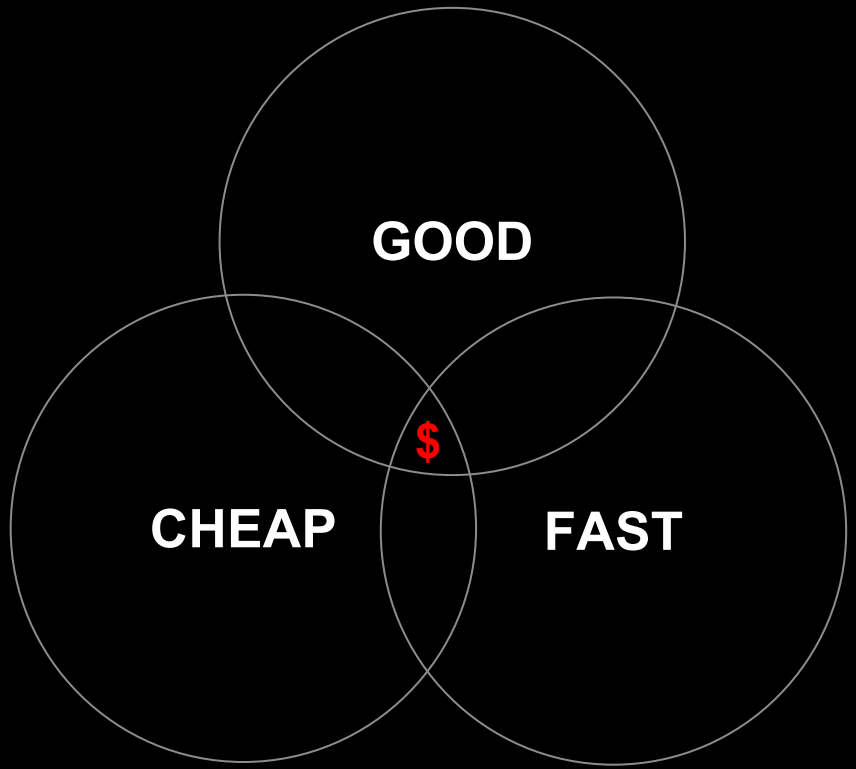
Cloud native and legacy apps will co-exist for the next 10+ years



Disruption is new reality



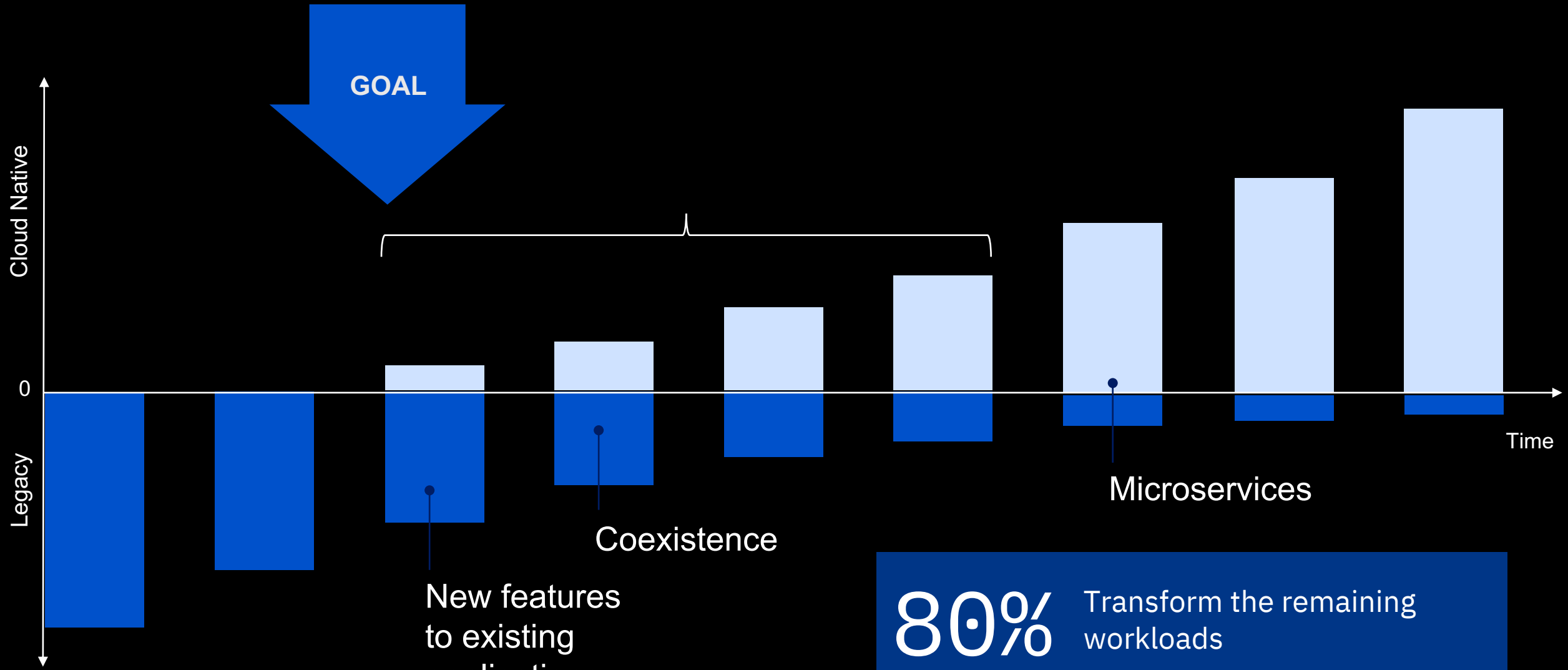
Need to optimize



Digital Transformation is the new normal

Digital Transformation

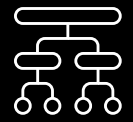
Cloud native and legacy apps will co-exist for the next 10+ years



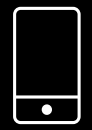
80% Transform the remaining workloads

Digital Transformation – How to get there

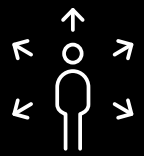
From



Central uniform management processes



Monolithic, proprietary apps



Infrastructure-based management

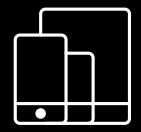
Adopt new **Processes**

Adopt new **Technology**

Adopt **Cloud**

To

Continuous Integration
Continuous Build
Continuous Deploy
Agile Dev Models



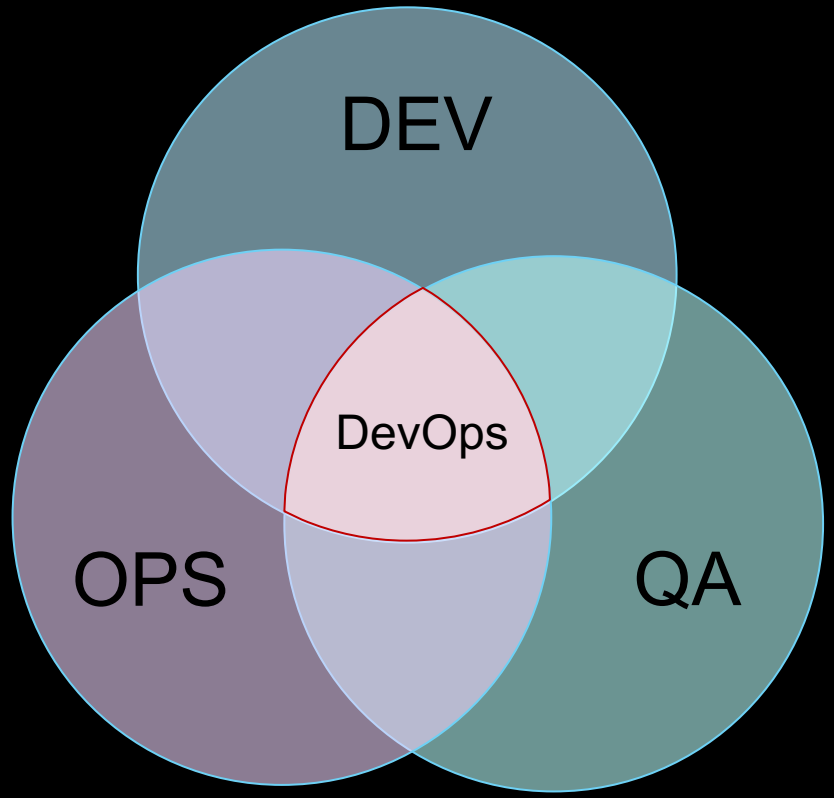
Open, scalable cloud-native applications
Architectural patterns (Microservices)
Frameworks (Spring, NodeJS, ...)

Application-centric management
Docker, Kubernetes
Virtualization, VMs
Monitoring, Dashboards, Alerts

DevOps to the rescue

What is DevOps

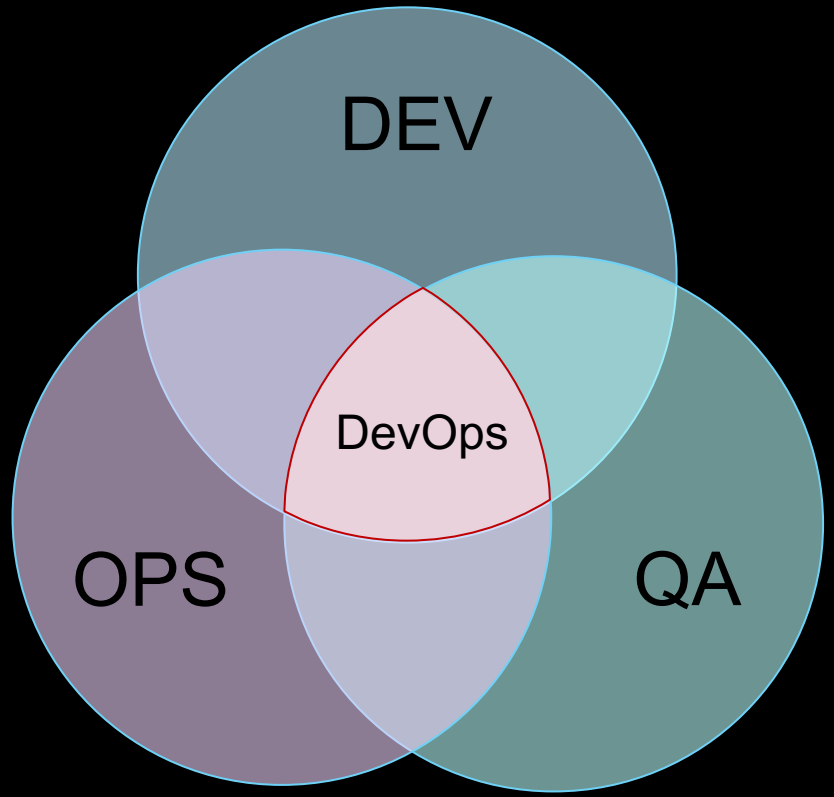
DevOps according to Wikipedia™



DevOps is a software development method that stresses communication, collaboration and integration between software developers and IT professionals.

DevOps is a response to the interdependence of software development and IT operations. It aims to help an organization rapidly produce software products and services.

DevOps according to Wikipedia™



CULTURE 🏛 **PROCESS**

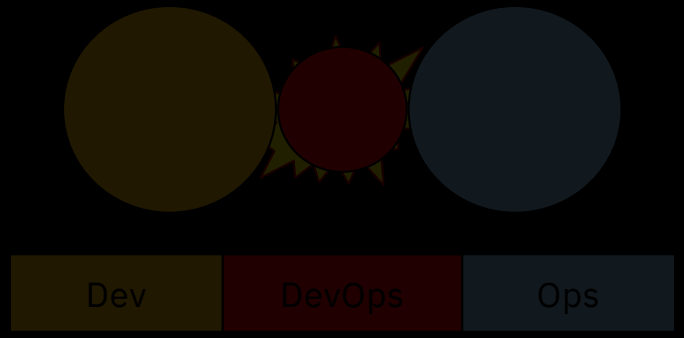
DevOps is a software development **method** that stresses **communication**, **collaboration** and integration between software **developers** and **IT professionals**

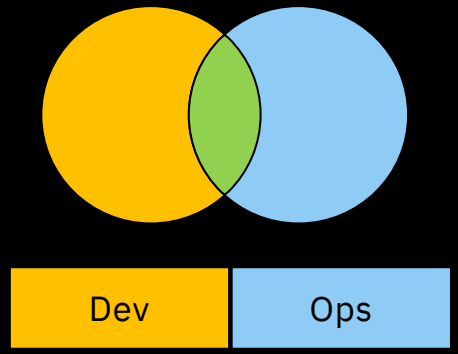
PEOPLE 🏛

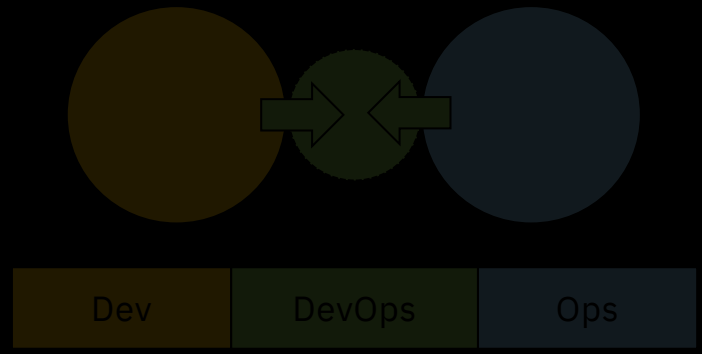
Tools, Processes
AND **People**!

DevOps - Culture

CULTURE





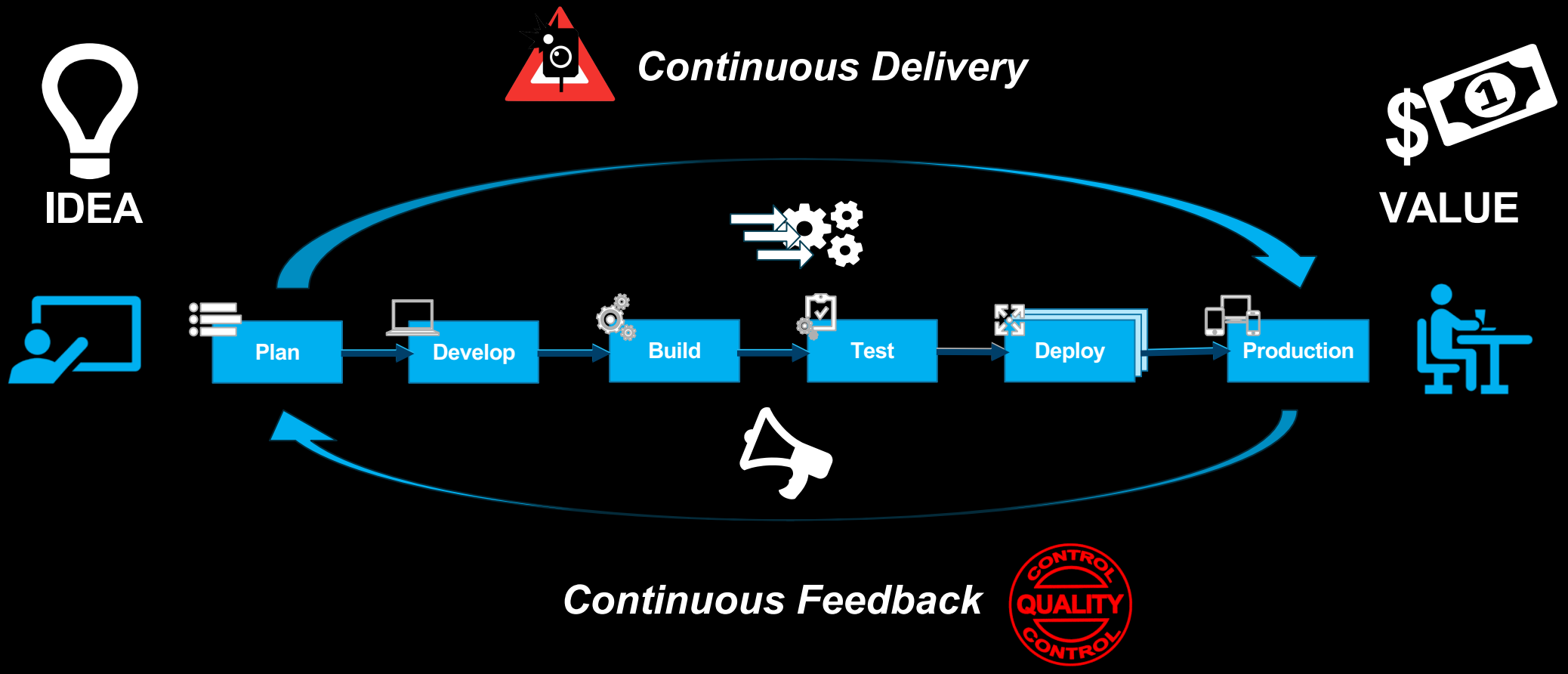


Matthew Skelton – DevOps Adoption Patterns - http://web.devopstopologies.com

DevOps – Process

Business Success - get capabilities to the customers

PROCESS



DevOps – Adoption

What DevOps is NOT

ADOPTION

- DevOps is not simply ***combining Development & Operations teams***
- DevOps is not a ***separate team***
- DevOps is not only a ***tool***
- DevOps is not only ***automation***
- DevOps is not a ***one-size-fits-all strategy***

Shelbee Smith-Eigenbrode http://devops.com/2017/03/17/what-devops-is-not/

DevOps – Adoption

Getting started on the DevOps journey

- **Be honest**
  - **Assess** your own DevOps maturity and aspirations – where are you and where do you want to be?
- **Don’t go nuts**
  - **Experiment** on a few smaller, low-risk projects – set up a couple of **“two pizza” teams** comprised of Dev/QA + Ops
  - Consider creating a (temporary!) **DevOps "center of excellence"**
- **Cultural change takes time – take reasonable steps**
  - Work on **shifting culture** through teambuilding, cross-training, improved communication, perhaps more collaborative tooling
- **Don’t try to change the laws of physics**
  - Reality is that your “Systems of Record” applications move at a **different pace** than your “Systems of Engagement” apps (e.g. mobile)

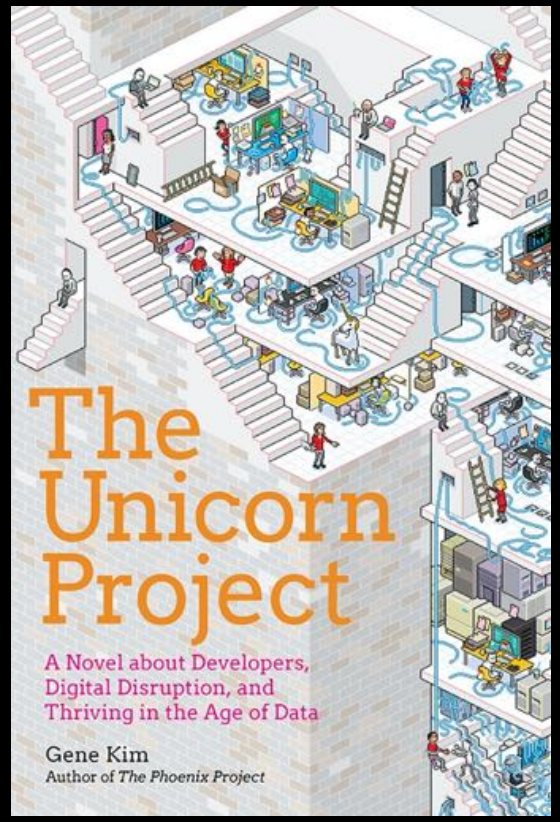
DevOps – Adoption

Remember you must

Start small you should

DevOps – Adoption

Guidance



A Novel about Developers, Digital Disruption, and Thriving in the Age of Data

DevOps – Adoption

The Five Ideals as of Gene Kim

«We’re QA.
We protect the organization from developers.
…
Do not trust them.
Do not get chummy with them.
You give developers an inch, and they’ll take a mile.»

from The Unicorn Project by Gene Kim

DevOps – Adoption

The Five Ideals as of Gene Kim

THE FIRST IDEAL:	Locality and Simplicity
THE SECOND IDEAL:	Focus, Flow, and Joy
THE THIRD IDEAL:	Improvement of Daily Work
THE FOURTH IDEAL:	Psychological Safety
THE FIFTH IDEAL:	Customer Focus

«…especially when you have **customers, change is a fact of life**.
A healthy software system is one that you can **change at the speed** you need, where people can **contribute easily**, without jumping through hoops.
This is how you make a project that's **fun and worthwhile** contributing to, and where you often find the most vibrant communities.»
from "The Unicorn Project by Gene Kim

Source: The Unicorn Project by Gene Kim

DevOps – Adoption

The Five Ideals as of Gene Kim

THE FIRST IDEAL : Locality and Simplicity

Locality in Systems

- Teams need to be able to build and test their changes locallly and deploy them without implicating 10 other teams

Organization

- Every team has the expertise, capability and authority to satisfy customer needs

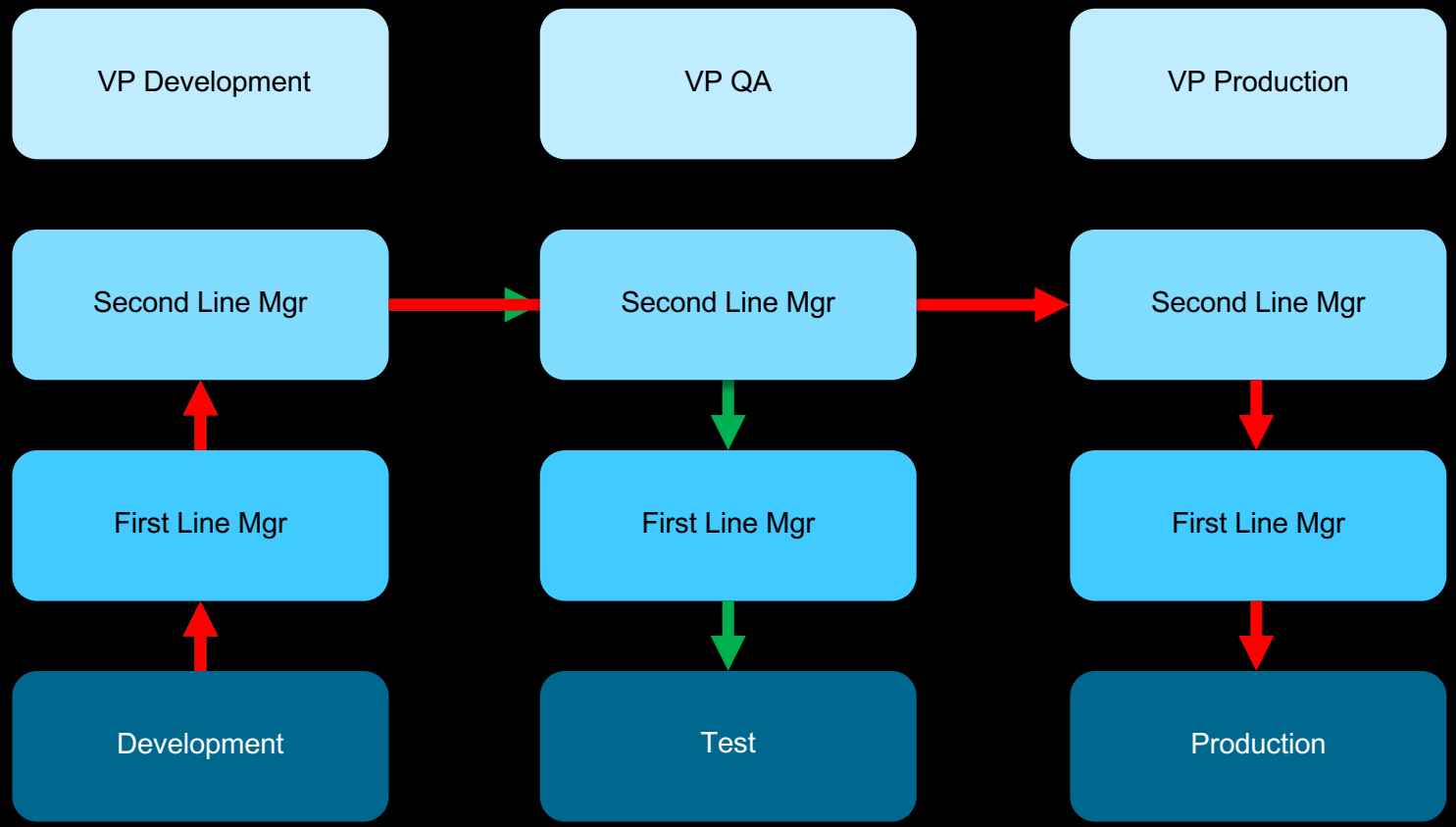
Simplicity

- Uncouple applications from each other
- Architectural patterns
- Prioritize technical debt reduction
- Two Pizza teams

Source: The Unicorn Project by Gene Kim

DevOps – Adoption

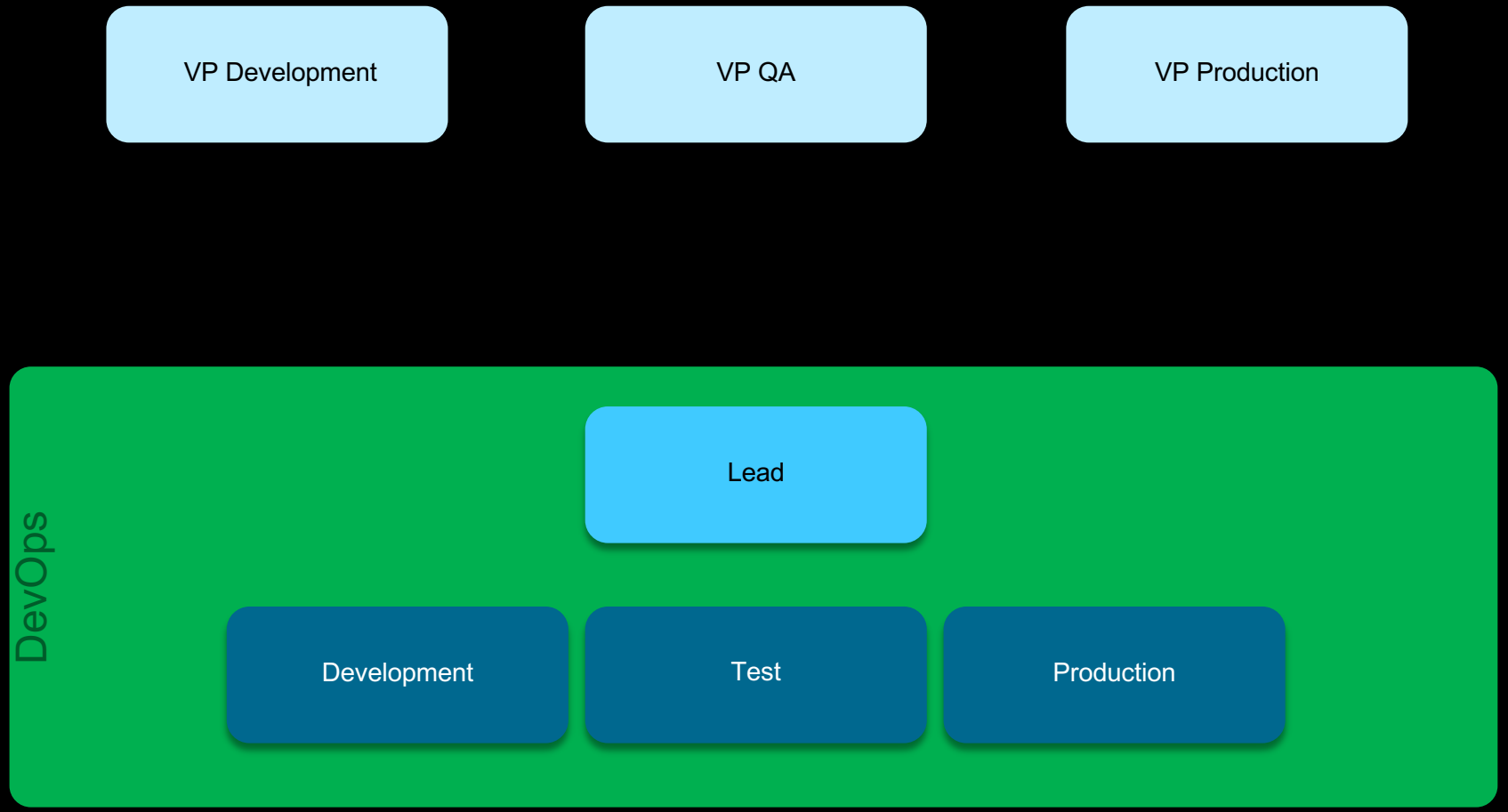
The Five Ideals as of Gene Kim



Source: The Unicorn Project by Gene Kim

DevOps – Adoption

The Five Ideals as of Gene Kim

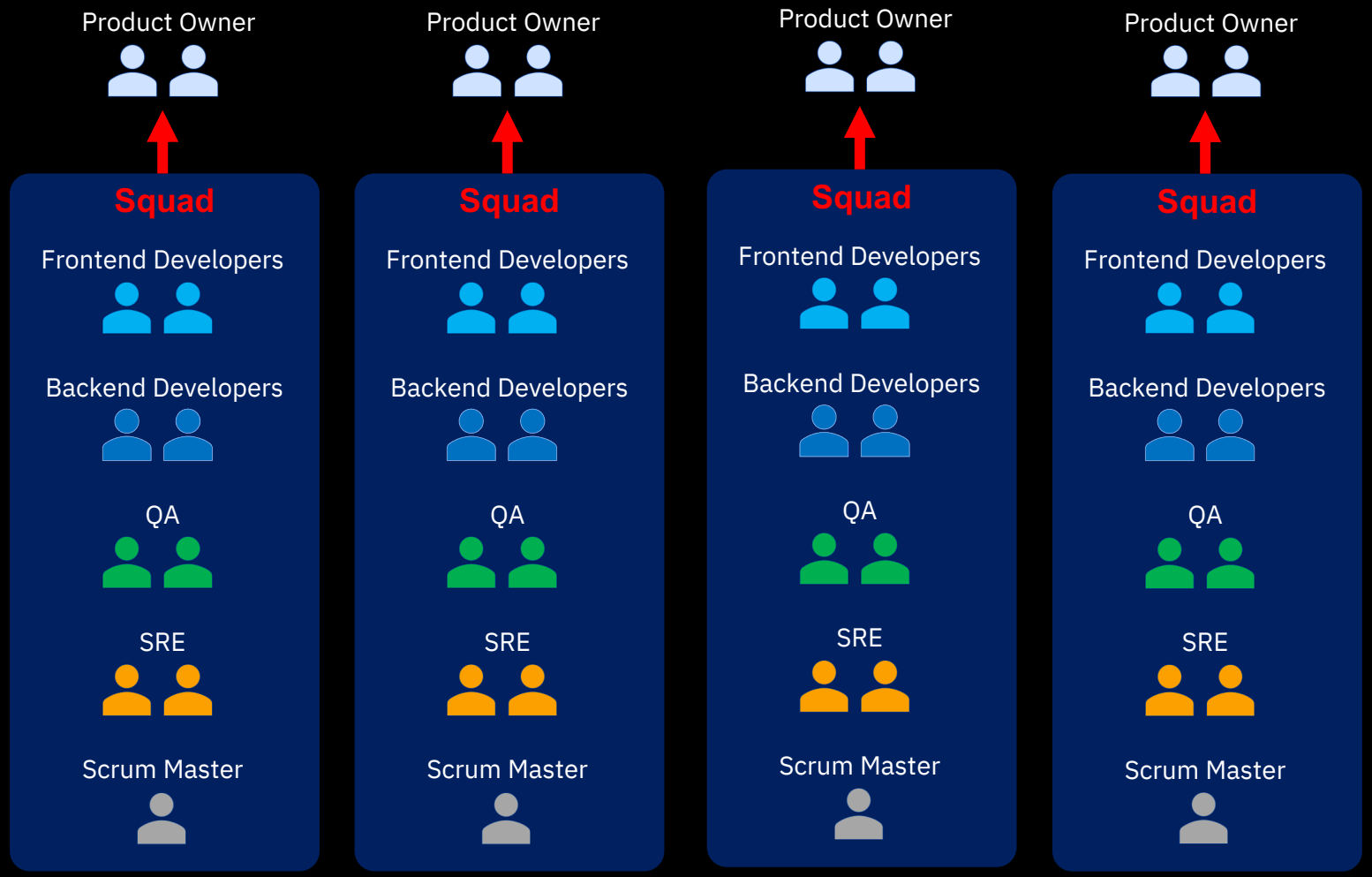


Source: The Unicorn Project by Gene Kim

DevOps – Adoption

The Five Ideals as of Gene Kim

Agile Team Organisation: Squads



Squads contains more than just developers – remove „stigma“

→ rename „Teams“ to „Squads“ 😉 or Crew, Party, …

Fully autonomous, cross functional team that has full responsibilities for an application component/functionality/ microservice and little to no dependencies on others

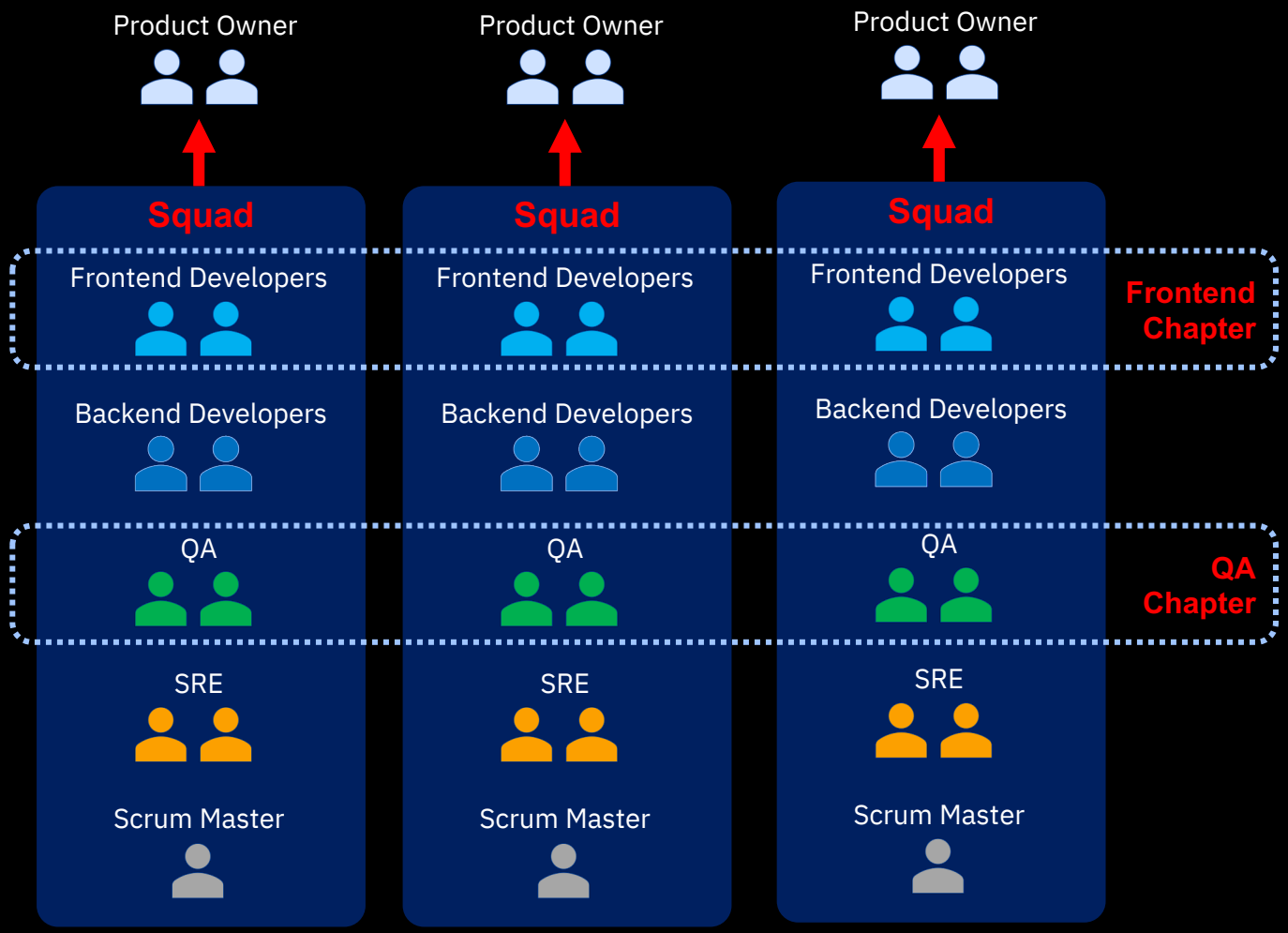
5-7 people ideally

Source: https://medium.com/@achardypm/agile-team-organisation-squads-chapters-tribes-and-guilds-80932ace0fdc

DevOps – Adoption

The Five Ideals as of Gene Kim

Agile Team Organisation: Chapters



Chapters are a group or team members working within a special area

Relay experience to the rest of the chapter and discuss on how other squads can use it.

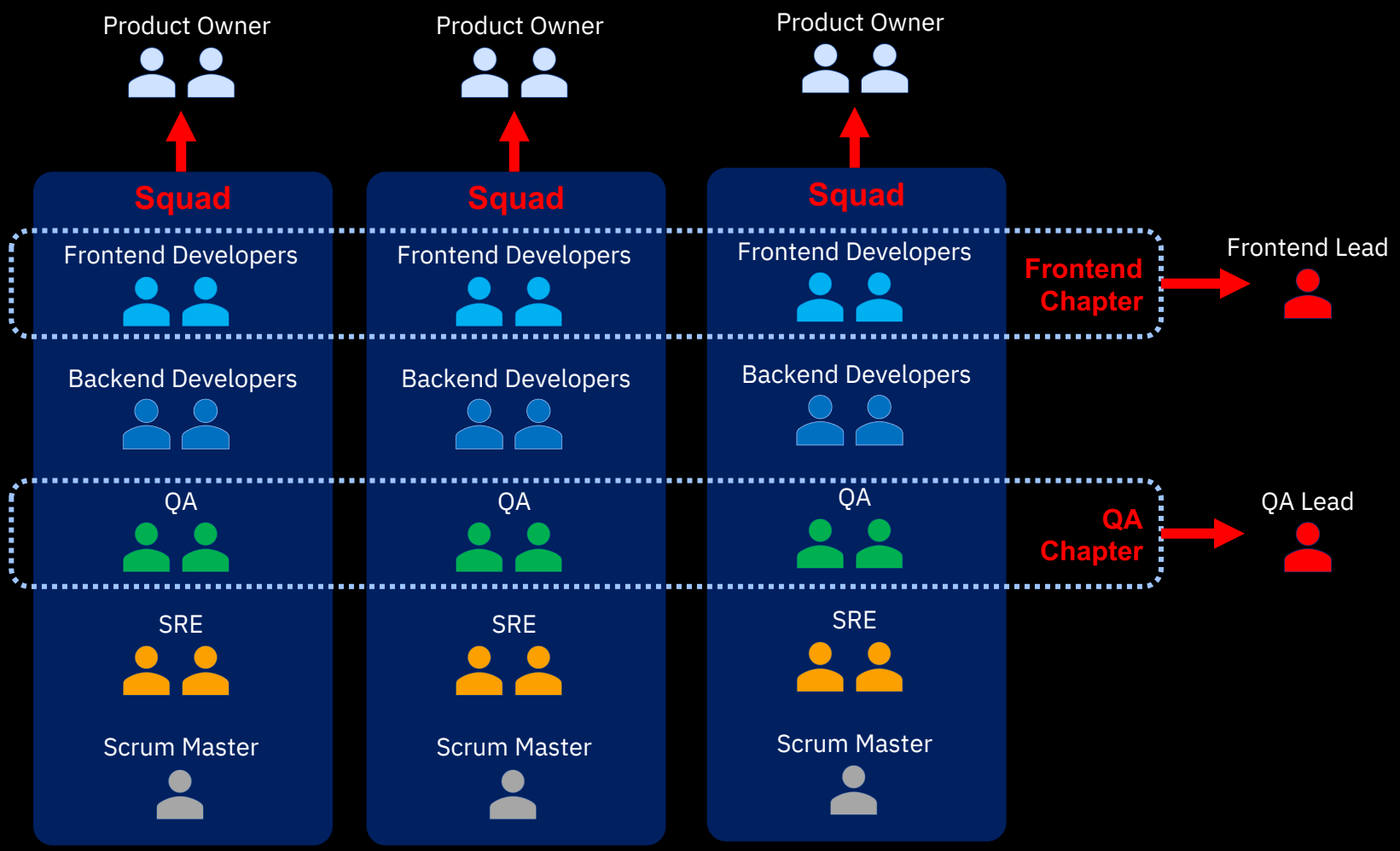
Promotes team collaboration and innovation.

Source: https://medium.com/@achardypm/agile-team-organisation-squads-chapters-tribes-and-guilds-80932ace0fdc

DevOps – Adoption

The Five Ideals as of Gene Kim

Agile Team Organisation: Chapters



A **Chapter Lead** is the line manager for chapter members.

Responsible for developing people .

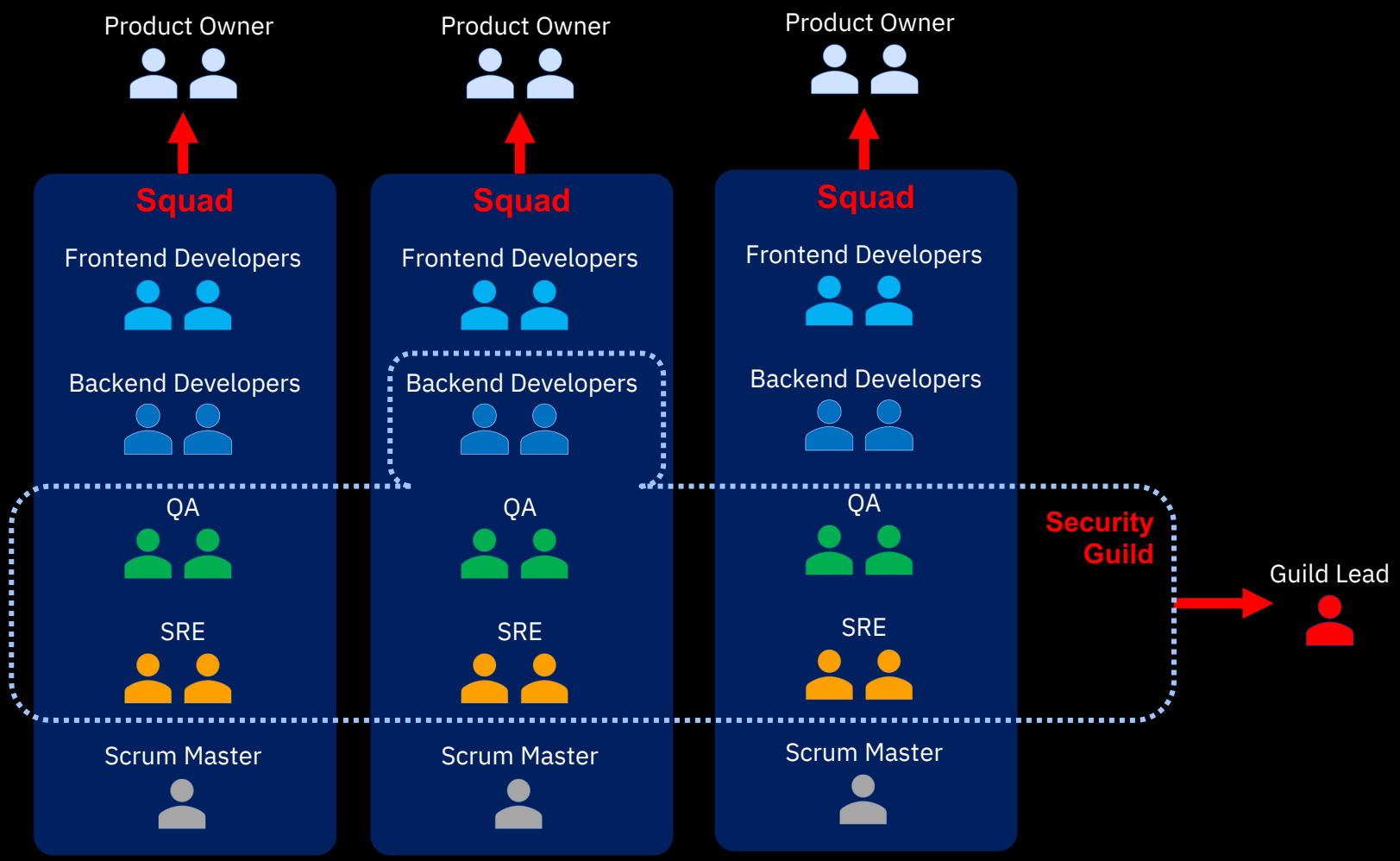
Remain part of a squad and still do day-to-day work.

Source: https://medium.com/@achardypm/agile-team-organisation-squads-chapters-tribes-and-guilds-80932ace0fdc

DevOps – Adoption

The Five Ideals as of Gene Kim

Agile Team Organisation: Guilds



Guilds cover a problem or something that can be improved and get together a small group of team members together to solve it.

Examples:

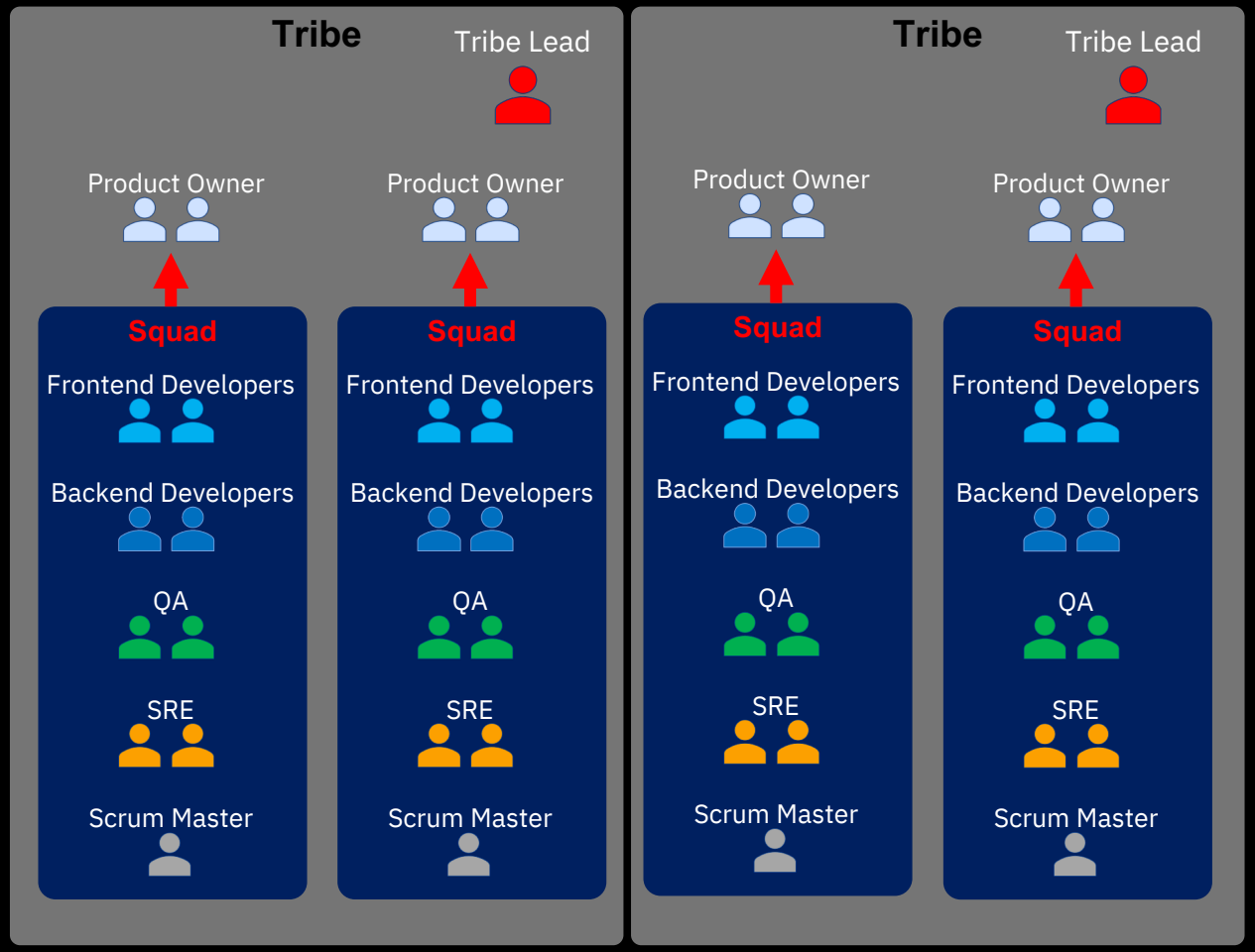
- Performance monitoring / optimization
- Testing automation
- Security & vulnerabilities
- Services & Architecture

Source: https://medium.com/@achardypm/agile-team-organisation-squads-chapters-tribes-and-guilds-80932ace0fdc

DevOps – Adoption

The Five Ideals as of Gene Kim

Agile Team Organisation: Tribes



Tribes are a collection of squads that work in related areas like mobile app, backend infrastructure, ...

"Incubator" for the **Squad** mini-startups

Challenging to implement

Only for large complicated infrastructure that needs to be split this way

< 100 people (Dunbar number)

Source: https://medium.com/@achardypm/agile-team-organisation-squads-chapters-tribes-and-guilds-80932ace0fdc

DevOps – Adoption

The Five Ideals as of Gene Kim

THE SECOND IDEAL: Focus, Flow, and Joy

Focus

- Focus on developing their code with minimum dependencies, delays and impediments

Flow

- Focus creates flow of value, therefore joy

Joy

- Energy and time is focused on solving the business problem, and you're having fun

Source: The Unicorn Project by Gene Kim

DevOps – Adoption

The Five Ideals as of Gene Kim

Enable developer productivity

- Self-service
- On-demand
- Immediacy and fast feedback
- Focus and flow

Core concepts

- Immutability
- Composability

Look at...

- Microservices
- Containers
- Kubernetes

Source: The Unicorn Project by Gene Kim

DevOps – Adoption

The Five Ideals as of Gene Kim

THE THIRD IDEAL: Improvement of Daily Work

Technical Debt

- Pay down technical debt

Architecture

- Enables teams to independently develop, test, and deploy value to customers without being coupled to many other teams

Flow

- Flow with as little interruption and impediments as possible

Source: The Unicorn Project by Gene Kim

DevOps – Adoption

The Five Ideals as of Gene Kim

«**Technical Debt** is what you feel the next time you want to make a change.»

Usually refers to things we need to clean up, or where we need to create or restore simplicity, so that that we can quickly, confidently, and safely make changes to the system

Ward Cunningham in 2003

Source: The Unicorn Project by Gene Kim

DevOps – Adoption

The Five Ideals as of Gene Kim

THE FOURTH IDEAL: Psychological Safety

Safety

- Team members feel safe to talk about problems

Blame

- Blameless post-mortems

Feedback

- Create feedback in the system, sooner and faster, with clarity between cause and effect.

Source: The Unicorn Project by Gene Kim

DevOps – Culture

Start by understanding your Flow

Pathological **Power-oriented**	**Bureaucratic** **Rule-oriented**	**Generative** **Performance-oriented**
Low co-operation	Modest co-operation	High co-operation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shriked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented
Characterized by large amounts of fear and threat. People often hoard information or withhold it for political reasons, or distort it to make themselves look better.	Organisations protect departments. Those in the department want to maintain their “turf,” insist on their own rules, and generally do things by the book — their book	Organisations focus on the mission. How do we accomplish our goal? Everything is subordinated to good performance, to doing what we are supposed to do.

Source: The Study of Information Flow: A Personal Journey by Ron Westrum

DevOps – Adoption

The Five Ideals as of Gene Kim

THE FIFTH IDEAL: Customer Focus

Customer Focus

- Customer focus relates to the difference between core and context
- Decisions based on what the customer values
- Make sure that context doesn't kill core

Core - differentiation

- Lasting durable business advantage
- What customers are willing to pay for

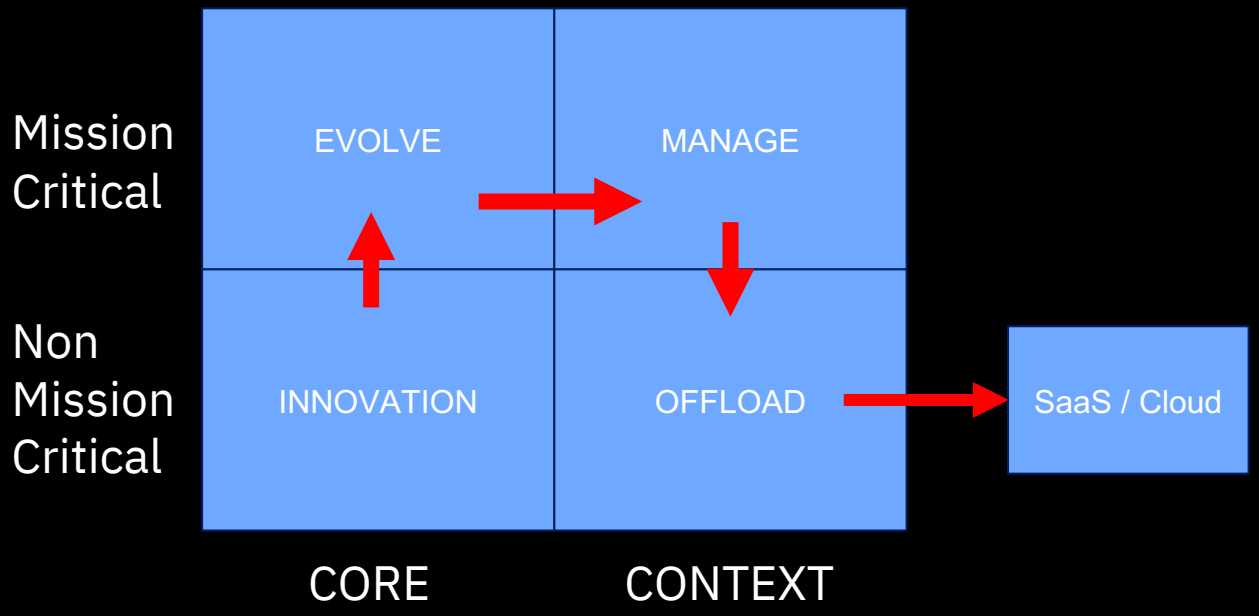
Context - commodity

- Systems that are mission critical, but don’t create competitive advantage
- What customers don’t want / see, like HR systems, payroll, …

Source: The Unicorn Project by Gene Kim

DevOps – Adoption

The Five Ideals as of Gene Kim



New Products start in the bottom left quadrant to create differentiation for customers but have no customer impact initially.

New products will grow in profitability, becoming mission critical.

Customer needs evolve, products can migrate out of the Core to become commodity because competitiors have caught-up.

At the end of life they might become non-critical to the business and can be offloaded.

Source: The Unicorn Project by Gene Kim

DevOps – Adoption

Measuring Adoption

	Elite	High	Medium	Low
Lead time for changes				
The time it takes to go from code committed to code successfully deployed	1 hour	1 day – 1 week	1 week – 1 month	1 month – 6 months
Deployment frequency				
Frequency of *production* deployments - how often you're delivering something of *value* to end users	> 1 per day	~1 per day	1 week	1 month
Time to restore service				
The average time it takes to restore a service (MTTR)	< 1 hour	< 1 day	1 day – 1 week	> 1 week
Change failure rate				
How often deployment failures occur in production that require i	0-15%	0-15%	0-15%	46-60%

... predictive of software delivery, operational performance, and organizational performance, and they correlate with burnout, employee engagement, and so much more."

Source: https://stelligent.com/2018/12/21/measuring-devops-success-with-four-key-metrics/

DevOps – Adoption

ADOPTION

Culture

Culture
- Empower the team - Foster Experimentation
- Generative Actor - "Rogue/TwoPizza Team" - Showcase
- Change takes Time - take reasonable steps
- If something is hard, do it repeatedly

Patterns / Antipatterns
- There should be no (permanent) DevOps Team
- Everyone is DevOps
- Development Executives own their solutions
- Be honest - Assess maturity and aspirations

Management buy-in
- DevOps is about solving a business problem
- DevOps starts at the top – Senior Leadership Team
- Optimize the whole and not the individual 'silos'

Process

Steer
- Continuous Business Planning
- Measure customer value
- Link requirements to releases

Develop/Build
- Continuous Integration
- Agile Methods / SAFe
- Link lifecycle information

Test
- Continuous Testing
- Test Automation
- Test/Service Virtualization
- Make Build Self-Testing

Deploy
- Continuous Deployment
- Automation and Infrastructure Patterns
- Provide Self-Service
- Containers

Operate
- Continuous Feedback
- Monitor resources
- Optimize to customer KPIs continuously
- Automate problem isolation and resolution

Tools

Steer
- Rational Doors Next Generation
- cloudMatrix
- Jira
- Rational Team Concert
- Rational Focal Point

Develop/Build
- Rational Team Concert
- IBM MobileFirst Platform
- Git
- Jenkins / Bamboo
- Jira

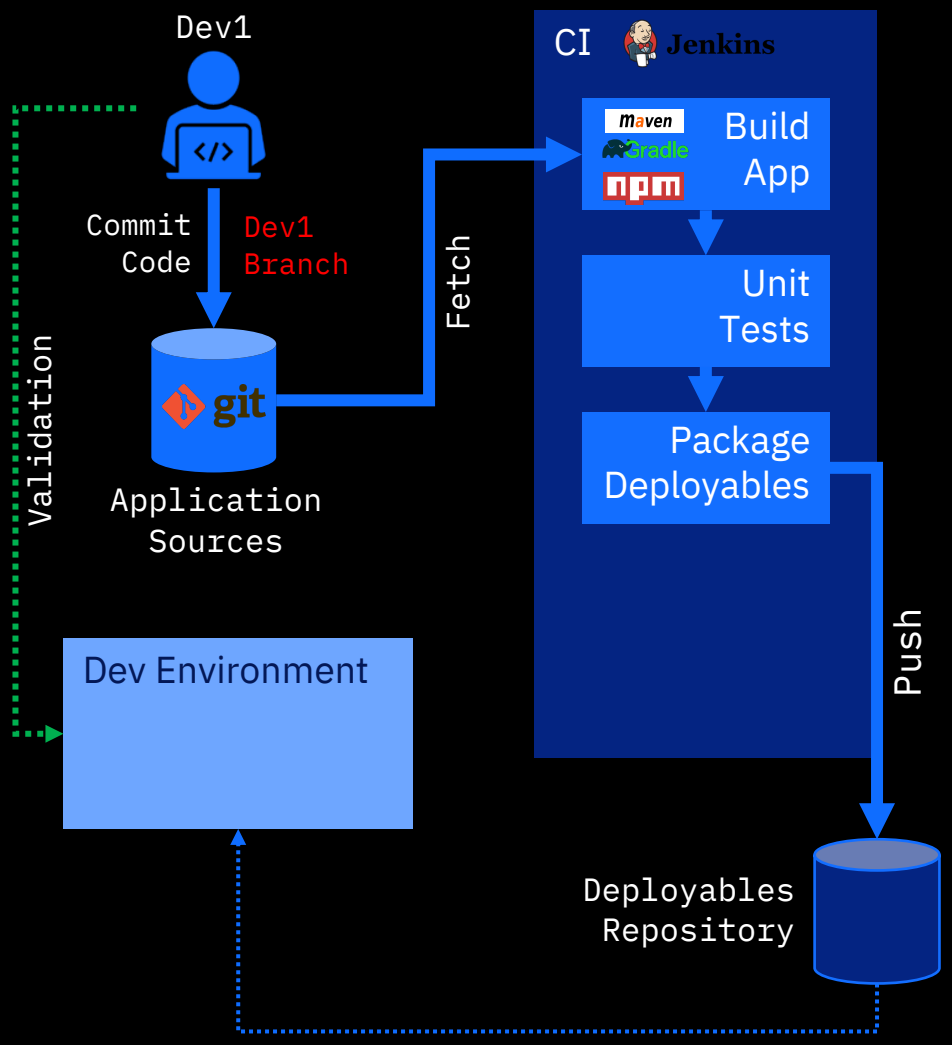
Test
- Rational Quality Manager
- Rational Test Workbench
- Rational Test Virtualization Server
- IBM Security AppScan
- Selenium

Deploy
- IBM UrbanCode Deploy
- IBM Cloud Orchestrator
- Terraform
- Chef Scripts
- Puppet Scripts

Operate
- Tealeaf Customer Behavior Analysis Suite
- IBM Control Desk
- IBM Application Performance Monitoring

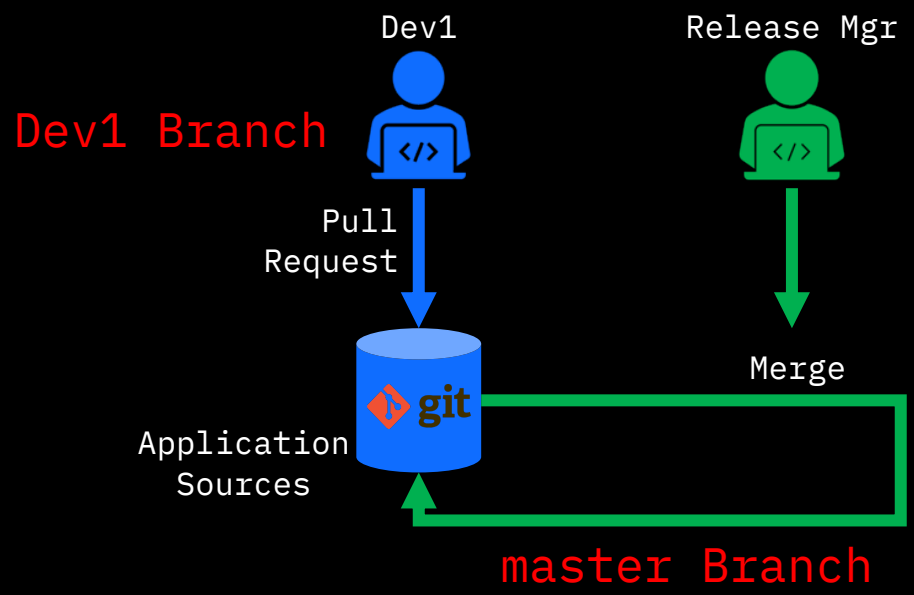
DevOps – CI/CD Pipeline

Development – Dev1 Branch



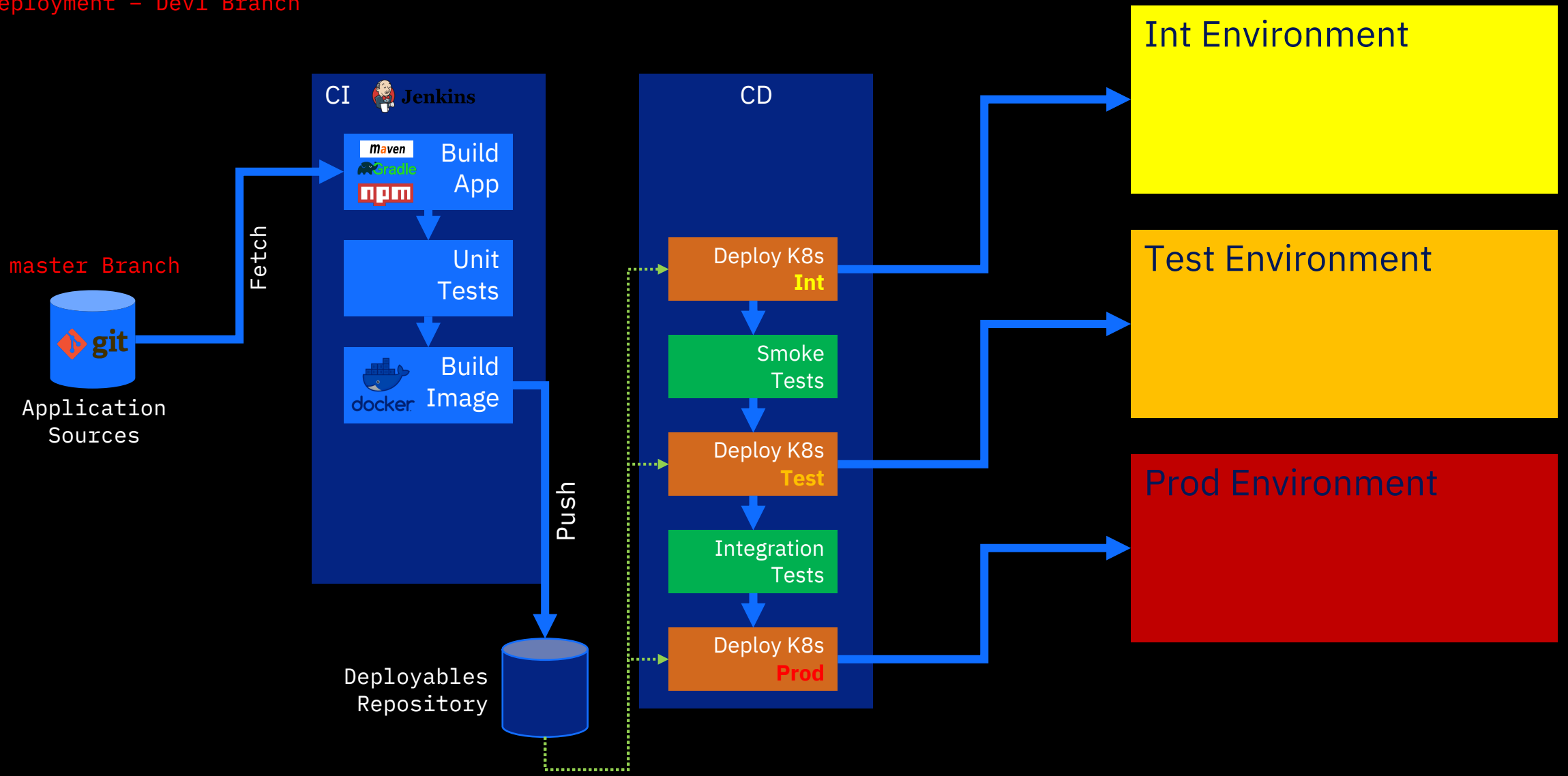
DevOps – CI/CD Pipeline

Merge – Dev1 Branch to master Branch



DevOps – CI/CD Pipeline

Deployment – Dev1 Branch



DevOps – **Git**Ops

GitOps builds on DevOps with Git as a single source of truth for declarative infrastructure and applications - the whole system.

1. Having a completely automated delivery pipeline that can roll out changes to your infrastructure when changes are made to Git.
2. Operating a fast paced business 24/7 requires monitoring and observability baked into the beginning. Security is of critical importance.
3. Everything has to be version controlled and stored in a single source of truth from which you can recover.

DevOps – GitOps

Full Audit Trail

GitOps is using git, which is an excellent database to persist the changes made to the system.

Everything as a code

In the cloud, both CI and CD should be defined as code first. The desired state of environments should also be set in full as a declarative code.

...

Source: Dmitri Lerko - A Year With Gitops in Production and Arge - 5 GitOps Best Practices

DevSecOps

Putting Security in your DevOps transformation

Today, application security is mostly an afterthought, that is run in pre-production and often perceived as a roadblock to staying ahead of the competition.

Examples

- Getting access to a Git repository (IAM)
- Open firewall ports
- Getting credentials for an API
- Getting certificates

DevSecOps

Putting Security in your DevOps transformation

DevSecOps means thinking about application and infrastructure security from the start.

1. The idea is to get security back in to the lifecycle → shifting security left
2. The goal is to make security as silent and seamless as possible
3. «Everyone is responsible for security» mindset

- Developers write secure code
- SREs secure the environments

DevSecOps

Putting Security in your DevOps transformation

But we still need clear responsibilities!

Most organizations have clear governance of risk, and out of that we derive security policies.

In order to bring Sec into DevOps is to bake those policies into the processes.

DevSecOps

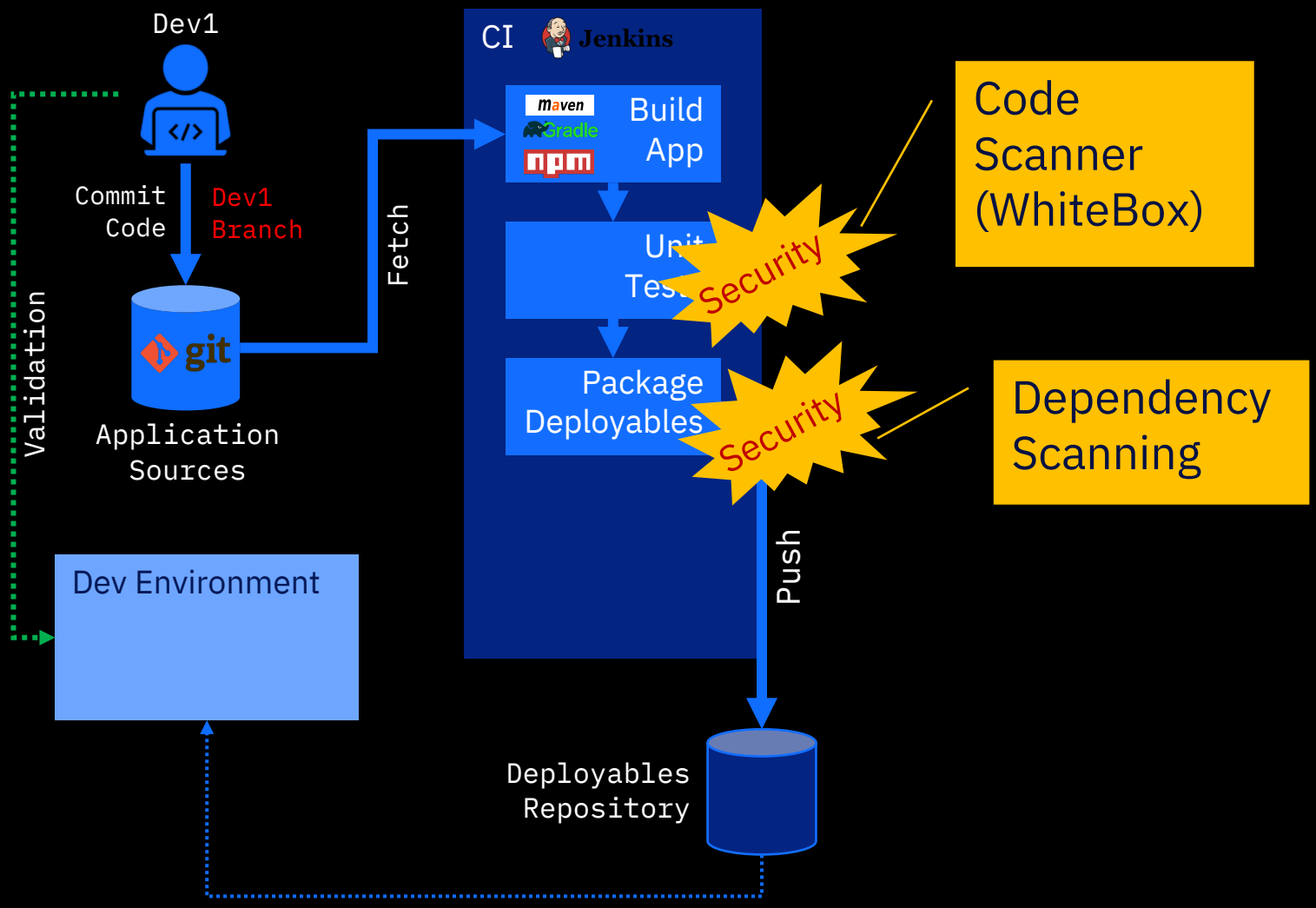
Putting Security in your DevOps transformation

But we still need clear responsibilities!

For example, when you are writing code,
do the developers have a tool that checks for vulnerabilities during the local build process at build time
or do you do this within your CI process using Jenkins?
or do you do this within your CD process before deployment?

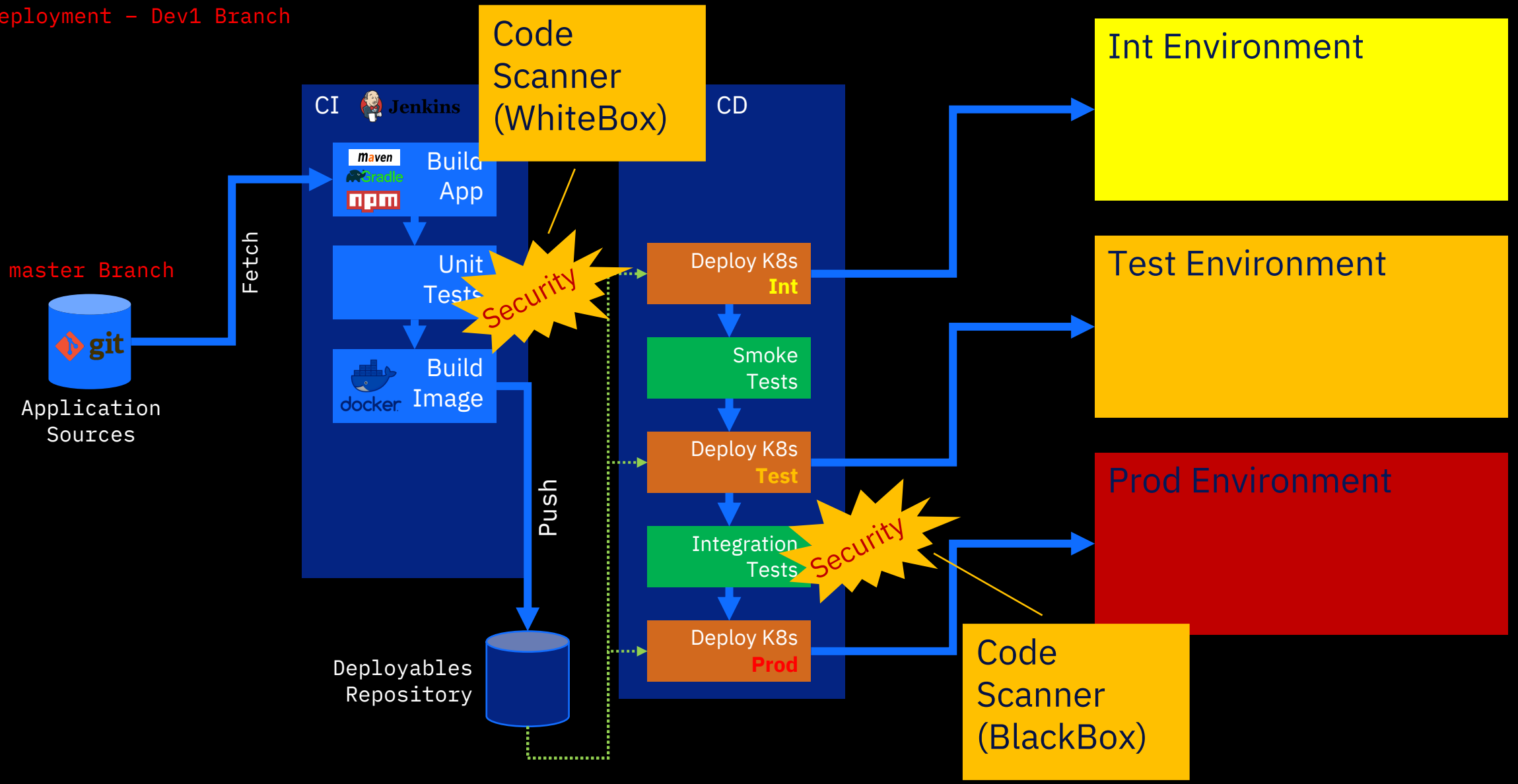
DevSecOps – CI/CD Pipeline

Development – Dev1 Branch



DevSecOps – CI/CD Pipeline

Deployment – Dev1 Branch



IBM

QUESTIONS?

IBM **Cloud**

IBM

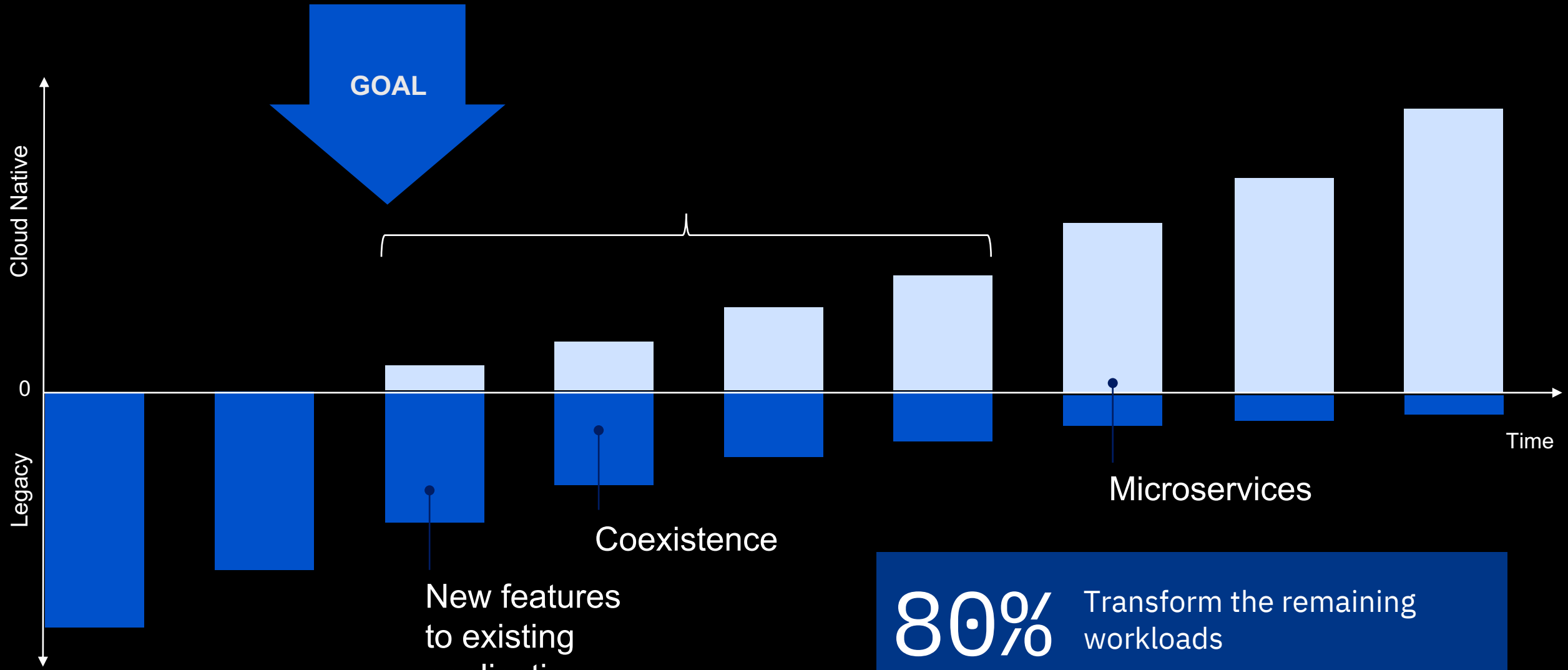
The Journey to Cloud
Microservices

02

IBM **Cloud**

Digital Transformation

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



80% Transform the remaining workloads

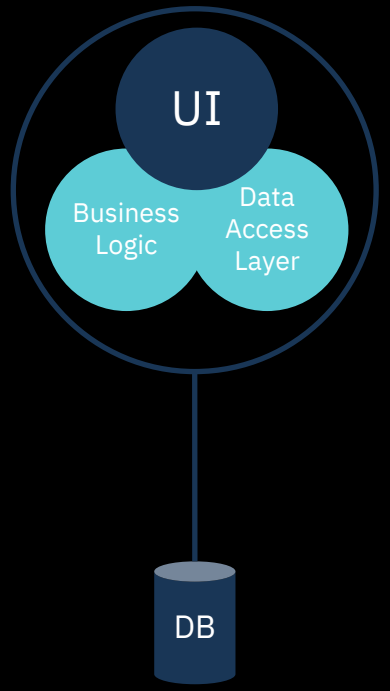
Microservices

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

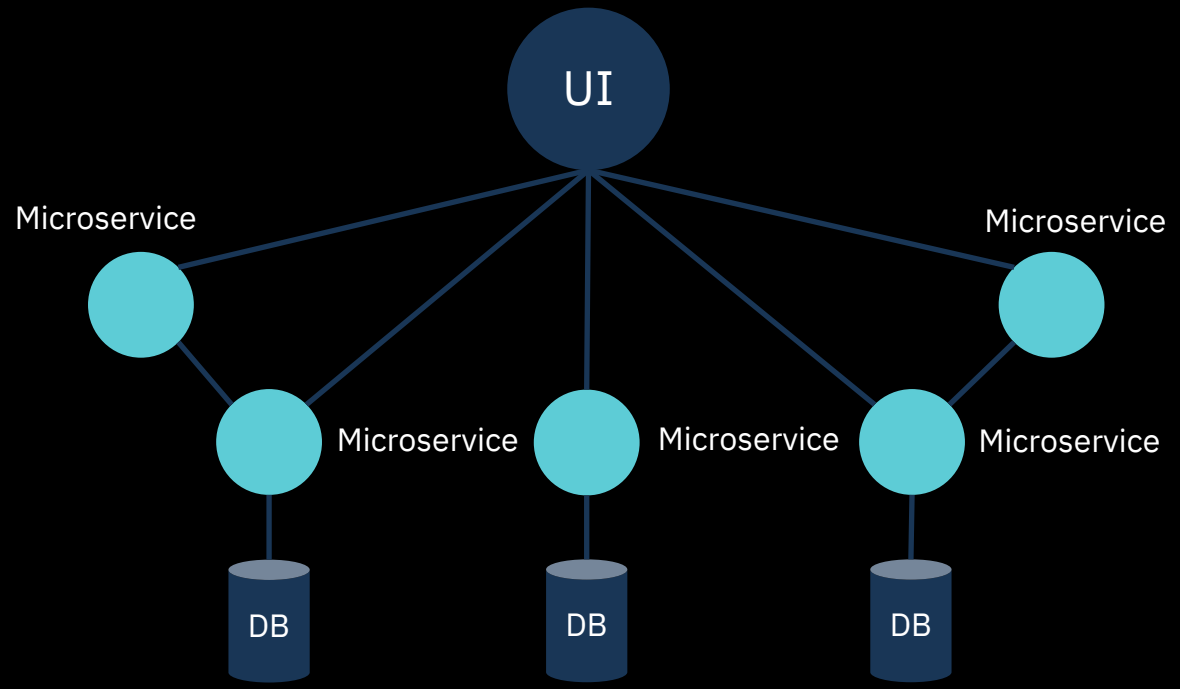
Accelerate delivery by minimizing communication and coordination between people

Microservices architecture

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



Monolithic Architecture

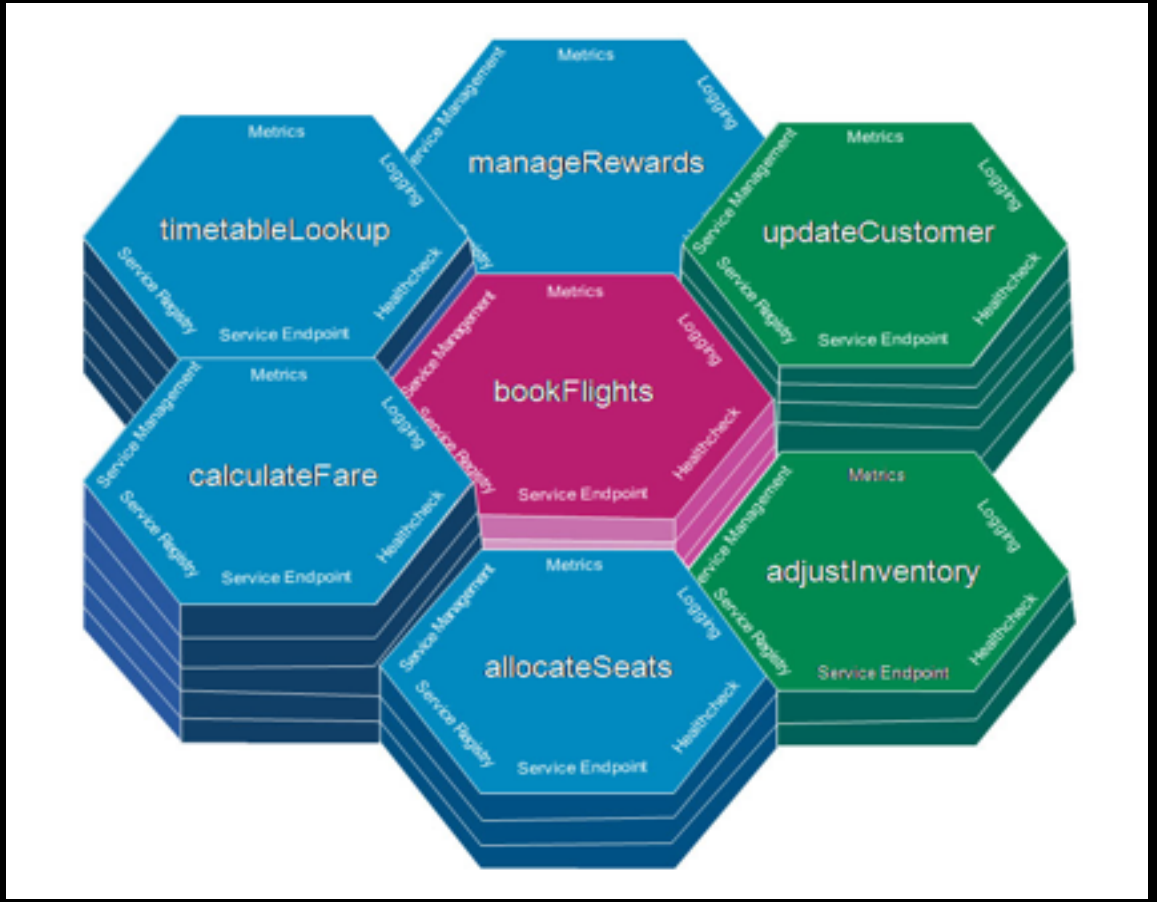


Microservices Architecture
A microservice is a granular, decoupled component within a broader application

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

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

Obvious 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 assume any specific infrastructure dependency
5. Don’t use infrastructure APIs from within your application
6. Don’t rely on OS-specific features
7. Don’t manually install your application

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

That said....

Microservices are hard

..but we're getting ahead of ourselves

IBM

QUESTIONS?

IBM **Cloud**

IBM

The Journey to Cloud
Docker

03

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

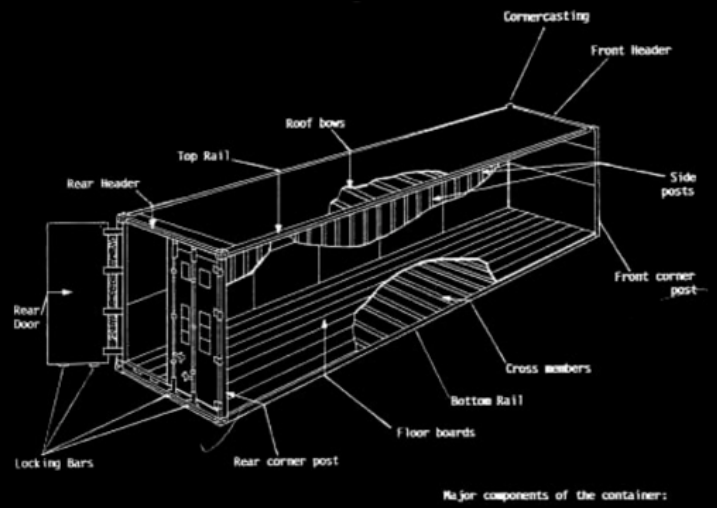
Containers work by **isolating the differences between applications** inside the container so that everything outside the container can be **standardized**

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

1979

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

2007

cgroups

2008

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

2011

warden

2012

Docker

2014

rkt

2016

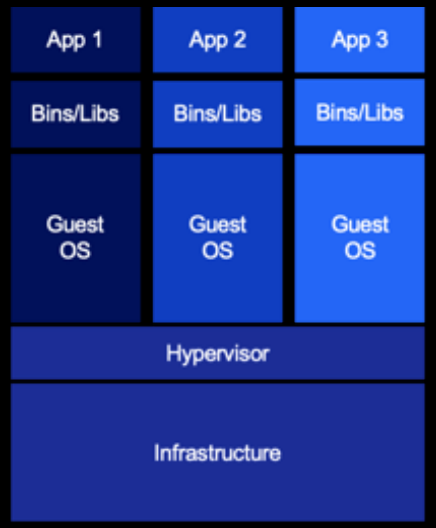
Garden
runC

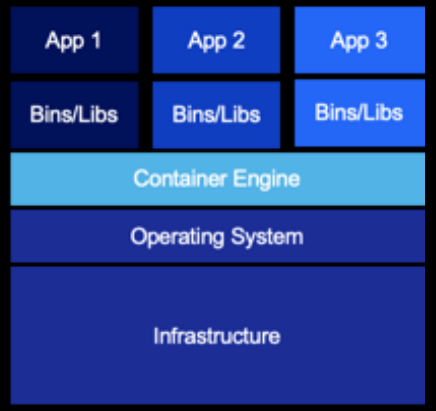
2018

containerd

https://containerd.io/

VMs vs. Containers

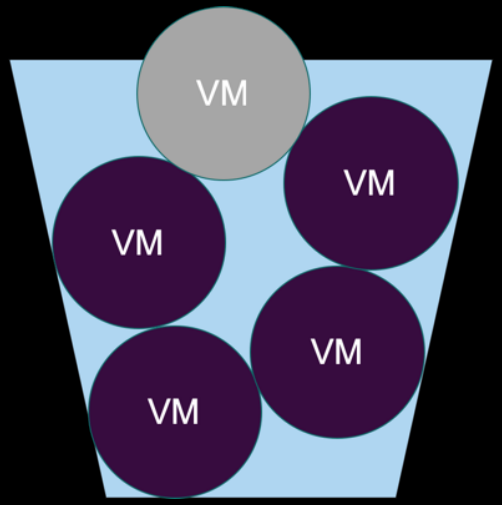


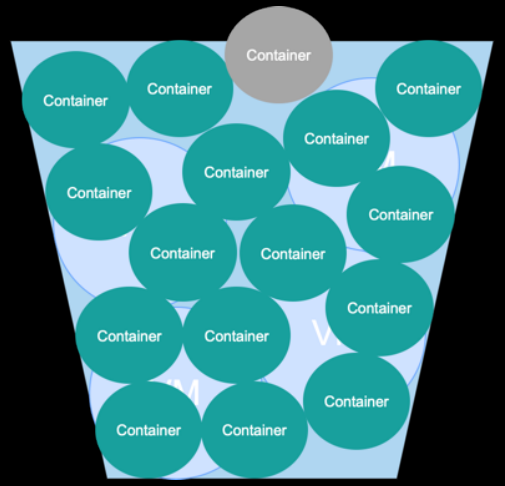


+ VM Isolation
- Complete OS
- Static Compute
- Static Memory

+ 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

Advantages of Containers

Containers are **portable**

Containers are **easy to manage**

Containers provide **“just enough” isolation**

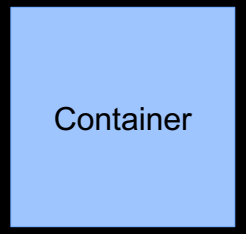
Containers use hardware **more efficiently**

Containers are **immutable**

Docker Components

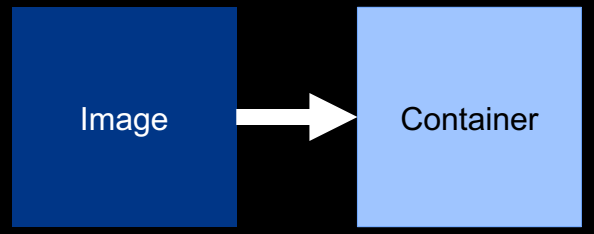
Container

Smallest compute unit



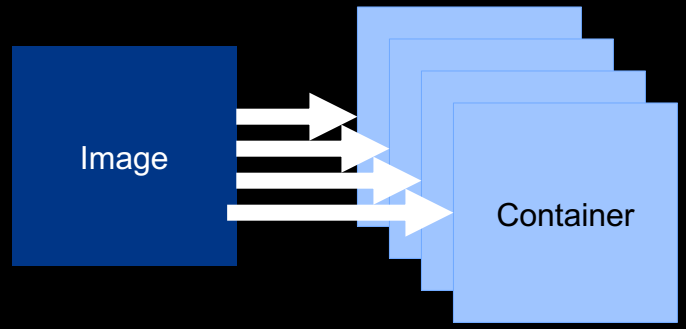
Docker Components

Containers
are created from
Images



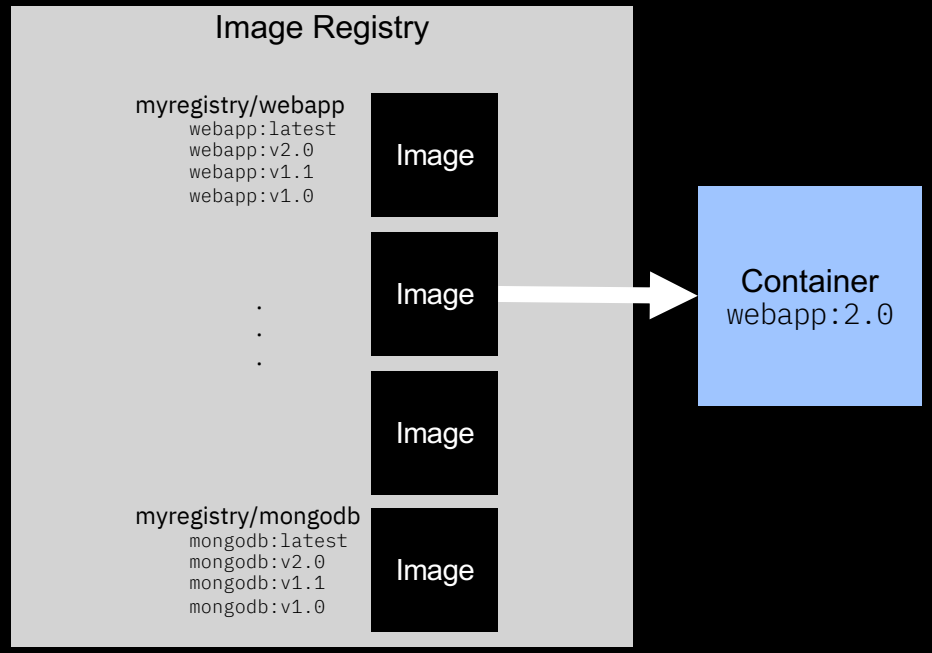
Docker Components

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

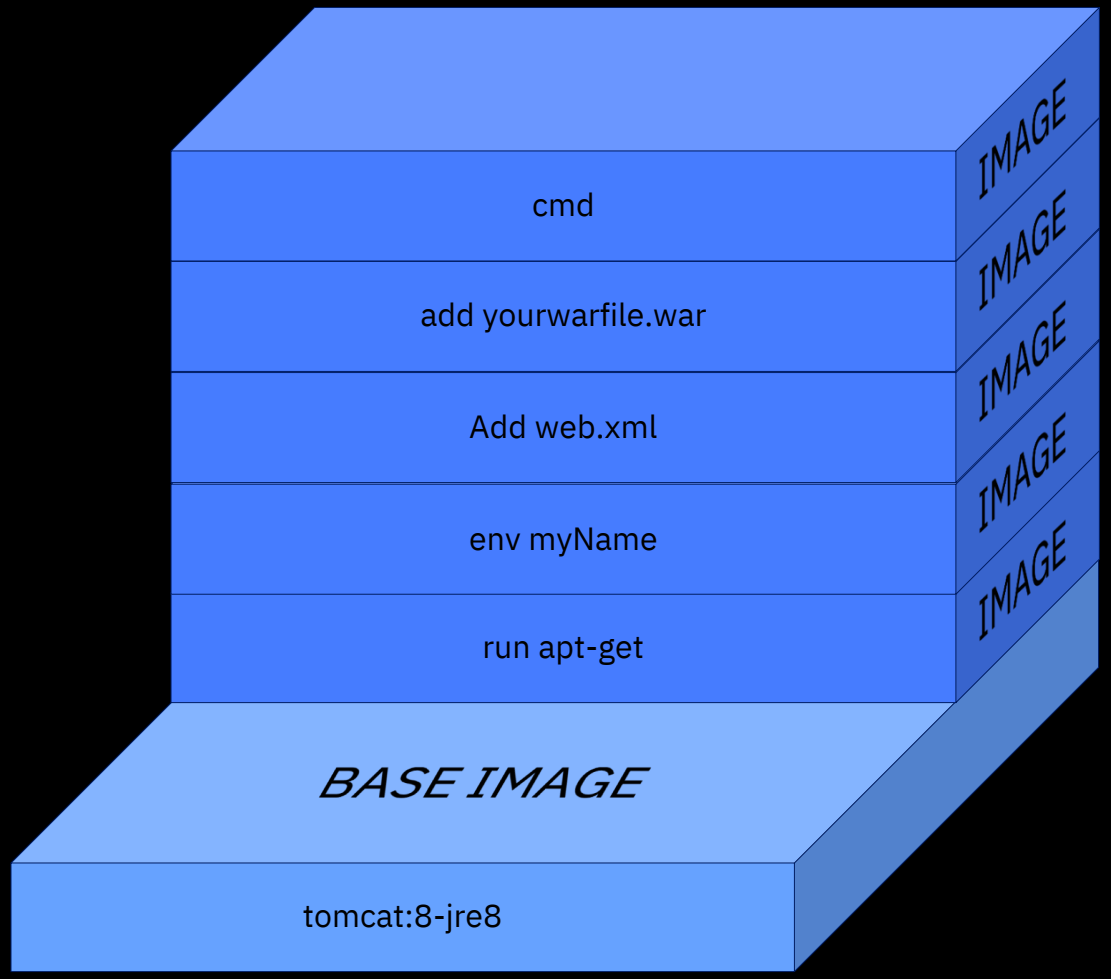
Why Containers

Docker Layers

Inheritance

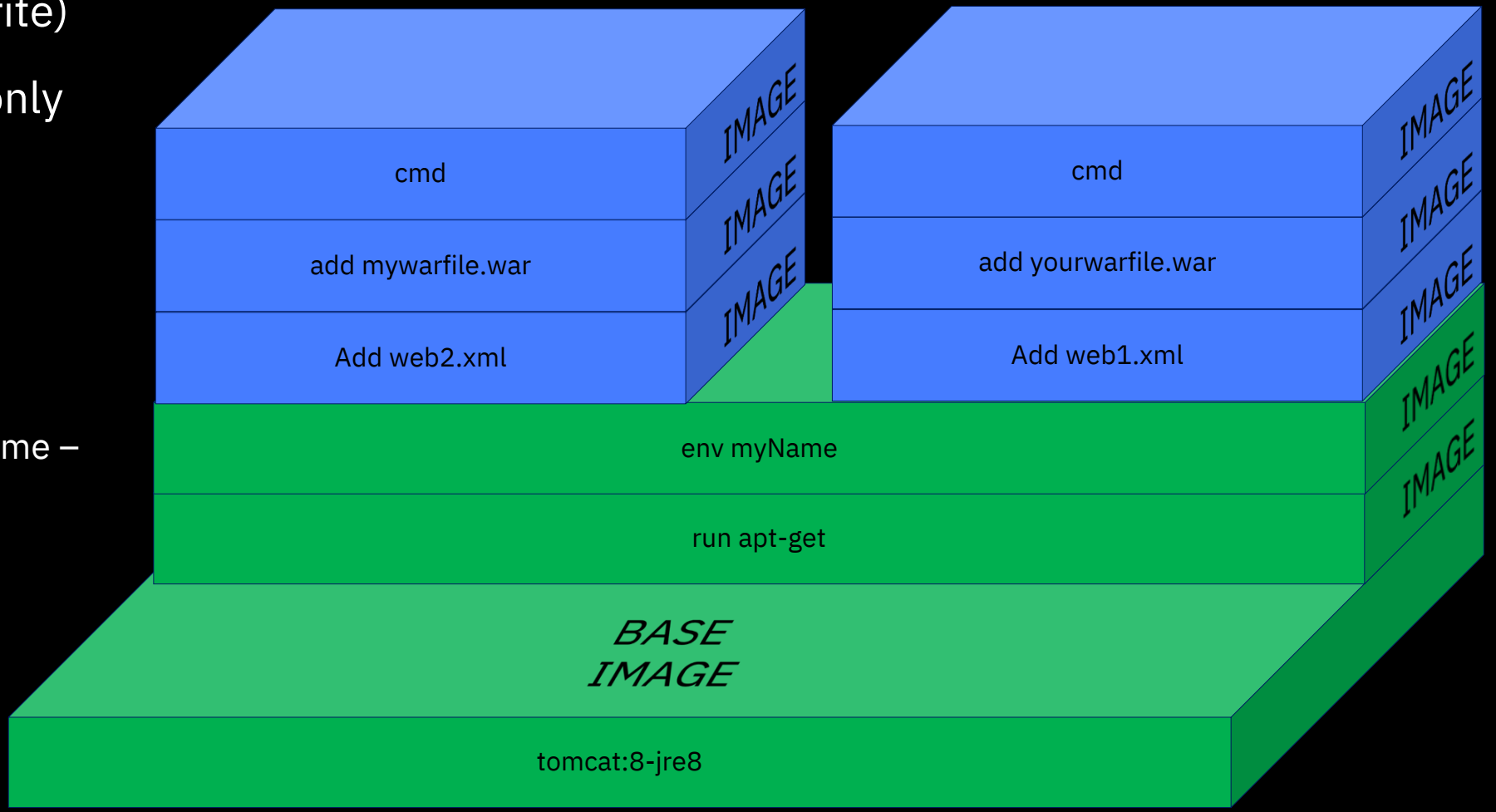
- Build on the work of those who came before you

```
# 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"]
```

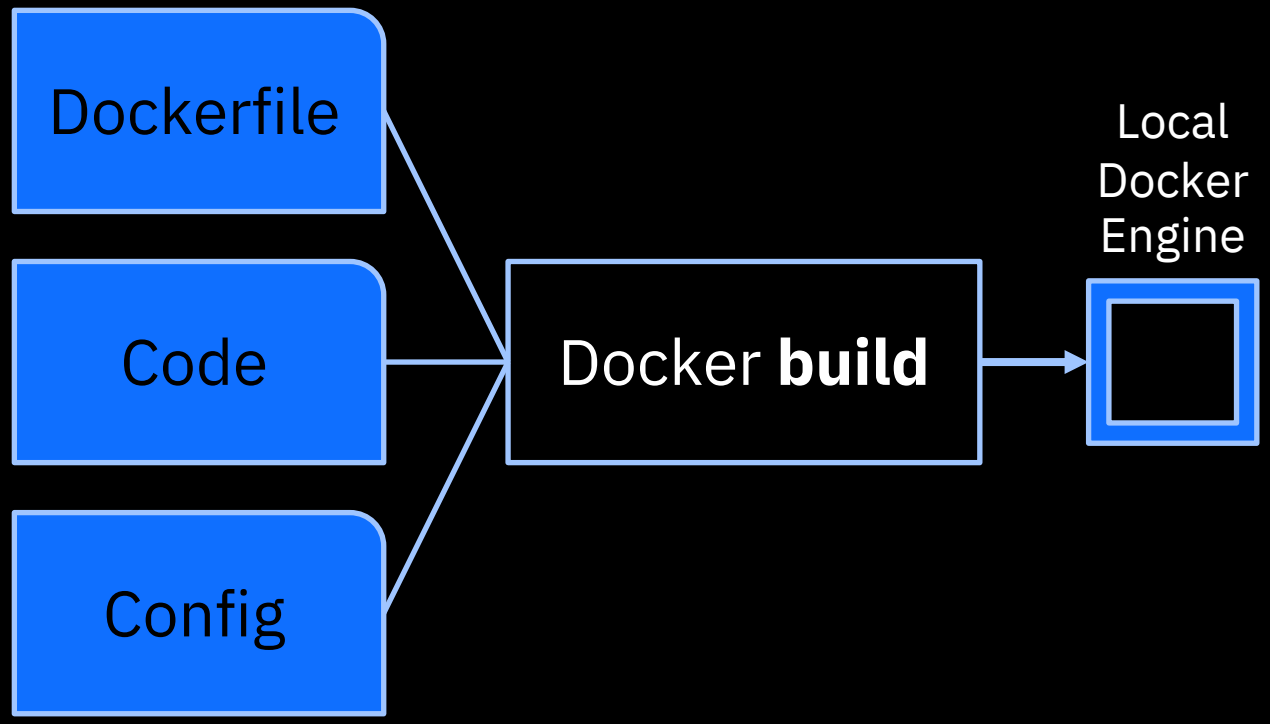


Why Containers

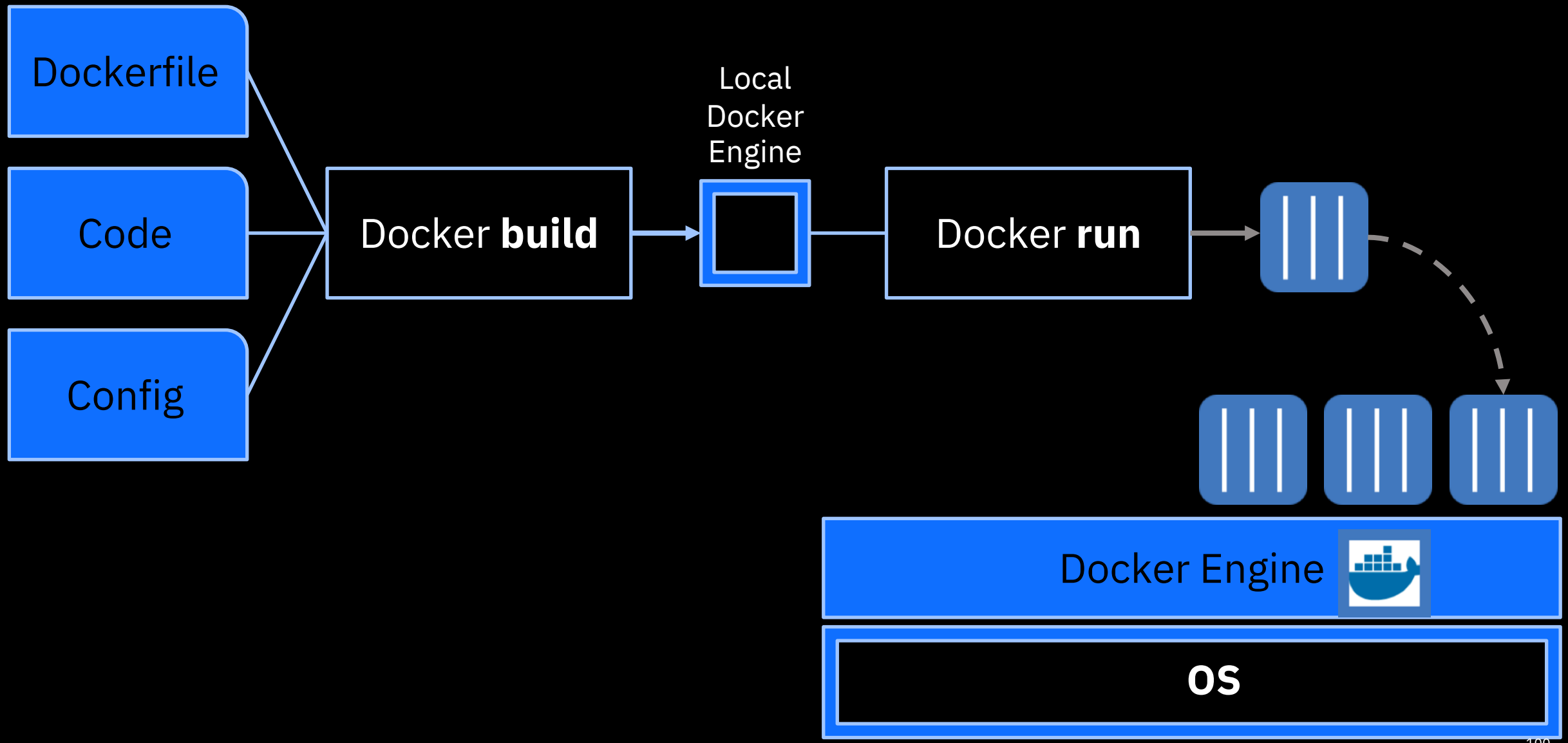
- Docker uses a **layered filesystem** (copy-on-write)
- New files (& edits) are only visible to current/above layers
- Layers allow for reuse
  - More containers per host
  - Faster start-up/download time – base layers are "cached"



Docker Basics – Build



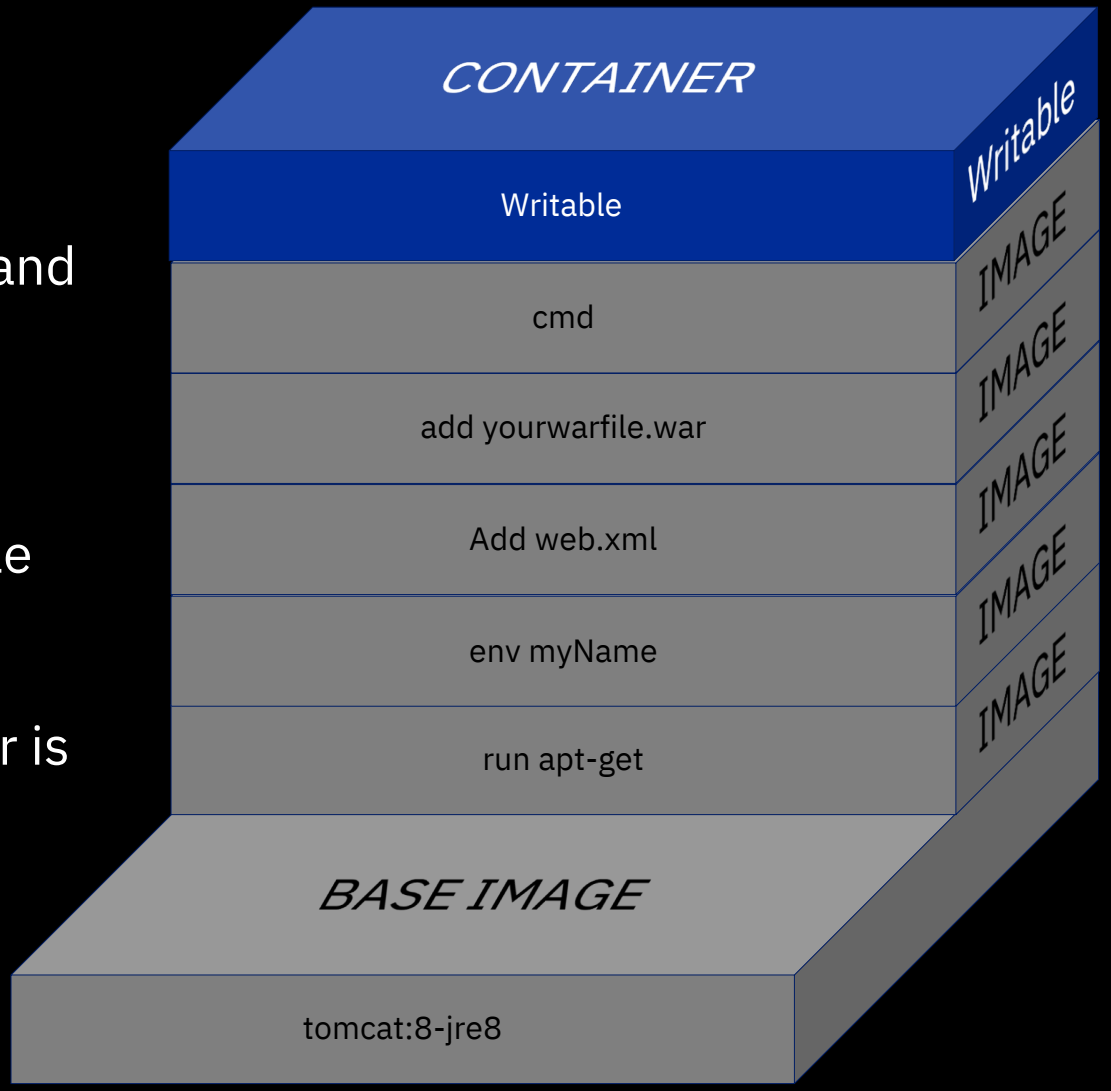
Docker Basics – Run



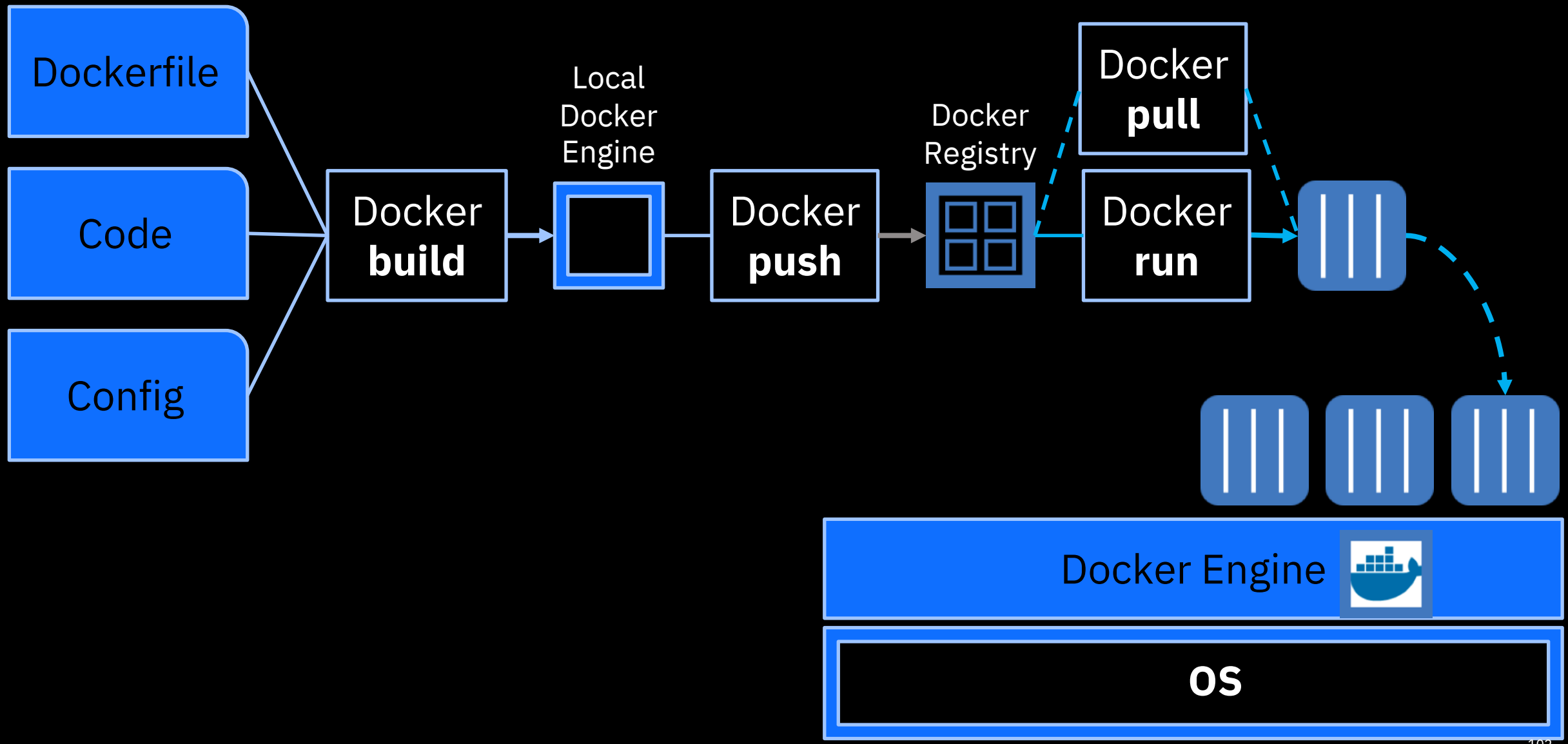
Docker Basics – Run

Docker Layers

- 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 – 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

04

IBM **Cloud**

Everybody Loves Containers

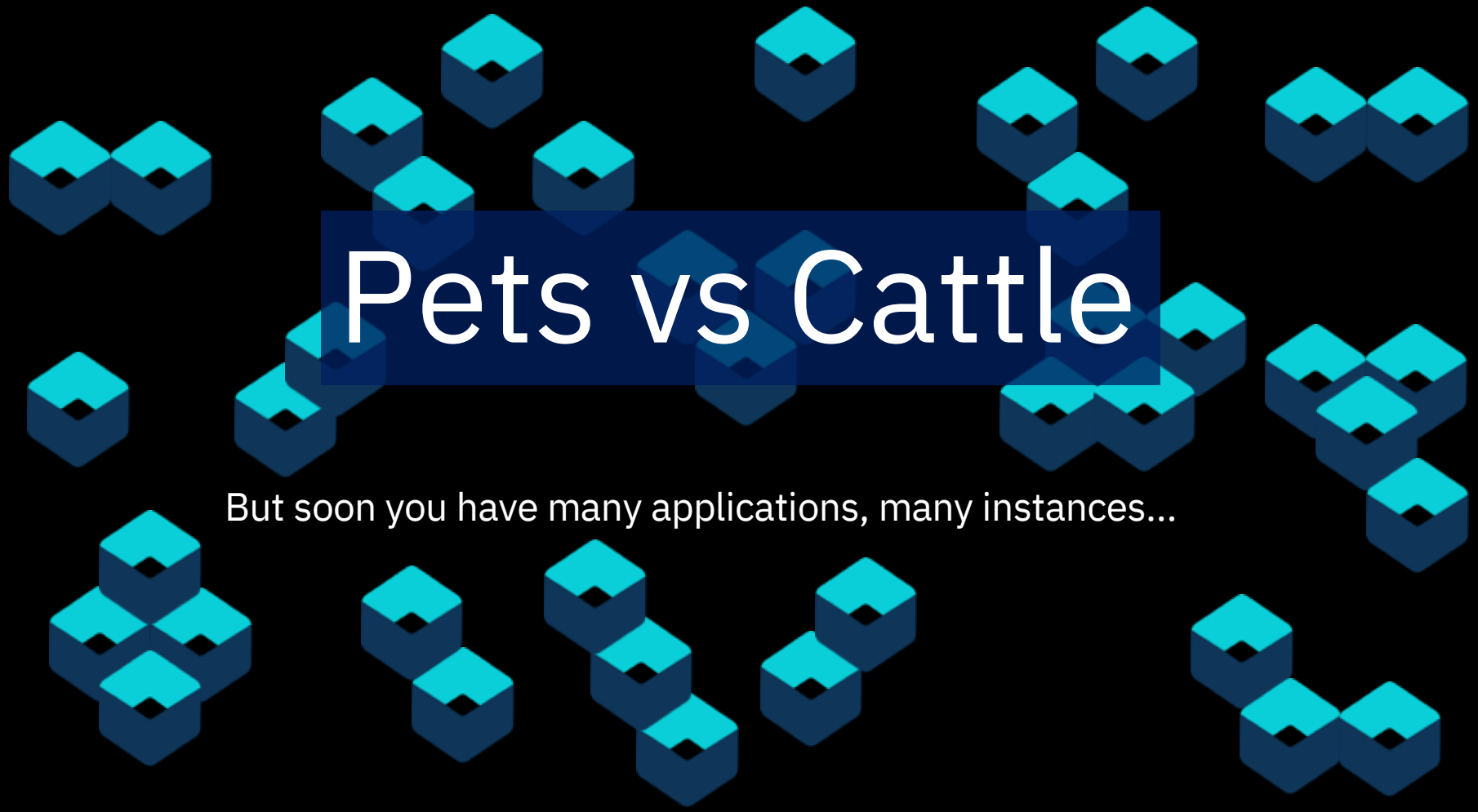
But when you go

From this....

To this....

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

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



Pets vs Cattle

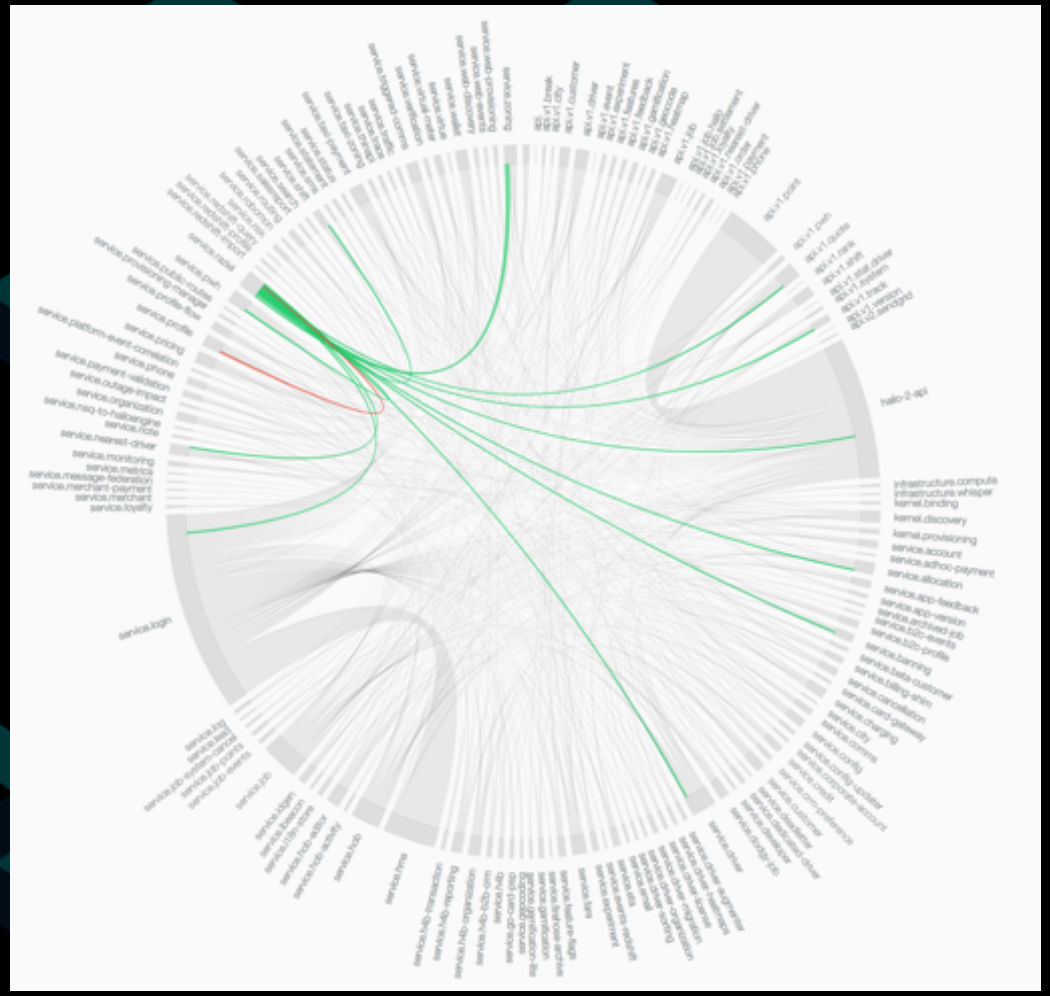
But soon you have many applications, many instances...

The trade off

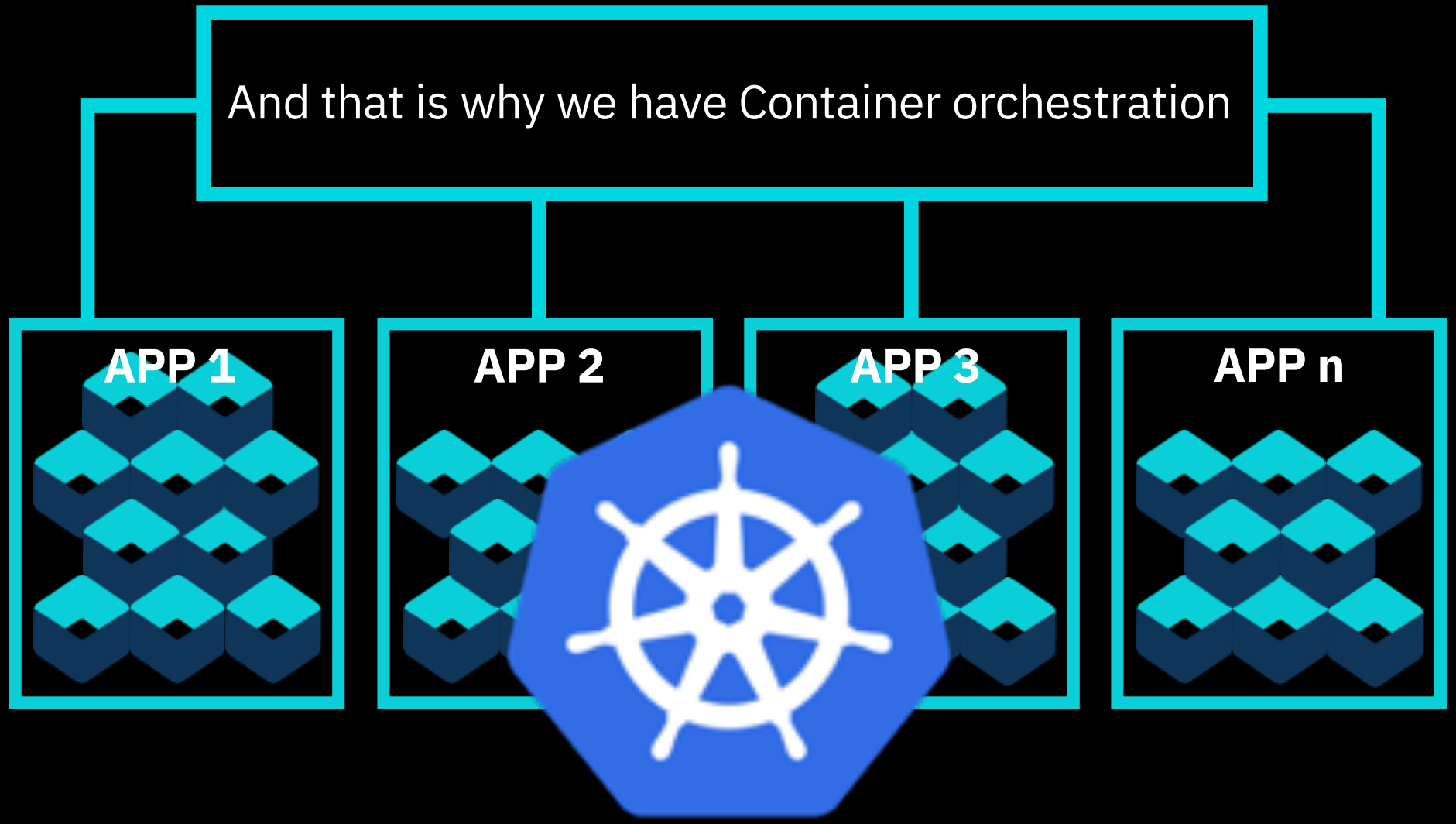
Improved delivery velocity

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.

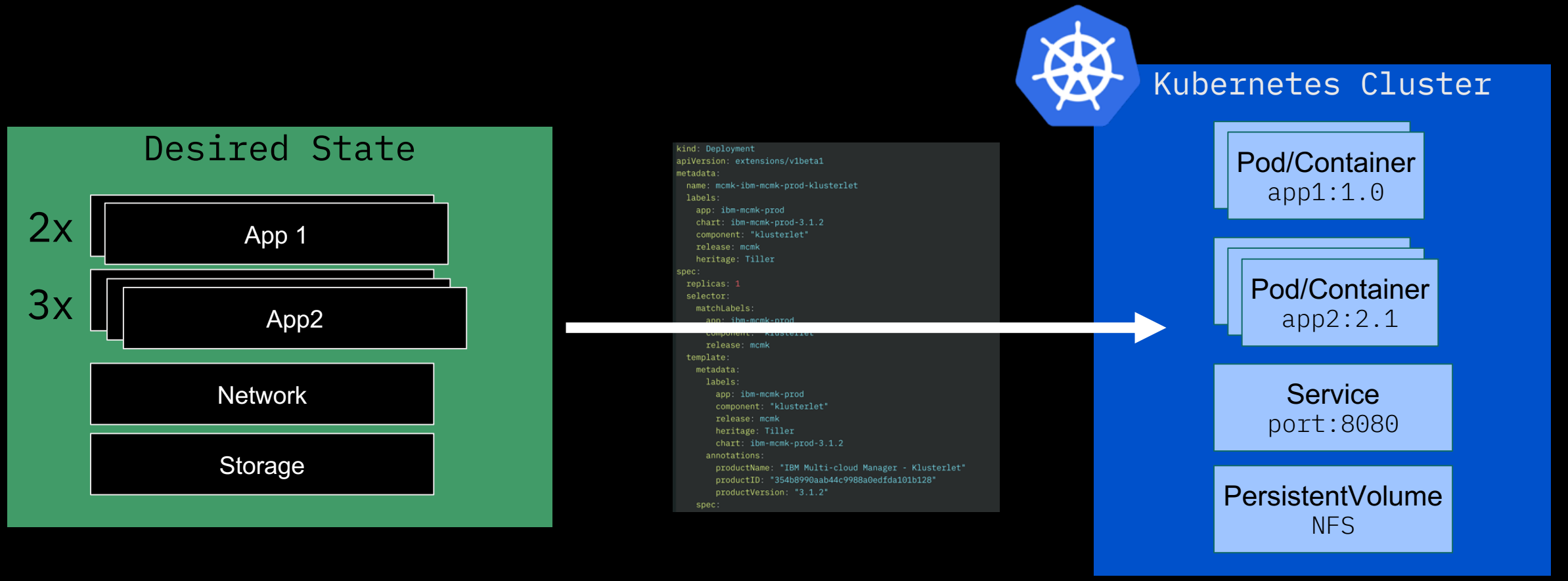
Declarative Systems

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.

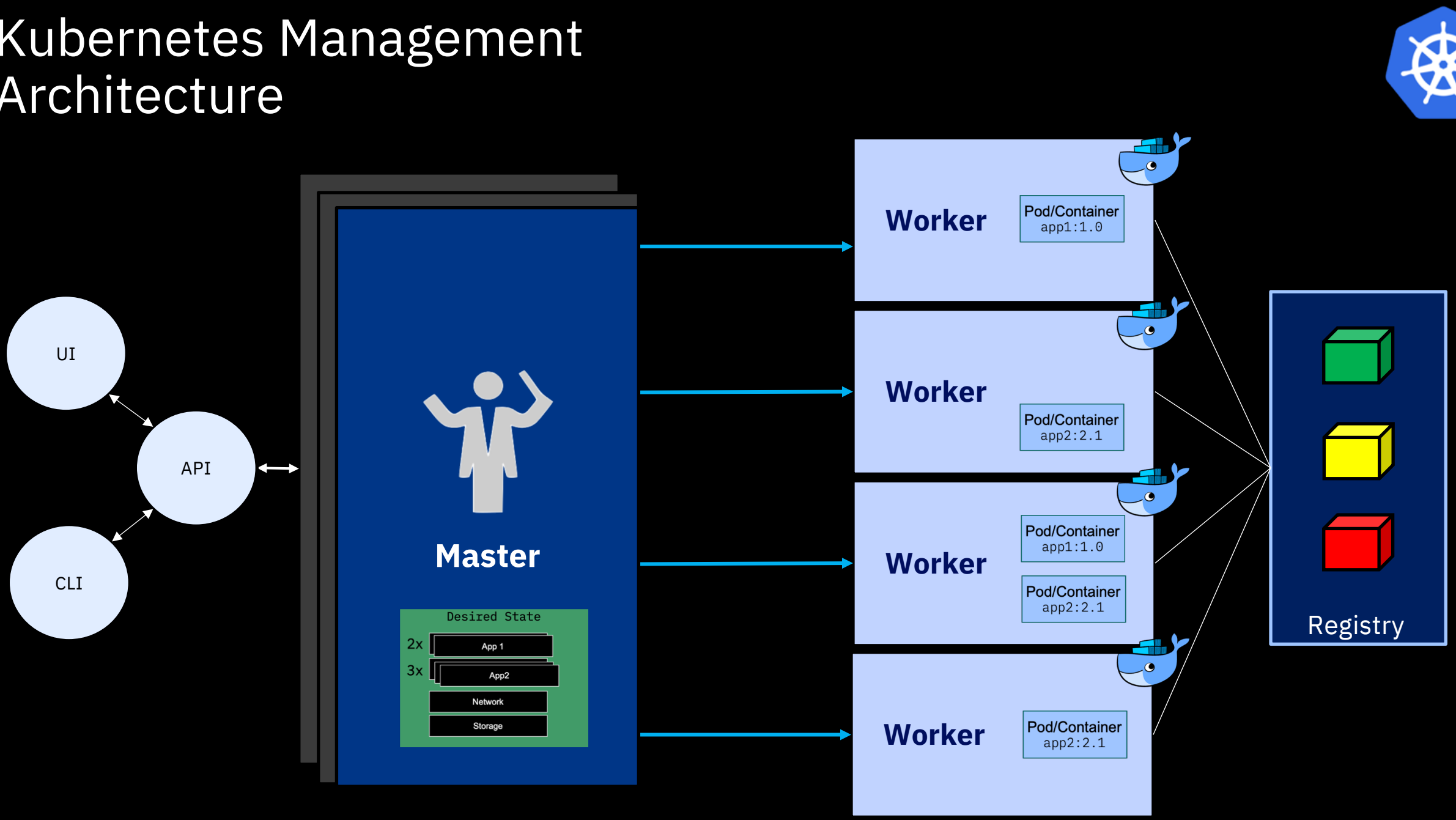
Kubernetes – Declarative System

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

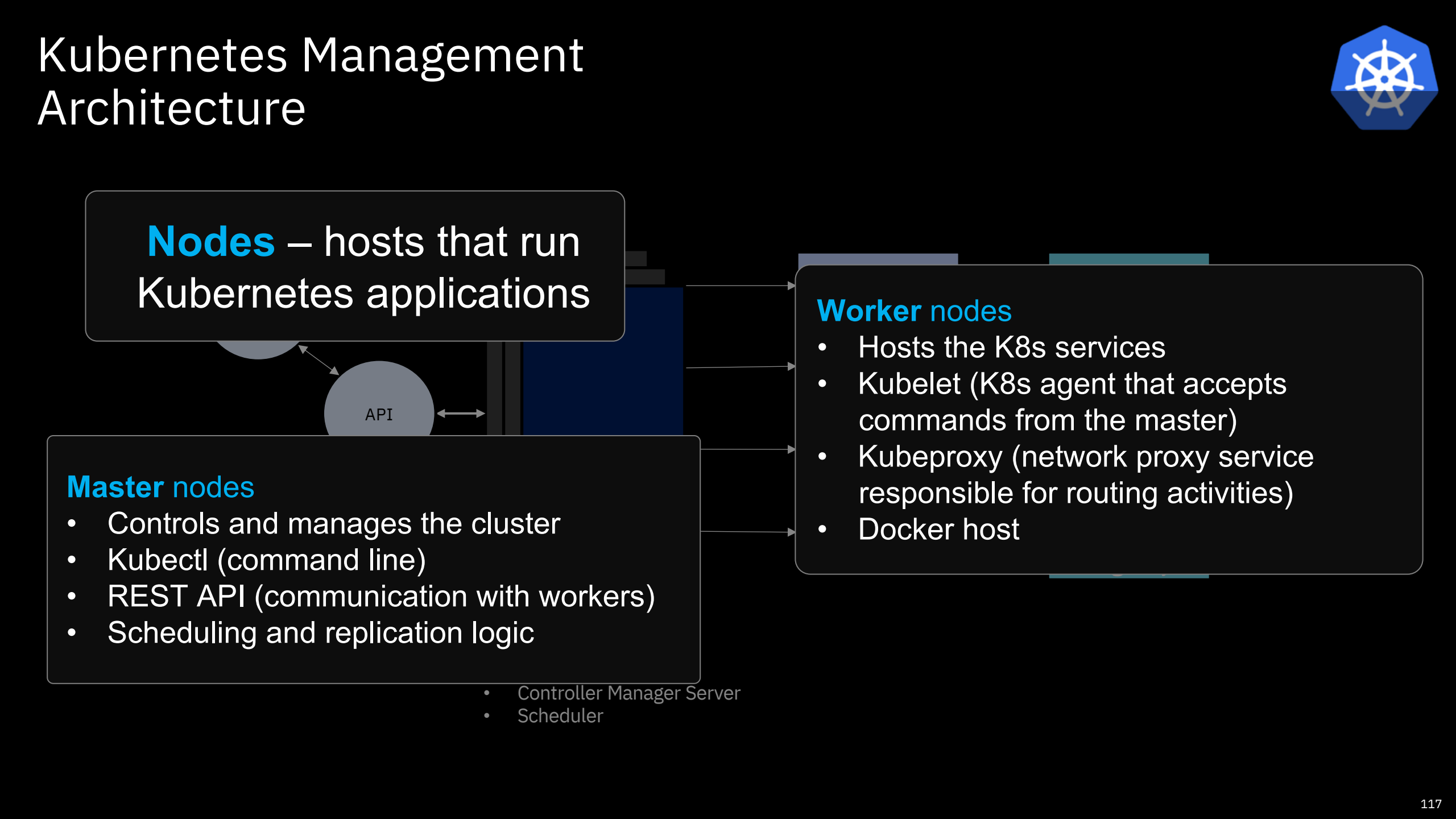
Designed for extensibility

- 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 Management Architecture



Nodes – hosts that run Kubernetes applications

API

Master nodes

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

- Controller Manager Server
- Scheduler

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

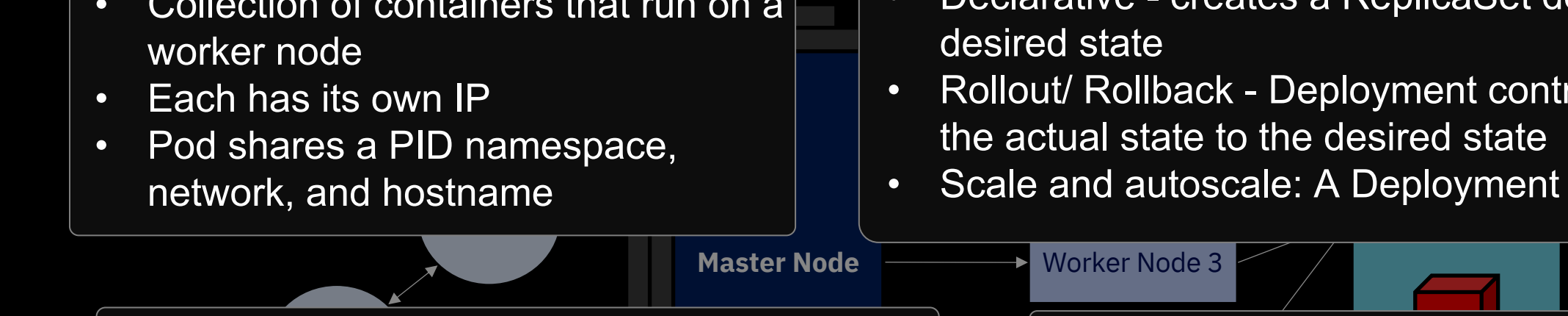
Kubernetes Management Architecture

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

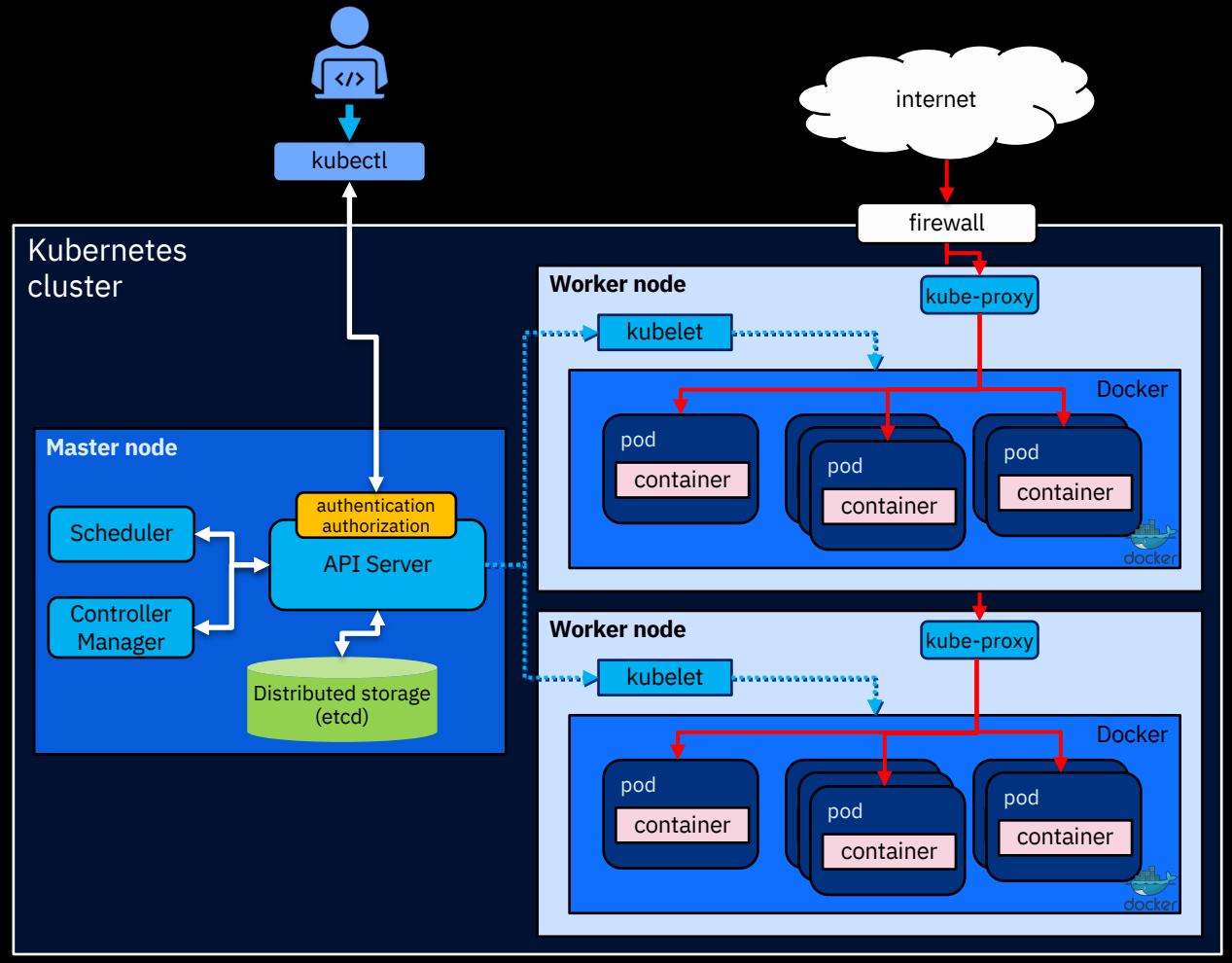
Kubernetes Cluster Architecture

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
- kube-proxy – routes inbound or ingress traffic



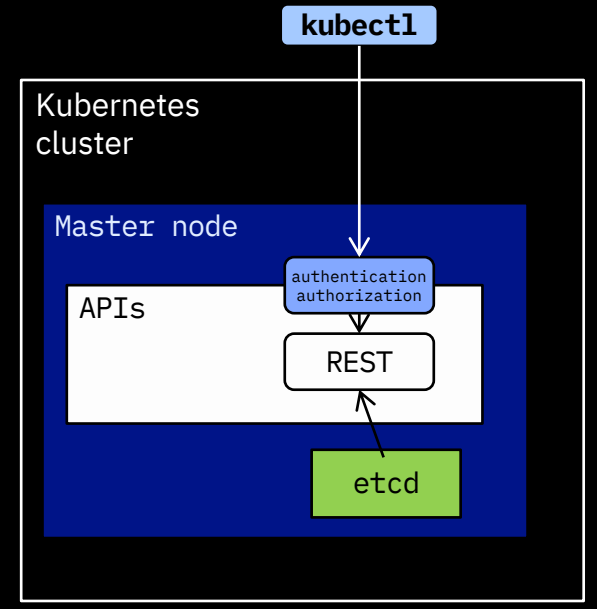
kubectl – talking to the Cluster

`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.



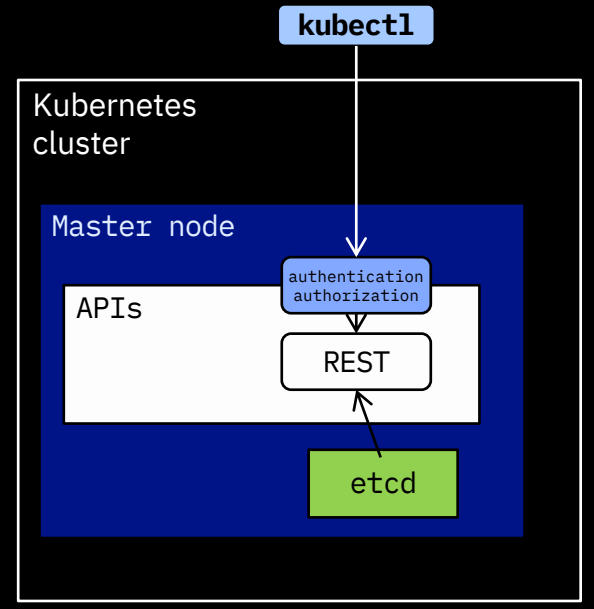
kubectl – talking to the Cluster

`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]

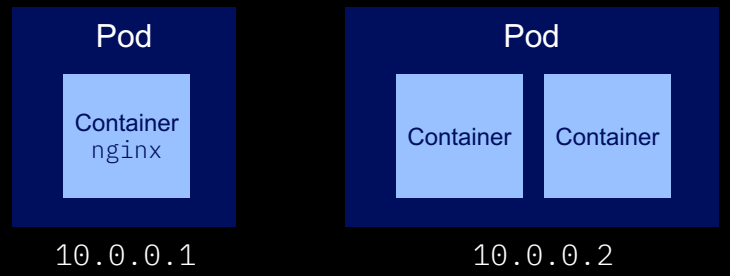
kubectl create -f example.yaml	Create objects in yaml file
kubectl apply -f example.yaml	Modify objects in yaml file
kubectl delete -f example.yaml	Delete objects in yaml file
kubectl get pods	List Pods in Namespace
kubectl describe nodes <node-name>	Details about K8s object
kubectl logs <pod-name>	Get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



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.
- **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
```



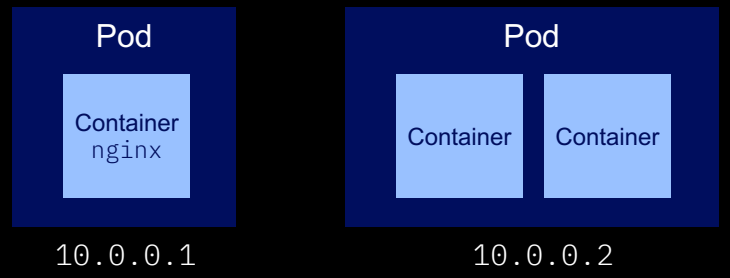
Creating a Pod

```
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
```

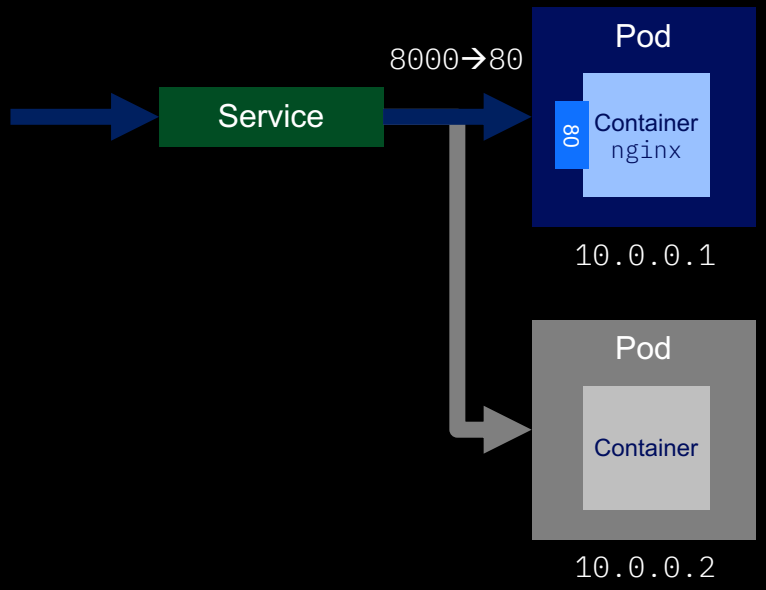
example.yaml

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



Service

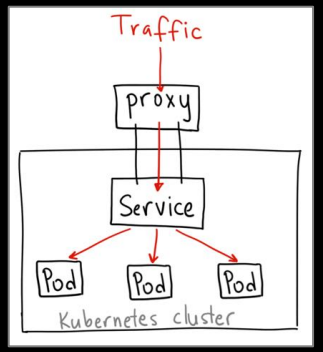
- A service provides a way to refer to a set of Pods (selected by labels) with a single static IP address.
- Also provide load balancing



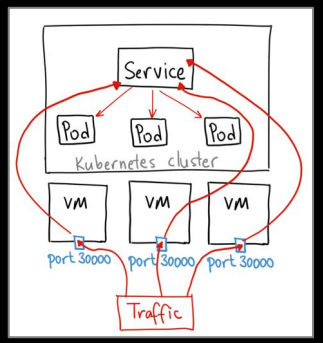

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
  selector:
    app: nginx
```

Service

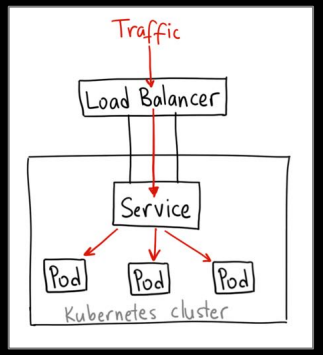
ClusterIP: This type exposes the service on the cluster internal IP. This means that the service is only reachable from within the cluster.



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



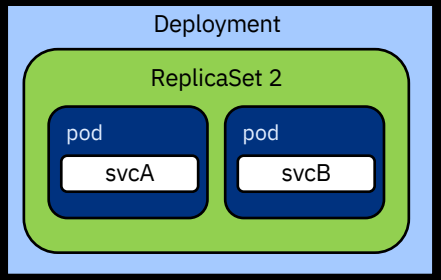
LoadBalancer: This service type exposes a service using the cloud provider load balancer.



Deployments & ReplicaSets

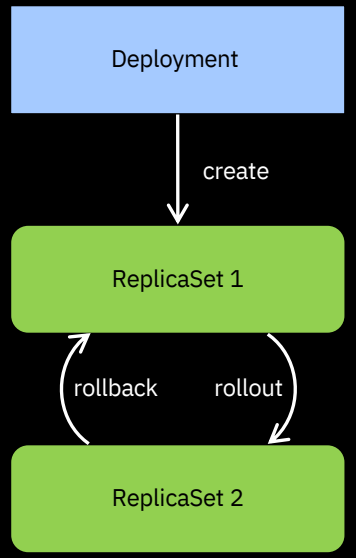
Deployment

- 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
```

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

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

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

05

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 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 – breaking down the ~~monolith~~ beast

How to identify candidates

- Is there anything that is scaling 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/

Microservices – Database refactoring

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

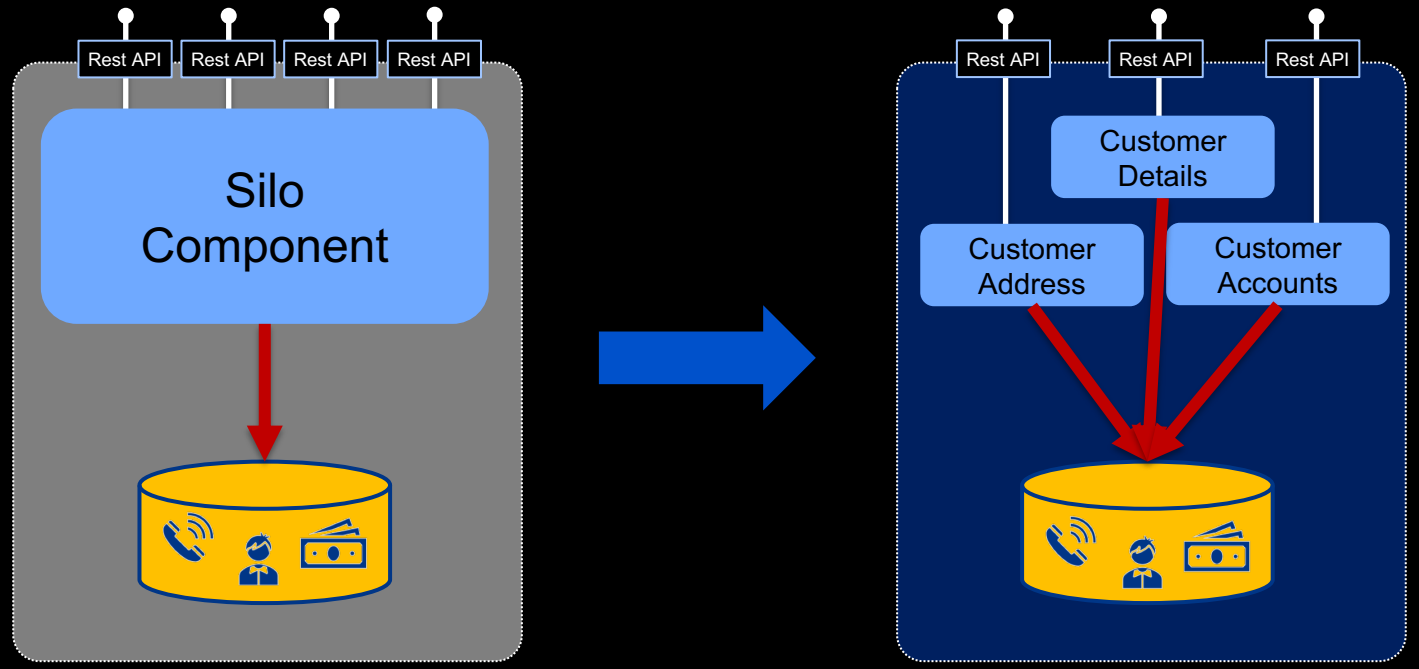
- What is the domain? What is reality?
- Where are the transactional boundaries?
- How should microservices communicate across boundaries?
- What if we just turn the database inside out?

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

Microservices – 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.

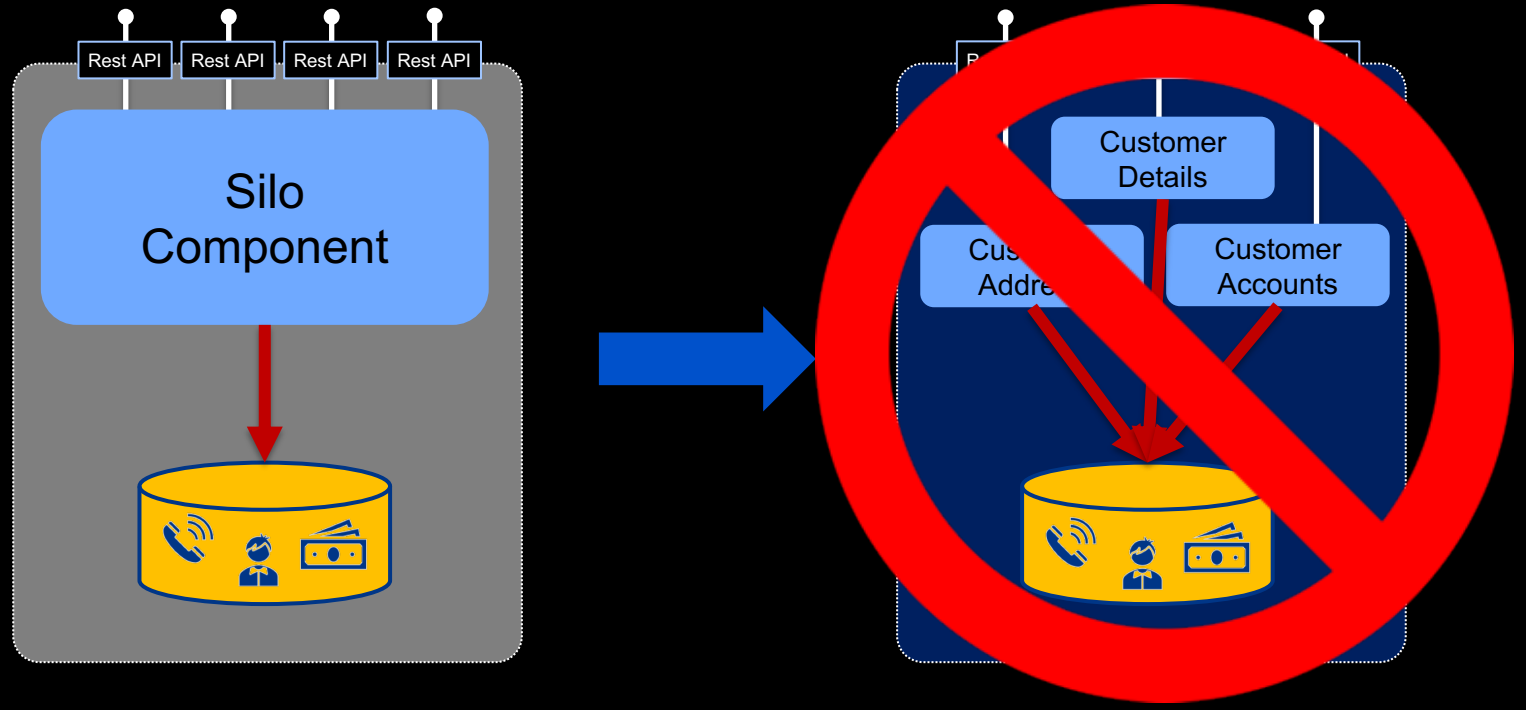


Source: IBM Architecture Center

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.

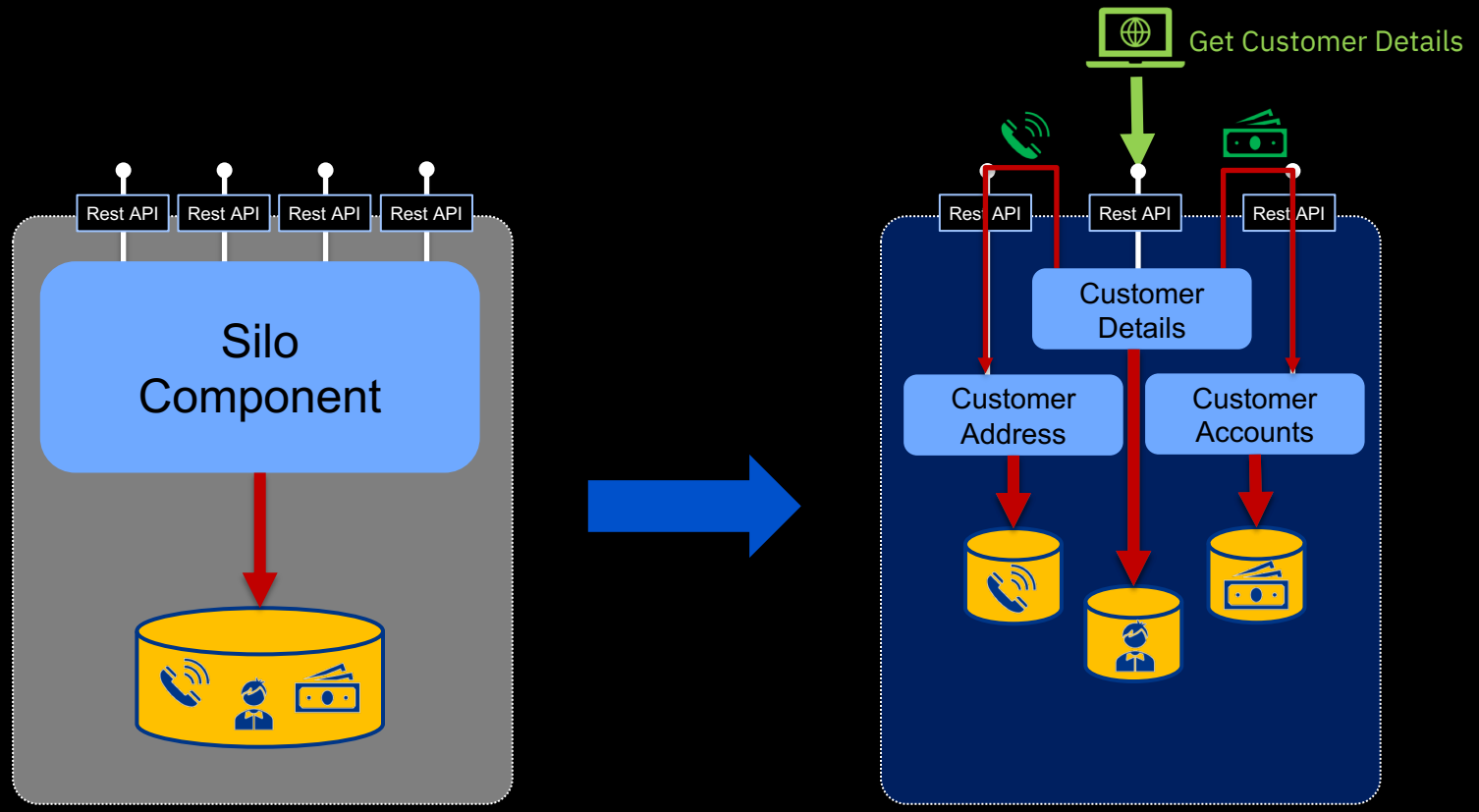


Source: IBM Architecture Center

Microservices – Database refactoring

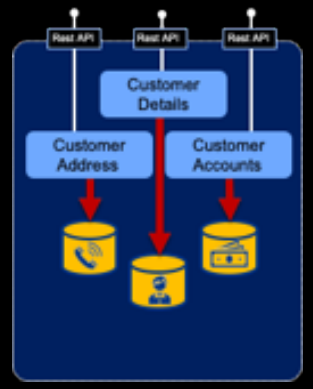
Data Isolation

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

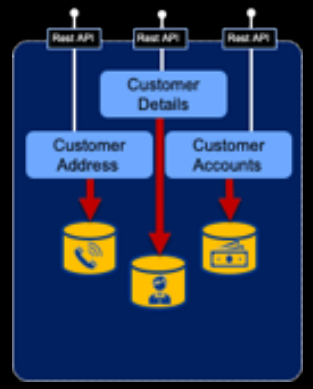


Benefits

- 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)

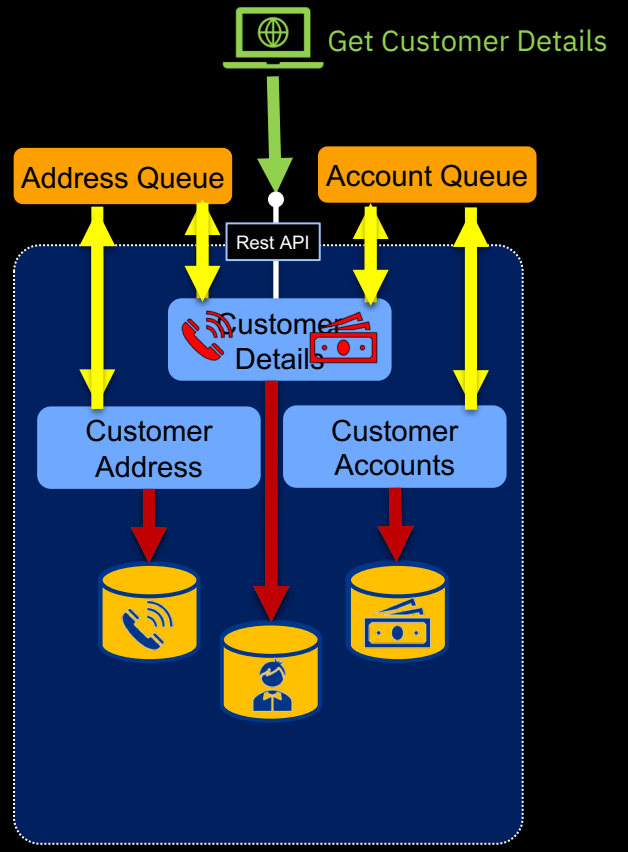
Source: https://blog.christianposta.com/microservices/the-hardest-part-about-microservices-data/

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)
- Not all applications are large enough to break down into microservices
- Applications that require tight integration between individual components and services (real-time processing, ...)
- A moderately large, moderately complex application being maintained by a relatively small development and operations team
- At what point is it in its lifecycle? Mission-critical?

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
- Many of your microservices share a datastore
- 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
- Your microservices share a lot of the same code or models

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

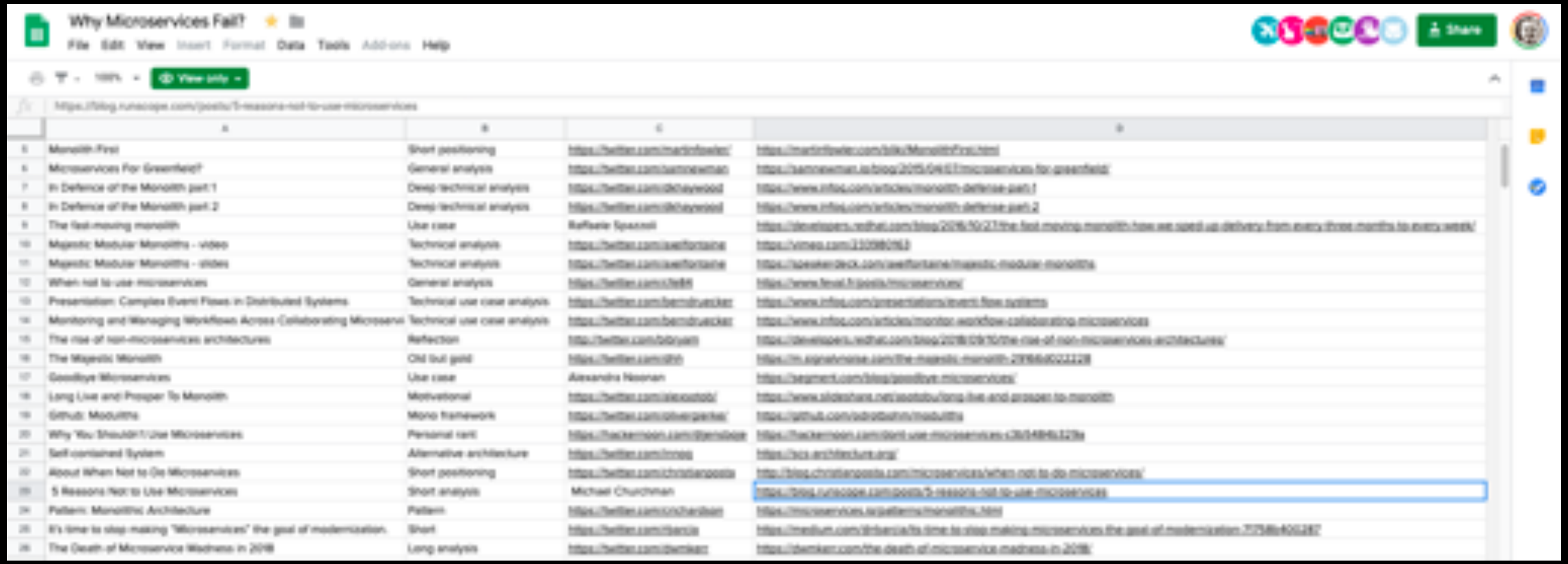
Microservices – reasons not to use them

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

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

IBM

QUESTIONS?

IBM **Cloud**

IBM

The Journey to Cloud
Kubernetes Security

06

IBM **Cloud**

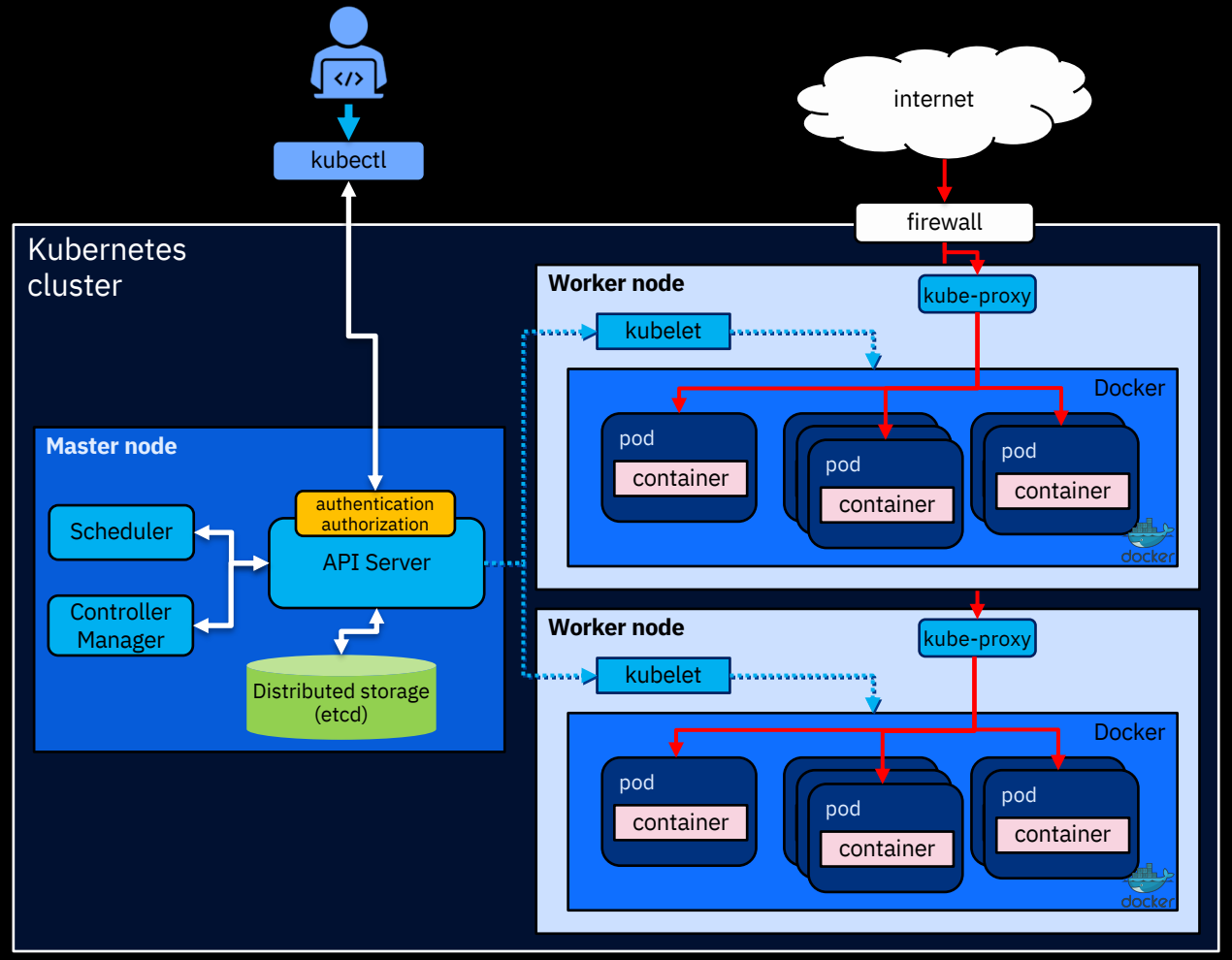
Kubernetes – **Security – Back to basics**

Master node

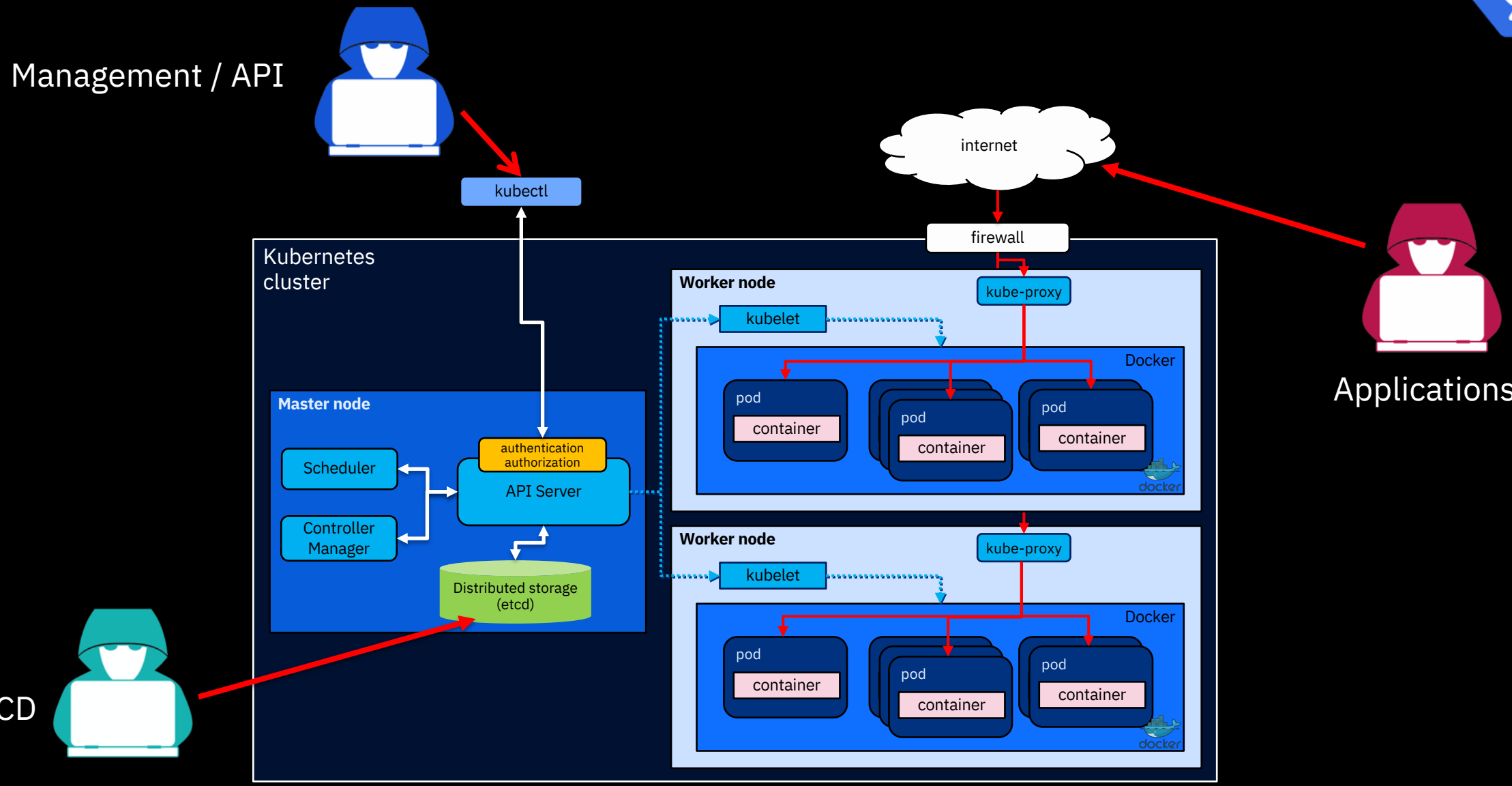
- Node that manages the cluster
- Scheduling, replication & control
- Multiple nodes for HA

Worker nodes

- Node where pods are run
- Docker engine
- kubelet agent accepts & executes commands from the master to manage pods
- kube-proxy – routes inbound or ingress traffic



Kubernetes – **Security – Attack surface**

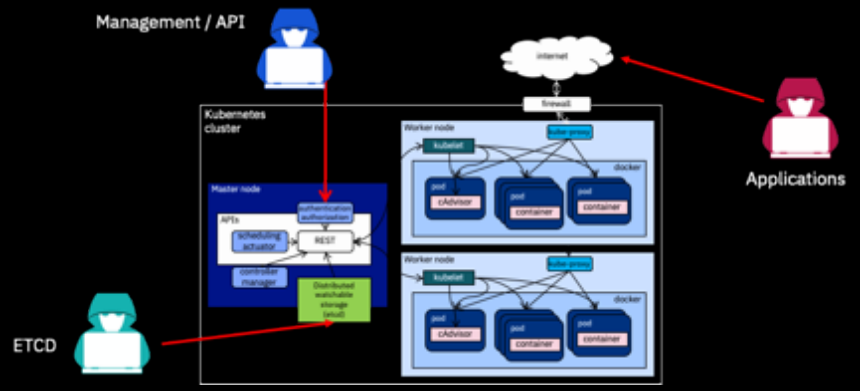


Kubernetes – **Security Basic Topics**

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

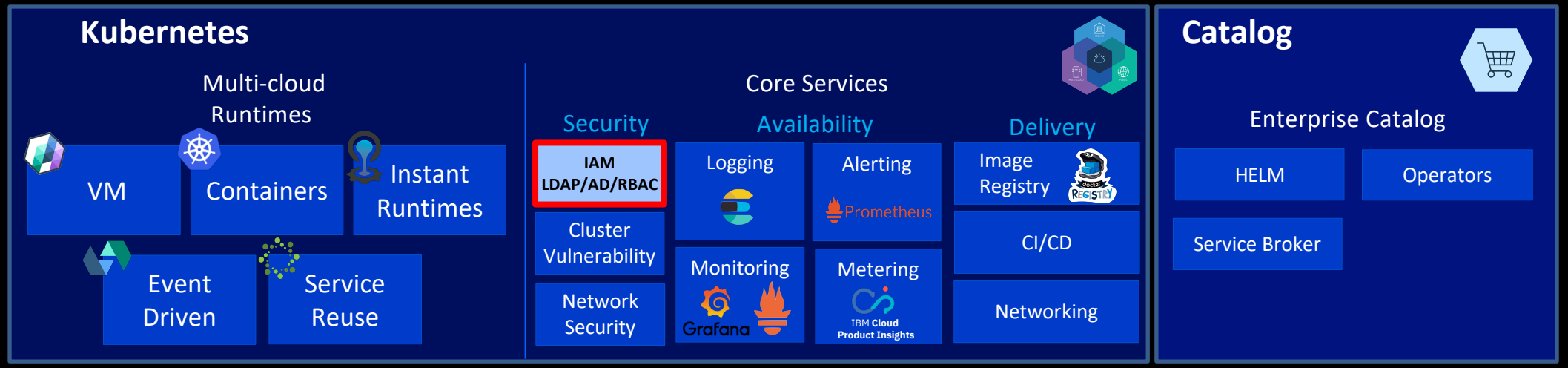
Management/API

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
- Use Kubernetes secrets

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)...)

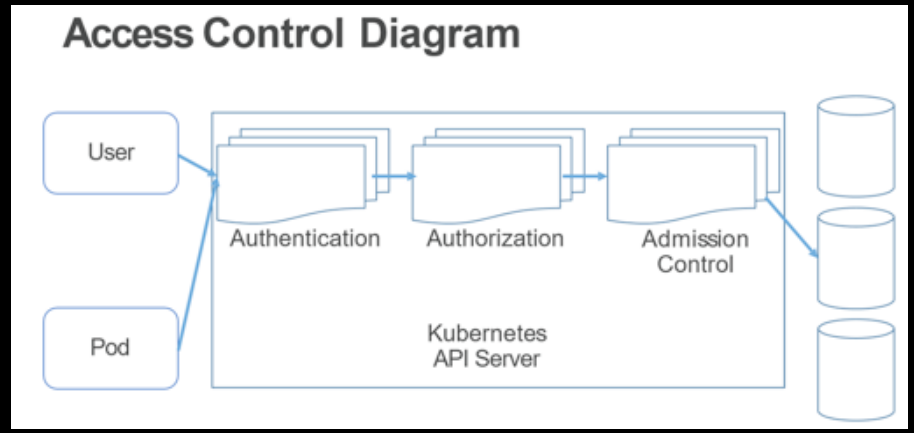
Developer — create, view, get permission.
Tester — create, delete, update, get permission.
Administrator — All permission

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



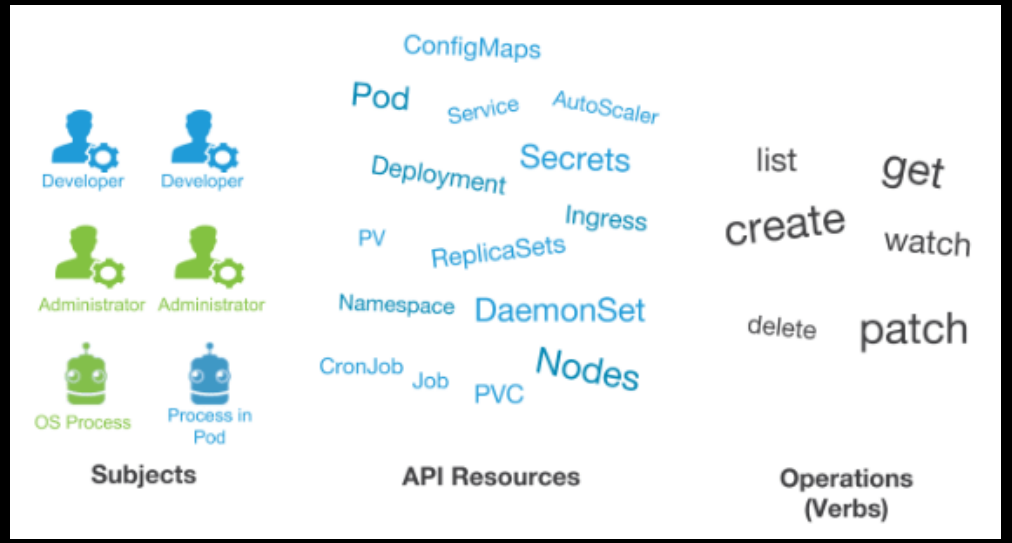
Source: https://kubernetes.io/docs/reference/access-authn-authz/controlling-access/

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/

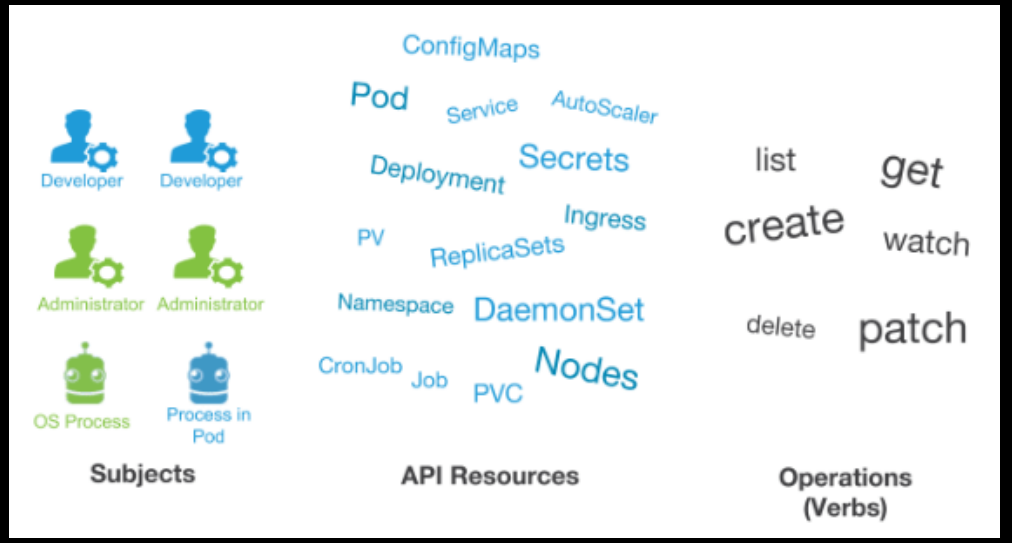
RBAC Basic Elements

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.

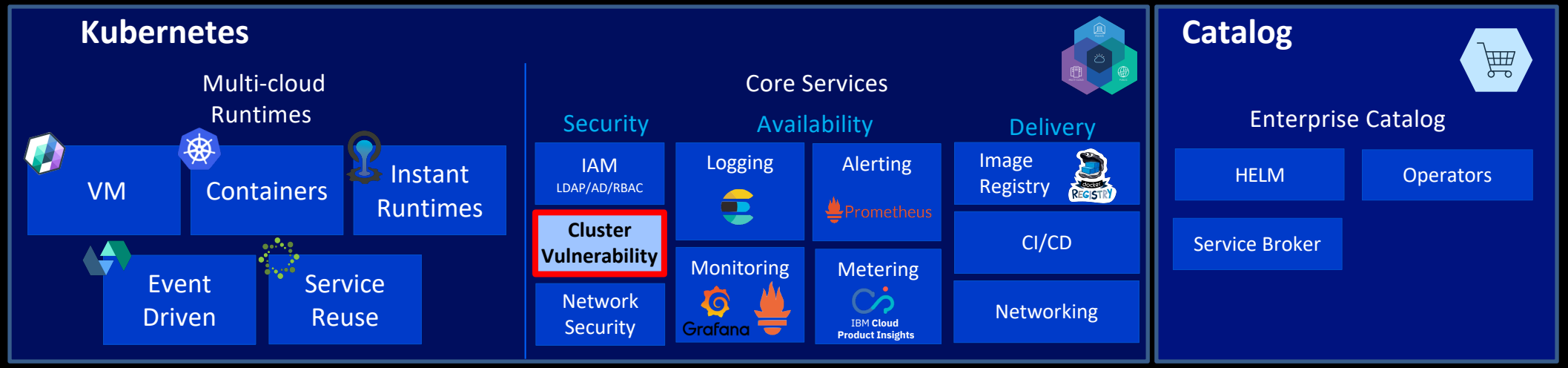
- **Unused Role**
  Roles that are created but not granted to any subject can increase the complexity of RBAC management.

- **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/

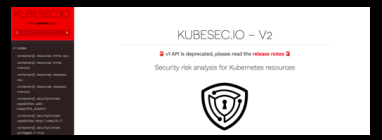
Always adapt least privileged access practices

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": [
```



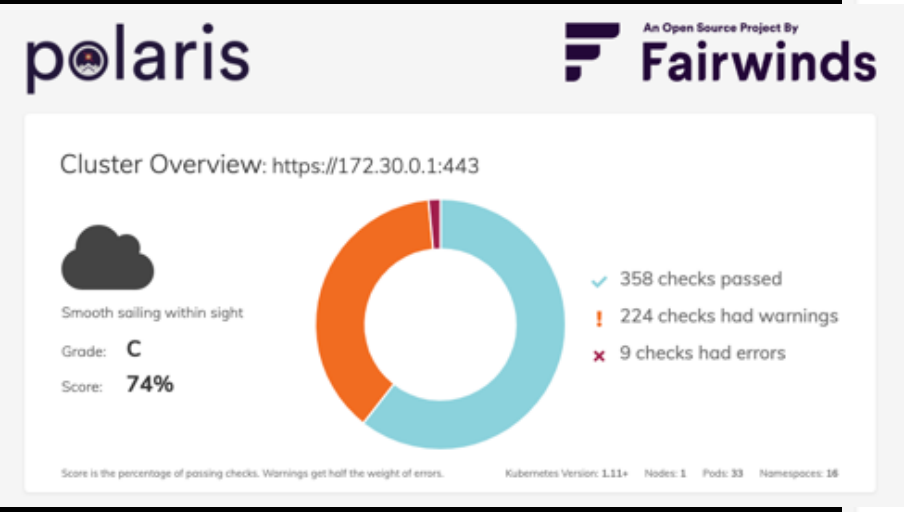
```
{
  "selector": ".spec .serviceAccountName",
  "reason": "Service accounts restrict Kubernetes API access and should be configured with least privilege",
  "points": 3
},
```

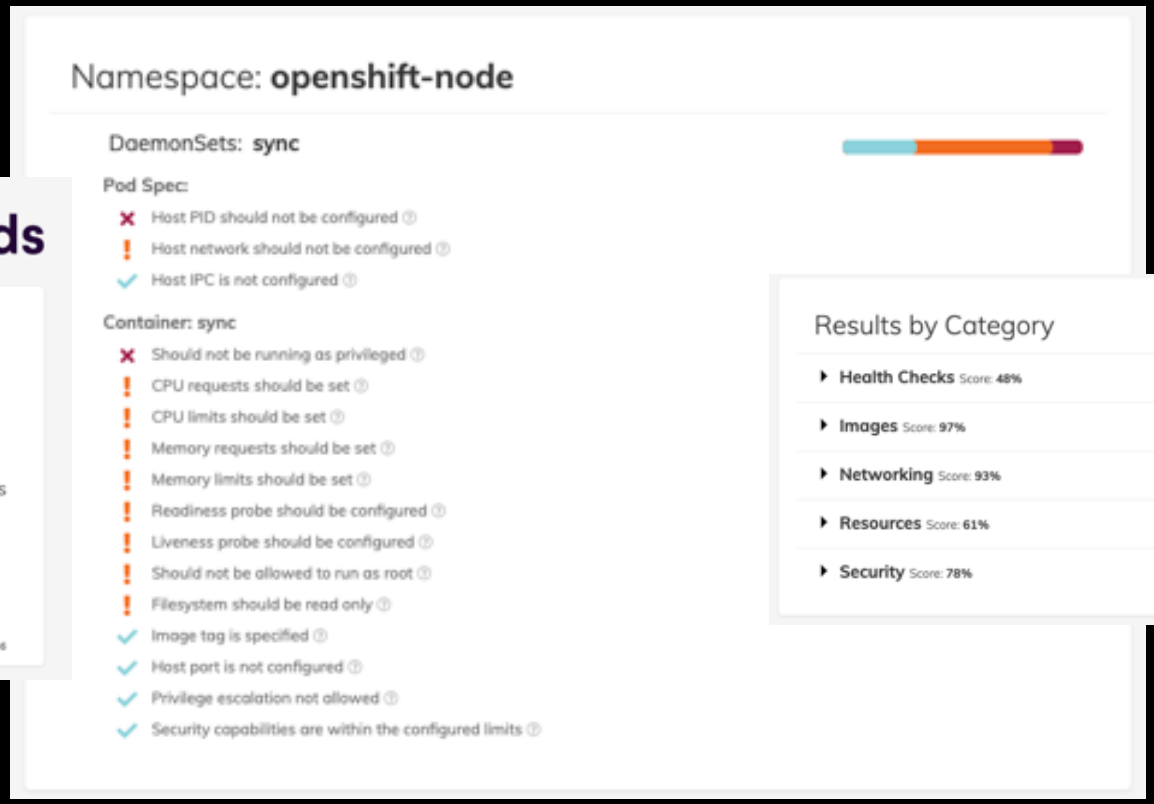

Source: https://kubesec.io/

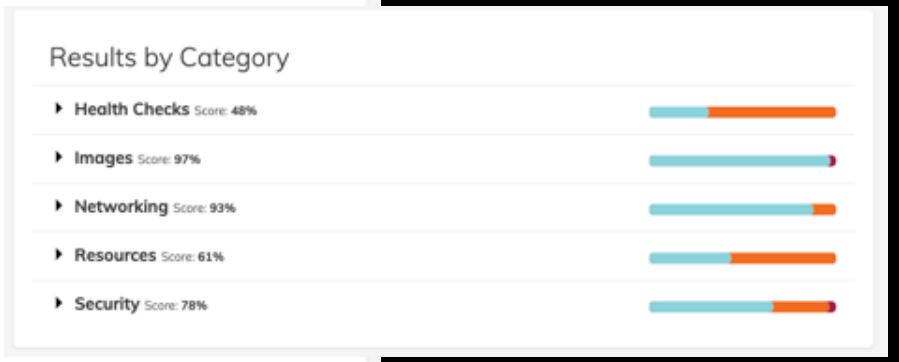
Kubernetes – Cluster Security – Secure Deployments

Scan for K8s Best Practices

Polaris – Validation of best practices





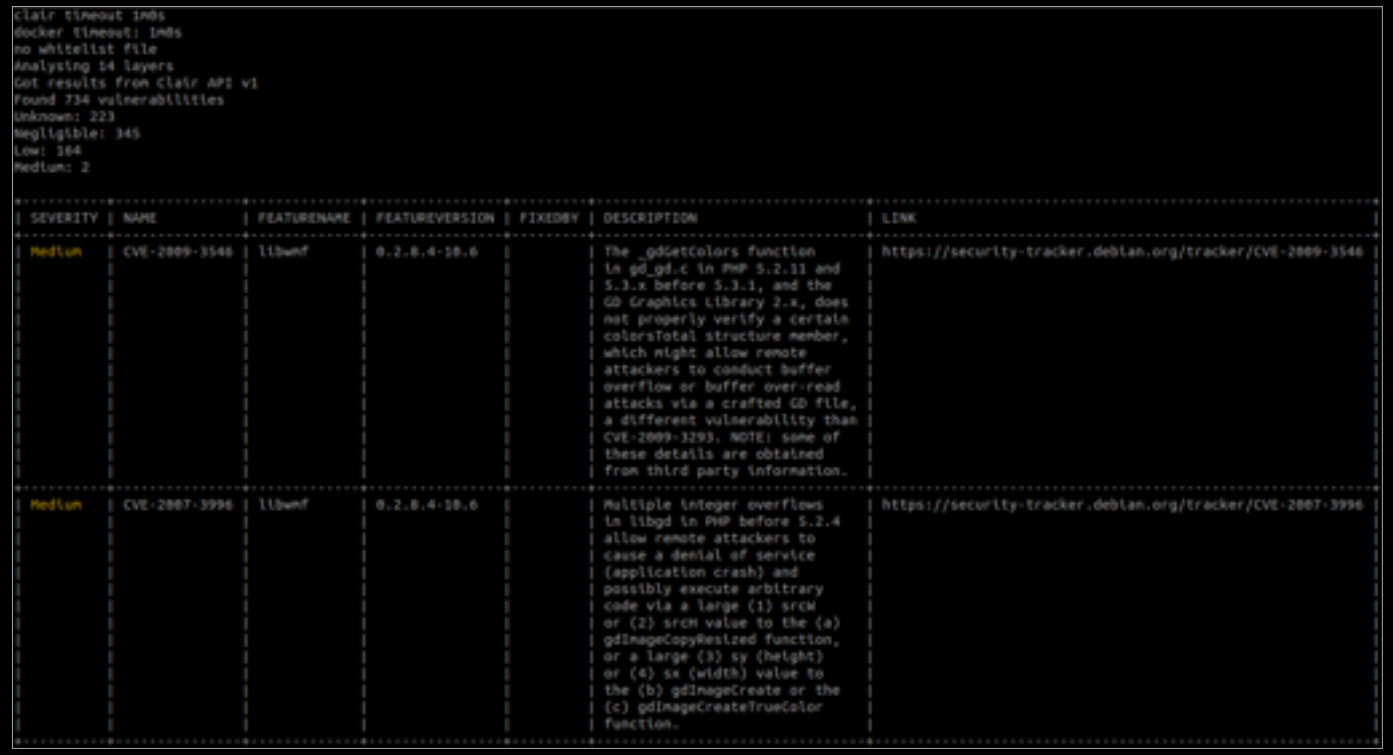


Source: https://github.com/FairwindsOps/polaris

Kubernetes – Cluster Security – Secure Images

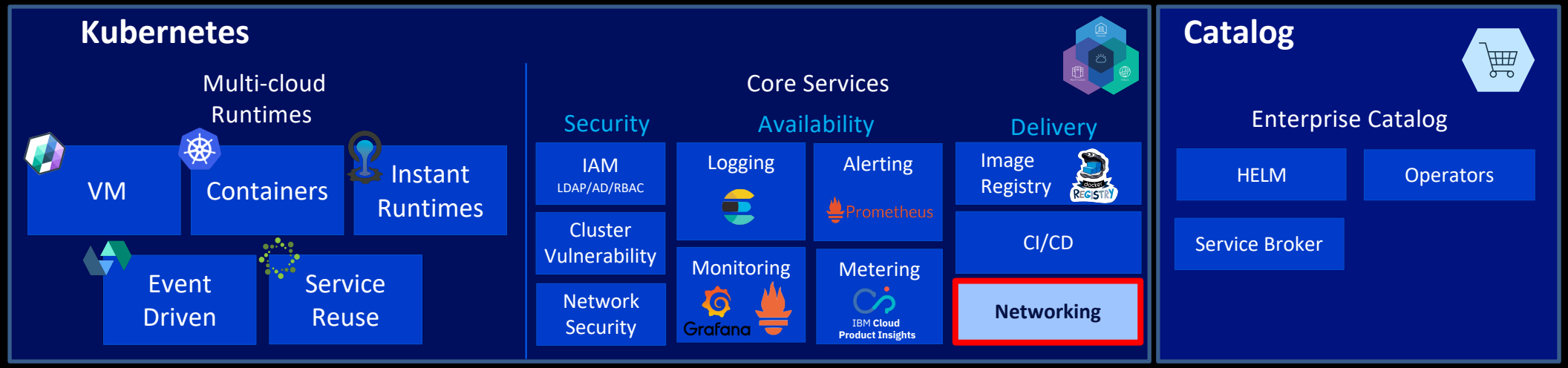
Scan images for vulerabilities

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).

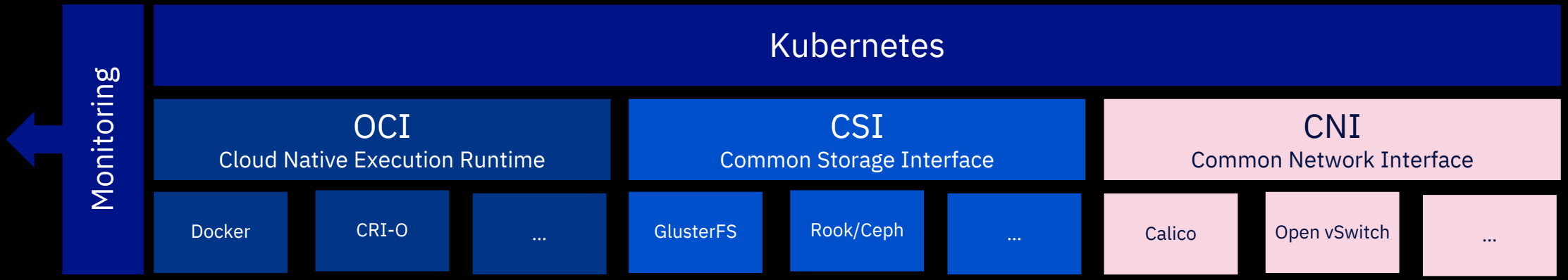


Source: https://github.com/quay/clair

Kubernetes – Core Services – Mesh Networking



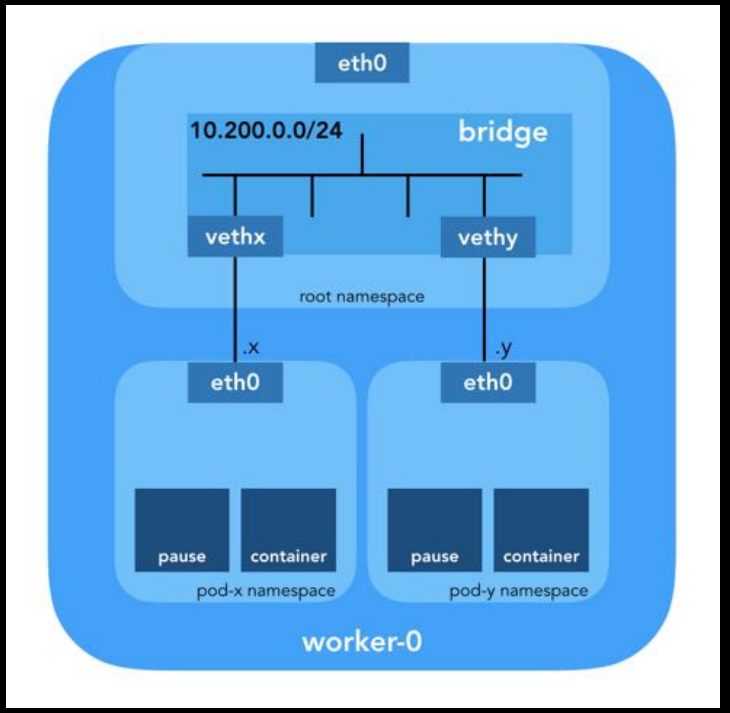
Kubernetes – Core Services – Network Security



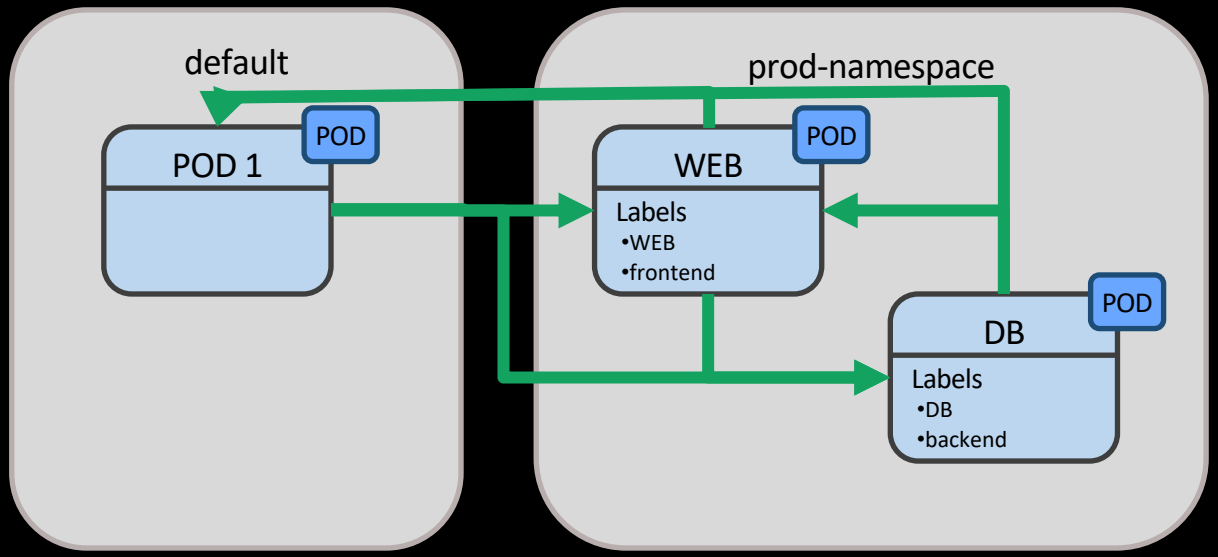
CNI – Common Network Interface

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

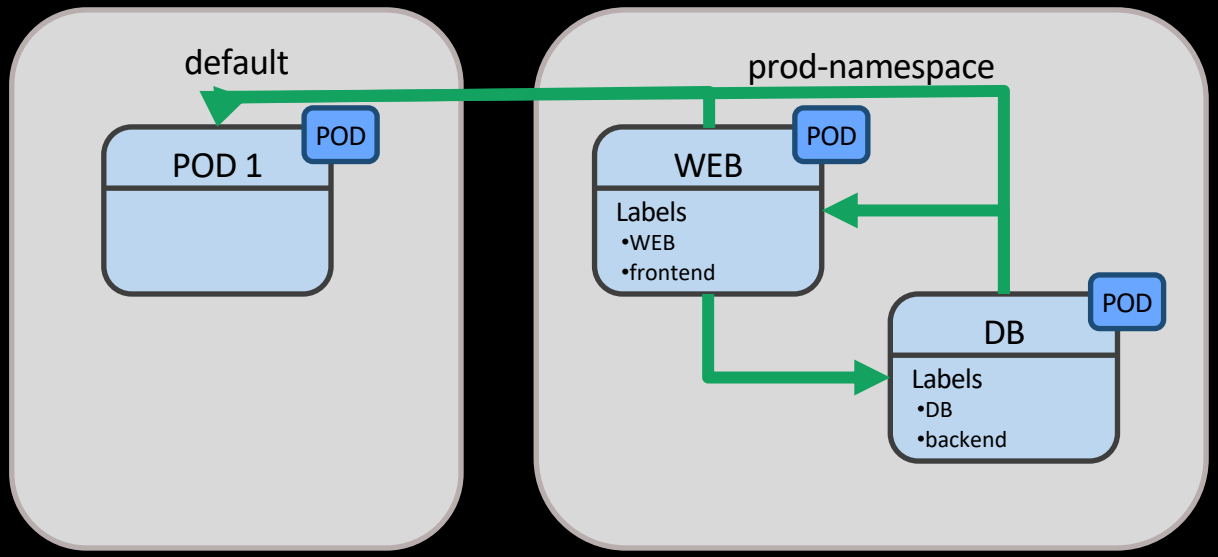
- **Calico**
  Virtual networking and network security for containers, VMs, and bare metal services.
- **Open vSwitch**
  Production quality, multilayer virtual switch
- ...



Kubernetes – Core Services – Network Security

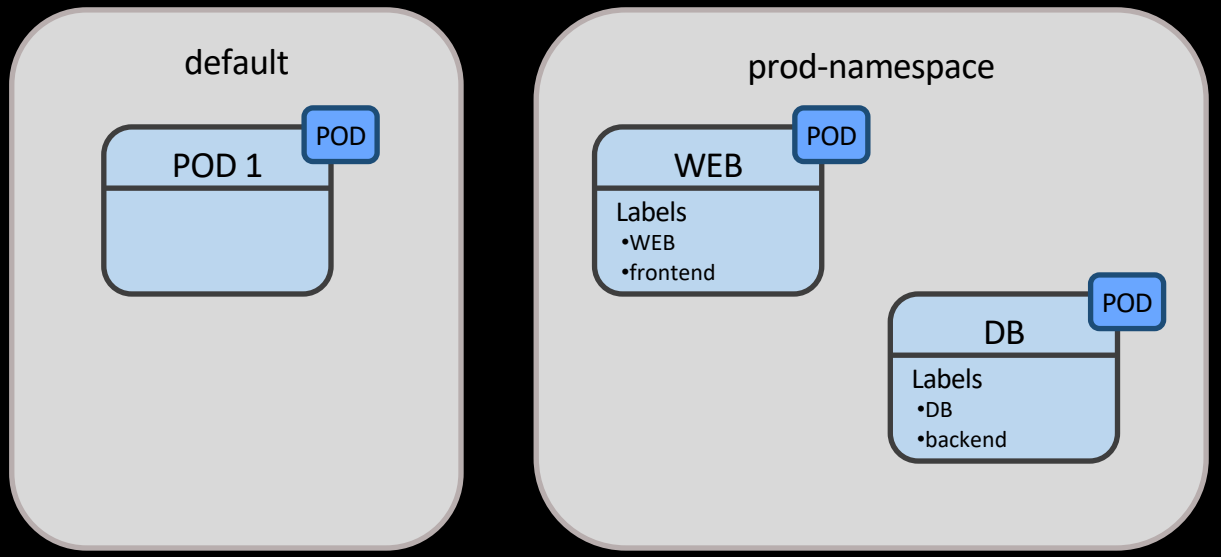


Kubernetes – Core Services – Network Security



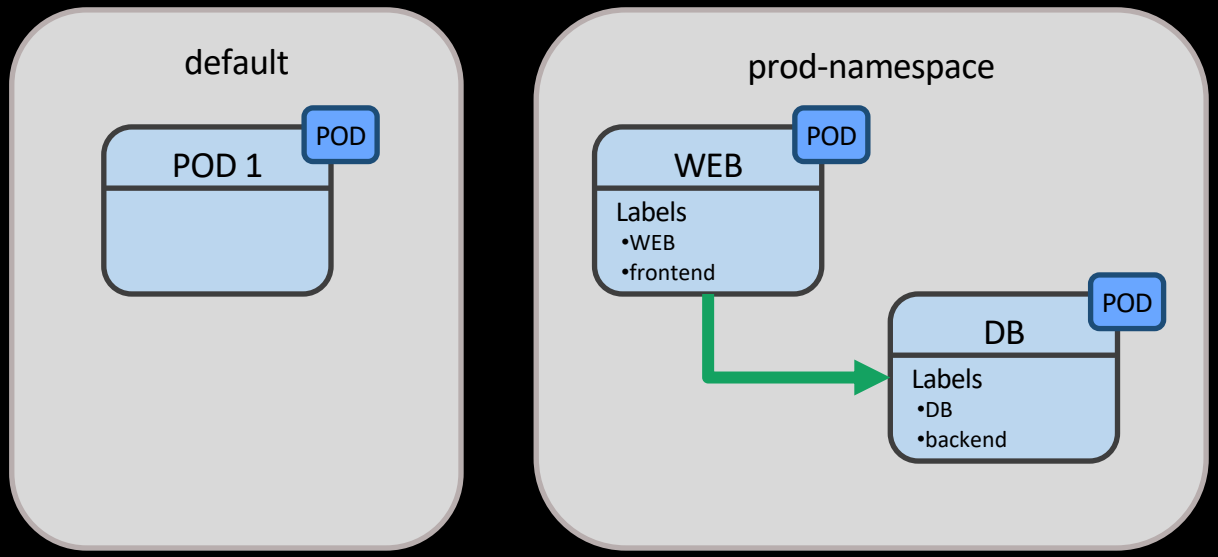

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
  namespace: demo-namespace
spec:
  podSelector:
    matchLabels: {}
```

Kubernetes – Core Services – Network Security



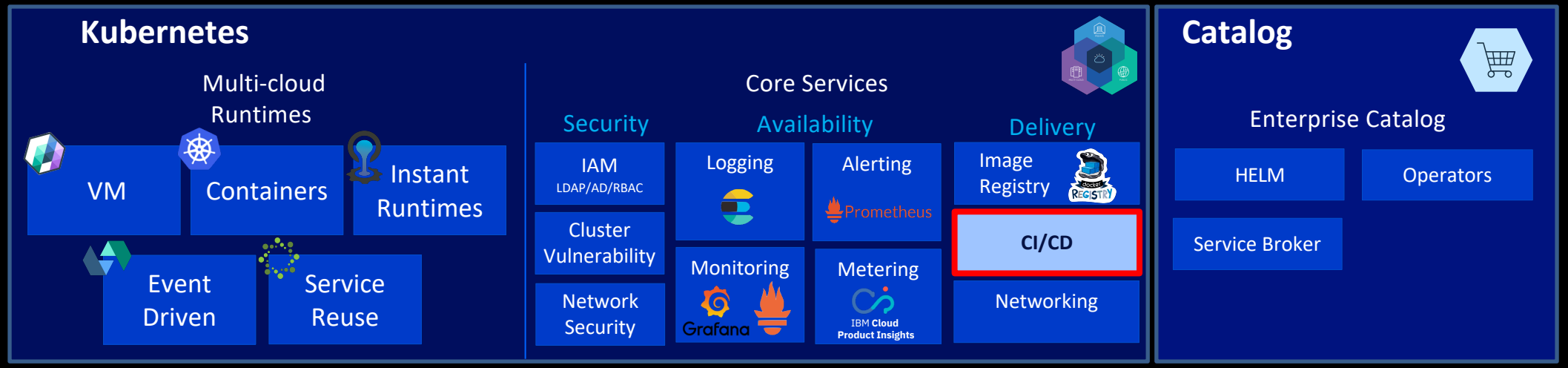

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny-all
spec:
  podSelector:
    matchLabels: {}
```

Kubernetes – Core Services – Network Security



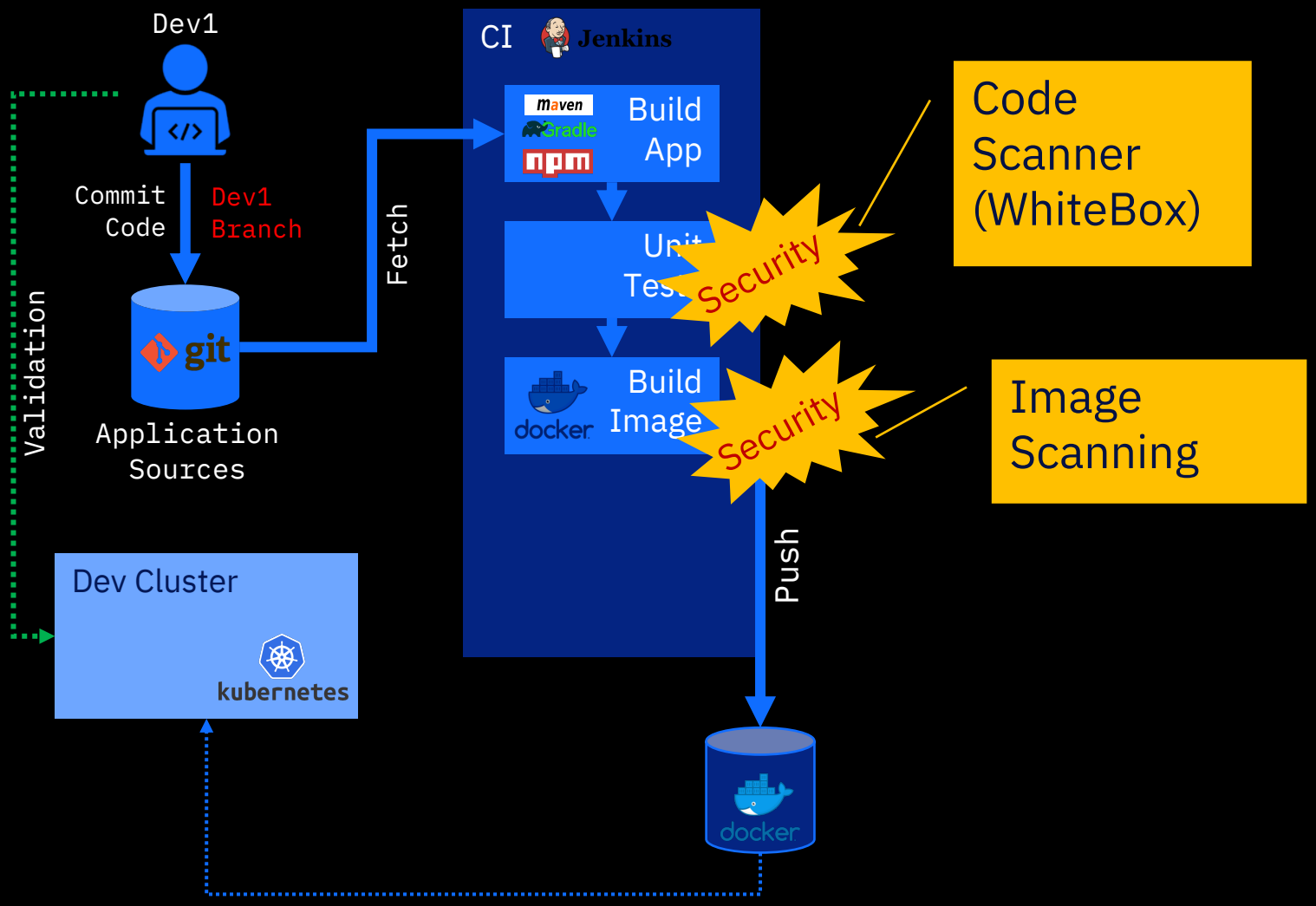

```
kind: NetworkPolicy
metadata:
  name: access-frontend-backend
  namespace: prod-namespace
spec:
  podSelector:
    matchLabels:
      run: DB
  ingress:
    - from:
      - podSelector:
          matchLabels:
            run: WEB
```

Kubernetes – Core Services – Mesh Networking



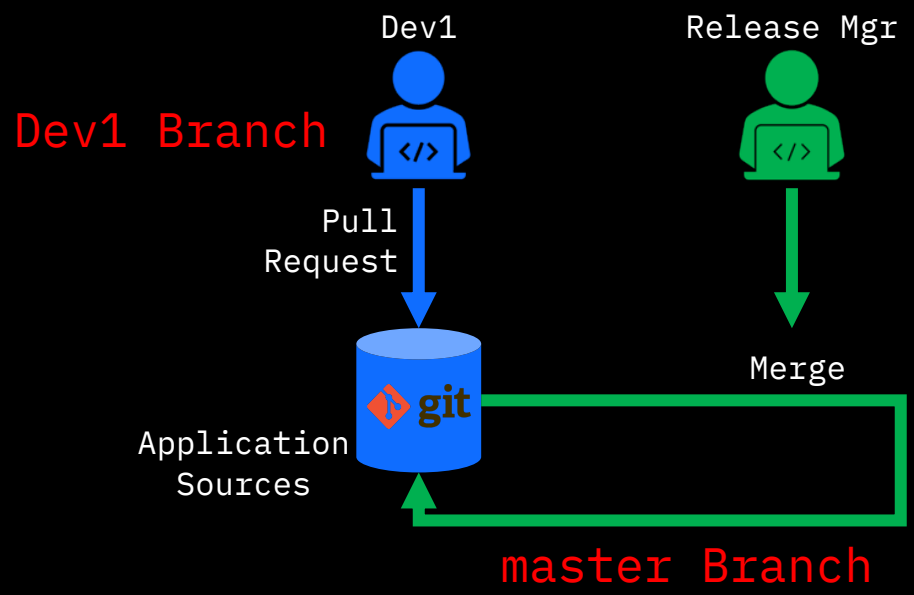
Kubernetes – CI/CD Pipeline

Development – Dev1 Branch

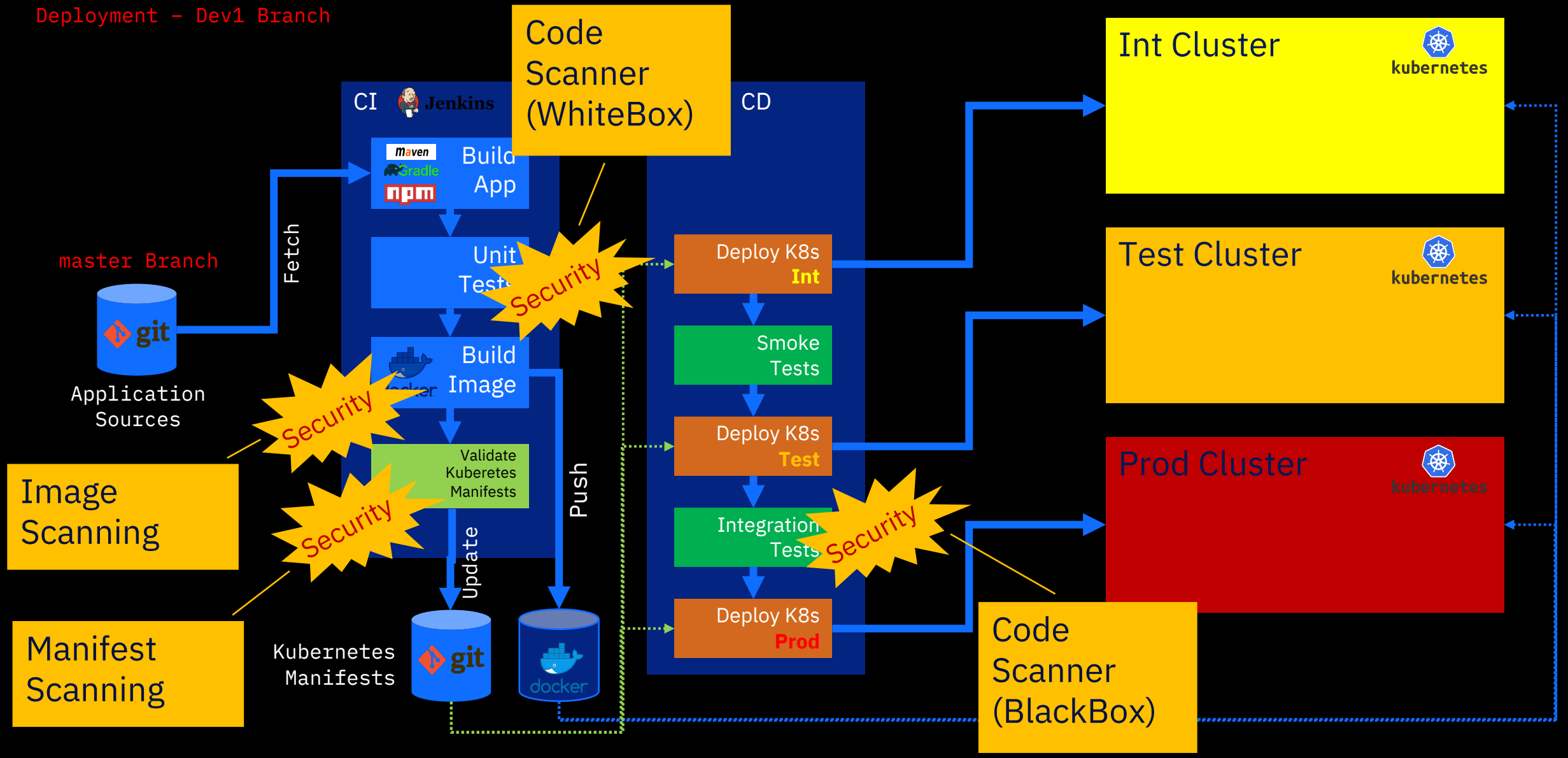


Kubernetes – CI/CD Pipeline

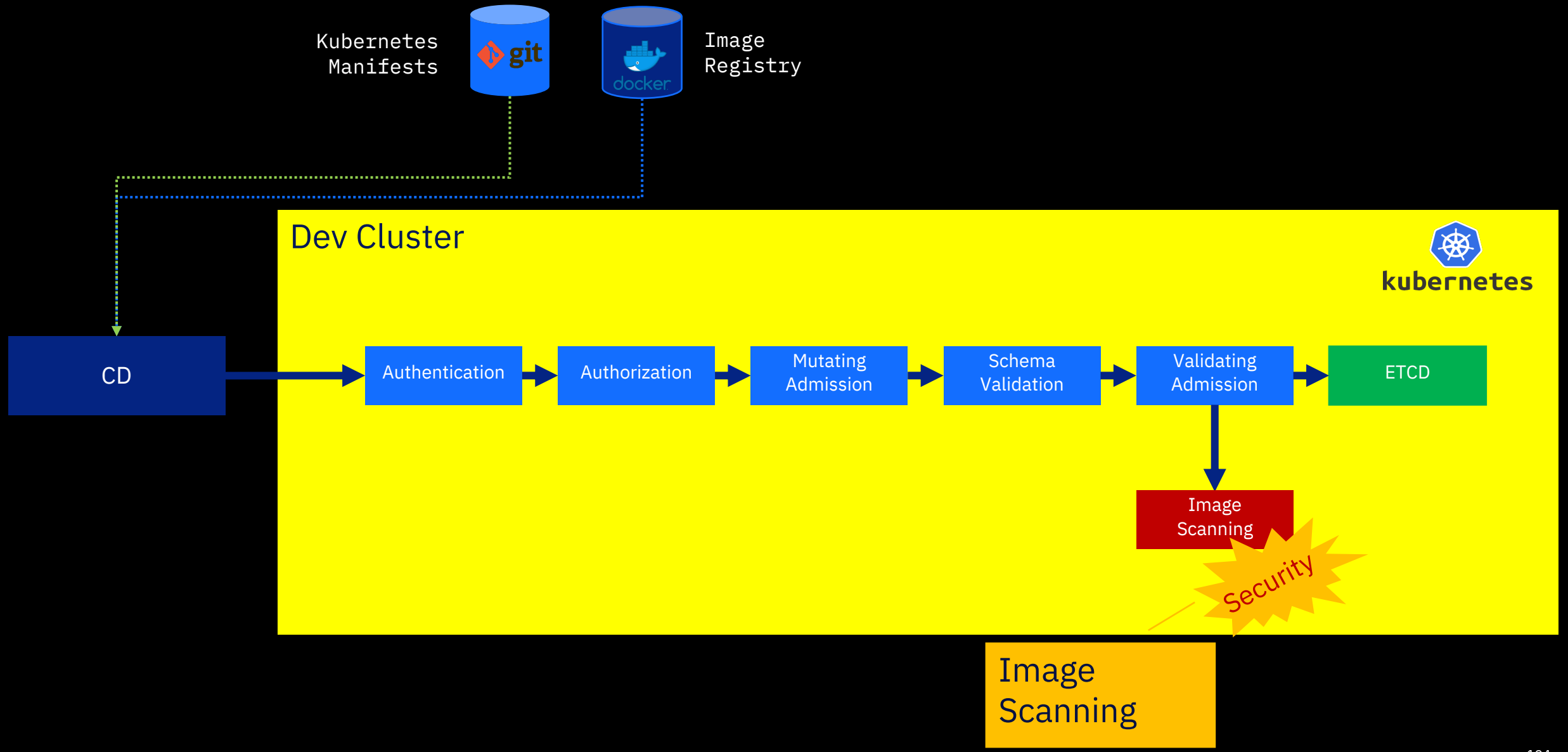
Merge – Dev1 Branch to master Branch



Kubernetes – CI/CD Pipeline



Kubernetes – CD Pipeline



DevOps – GitOps

Full Audit Trail

GitOps is using git, which is an excellent database to persist the changes made to the system.

Everything as a code

In the cloud, both CI and CD should be defined as code first. The desired state of environments should also be set in full as a declarative code.

Test Your Manifests Before You Commit

A lot of issues that escape into pre-production environments can be prevented by testing the changes before committing and pushing them.

Two Repos: One For App Source Code, Another For Manifests

Clean separation of application code vs. application config.

Separation of access

The developers who are developing the application, may not necessarily be the same people who can/should push to production environments, either intentionally or unintentionally.

Source: Dmitri Lerko - A Year With Gitops in Production and Arge - 5 GitOps Best Practices

Kubernetes Security – Some tools I use

RBAC

RBAC-lookup - https://github.com/FairwindsOps/rbac-lookup
rakkess - https://github.com/corneliusweig/rakkess
rbac-view – https://github.com/jasonrichardsmith/rbac-view
rback - https://github.com/team-soteria/rback

Scanning

Polaris - https://github.com/FairwindsOps/polaris
Clair - https://github.com/coreos/clair

Management

K9s - https://github.com/derailed/k9s

Kubernetes Security – Some Reading Tips

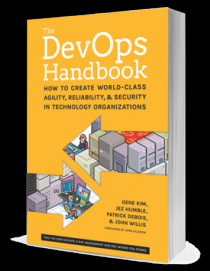
https://kubernetespodcast.com/episode/065-attacking-and-defending-kubernetes/
https://en.wikipedia.org/wiki/Red_team
https://blog.ropnop.com/attacking-default-installs-of-helm-on-kubernetes/
https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/
https://kubernetes.io/blog/2019/03/21/a-guide-to-kubernetes-admission-controllers/
https://kubernetes.io/docs/tutorials/clusters/apparmor/
https://kubernetes.io/docs/concepts/policy/pod-security-policy/
https://kubernetes.io/docs/concepts/storage/volumes/#hostpath

IBM

QUESTIONS?

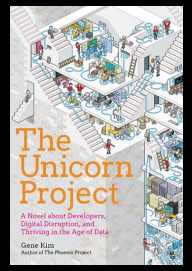
IBM **Cloud**

DevOps - – Some Reading Tips



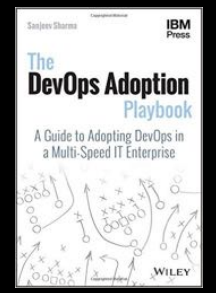
How To Create World-Class Agility, Reliability, & Security in Technology Organizations

A view into the cultural challenges of adopting DevOps and best practices

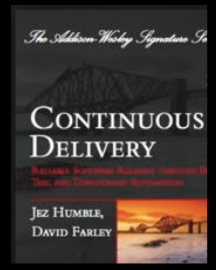


A Novel about Developers, Digital Disruption, and Thriving in the Age of Data

Achieve business goals, maximize IT productivity, and deliver innovation with enterprise-scale DevOps.



Automate deployments using production-like environments and accelerate delivery cycles



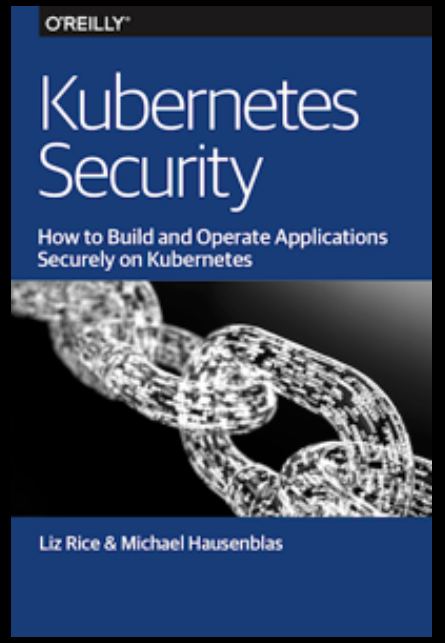
Kubernetes – Some Reading Tips

We wrote the books on cloud adoption...

ibm.com/cloud/garage/adoption

...and Kubernetes in the enterprise

https://ibm.co/2LQketN



https://kubernetes-security.info/

The Journey to Cloud
Docker - *Hands-On*

07

IBM **Cloud**

Remember your Team Color

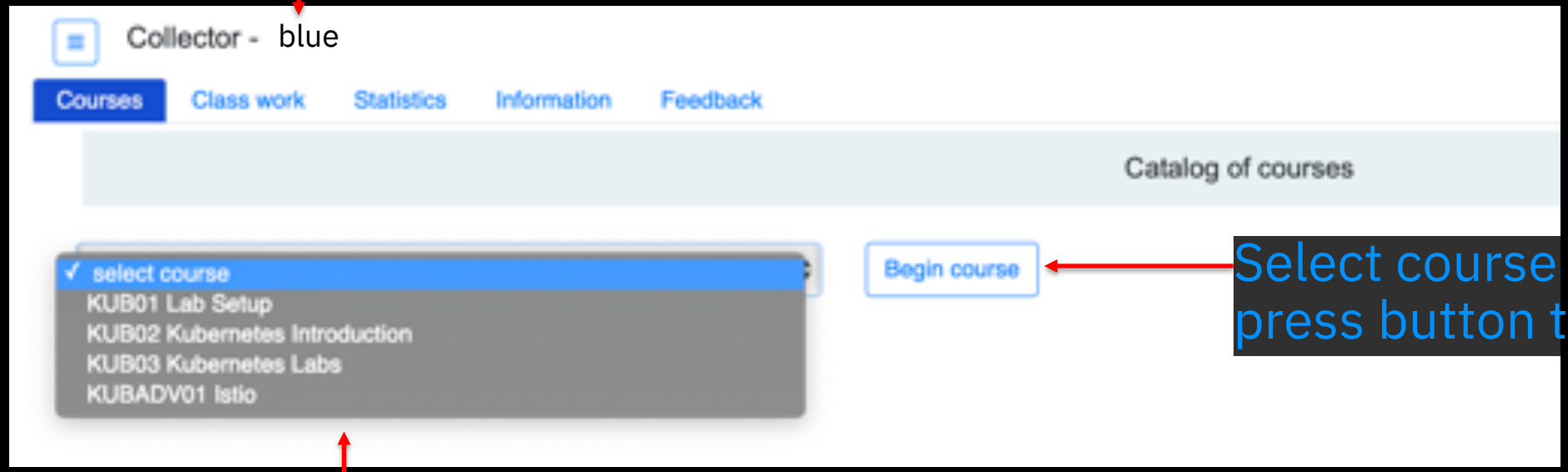
black 31701	olive 31711	peru 31715
white 31702	brown 31712	chocolate 31716
red 31703	lightblue 31713	orchid 31717
blue 31704	orange 31708	gold 31718
yellow 31705	purple 31709	pink 31719
lime 31706	maroon 31710	violet 31720
cyan 31707	firebrick 31714	

Collector - Accessing team web site

http://158.177.137.195:{port#}

blue 31704

Team name / color will be shown



JTC01 – Docker Labs

Lab 0 : Docker basics

Lab 1 : Build a Docker image

Lab 2 : Run a Docker image

Lab 3 : Use the Portainer tool

Lab 4 : Deploy a more complex Docker application

Lab 5 : Push a Docker image into a registry

READY SET GO!!!!

http://158.177.137.195:{port#}

Duration: 60 mins

IBM

QUESTIONS?

IBM **Cloud**

The Journey to Cloud
Kubernetes - *Hands-On*

08

IBM **Cloud**

JTC02 – Kubernetes Labs

Lab 1: Refresher/overview over 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

READY SET GO!!!!

http://158.177.137.195:{port#}

Duration: 120 mins

IBM

QUESTIONS?

IBM **Cloud**

The Journey to Cloud
Kubernetes Security - *Hands-On*

09

IBM **Cloud**

JTC16 – Kubernetes Labs

~~Lab 1: NetworkPolicies~~

~~Lab 2: Role Based Acces Control (RBAC)~~

~~Lab 3: Service Accounts~~

~~Lab 4: Security Tooling~~

Lab 5: Image scanning

READY SET GO!!!!

http://158.177.137.195:{port#}

Duration: 30 mins

IBM

QUESTIONS?

IBM **Cloud**

Documentation

https://github.com/niklaushirt/k8s_training_public

Sources

https://github.com/niklaushirt/training

Always-on training environment

http://158.177.137.195:31999/