Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Informatik

CAS Cloud and Platform Manager

Microservices, Containers und Kubernetes

Niklaus Hirt
Consulting IT Specialist
Cloud and DevOps Architect

nikh@ch.ibm.com

Rotkreuz, 26.05.2023

Who am I?

Niklaus Hirt

Passionate about tech for 40 years

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

✉ nikh@ch.ibm.com
@nhirt

Agenda – Microservices and Containers – A crash course

Module 1: Microservices
Module 2: Microservices - Let’s get real
Module 3: Containers
Module 4: Kubernetes - Basics
Lunch

Agenda – Microservices and Containers – A crash course

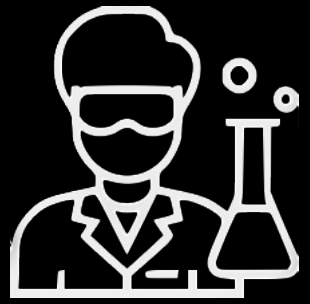
Module 5: Kubernetes - Applied
Module 6: Kubernetes - Advanced Concepts
Module 7: Kubernetes - State
Module 8: Mesh Networking (optional)

Self Study - Optional

Lab 1: Containers Hands-On

Lab 2: Kubernetes Hands-On

Instructions can be found in the Annex



Sources and documentation are available here:

Course Documents
https://github.com/niklaushirt/k8s_training_public

Course Files
https://github.com/niklaushirt/training
Online Course
https://niklaushirt.github.io/k8s_training_web/

Kubernetes Workshop Series
Microservices

01

Kodak

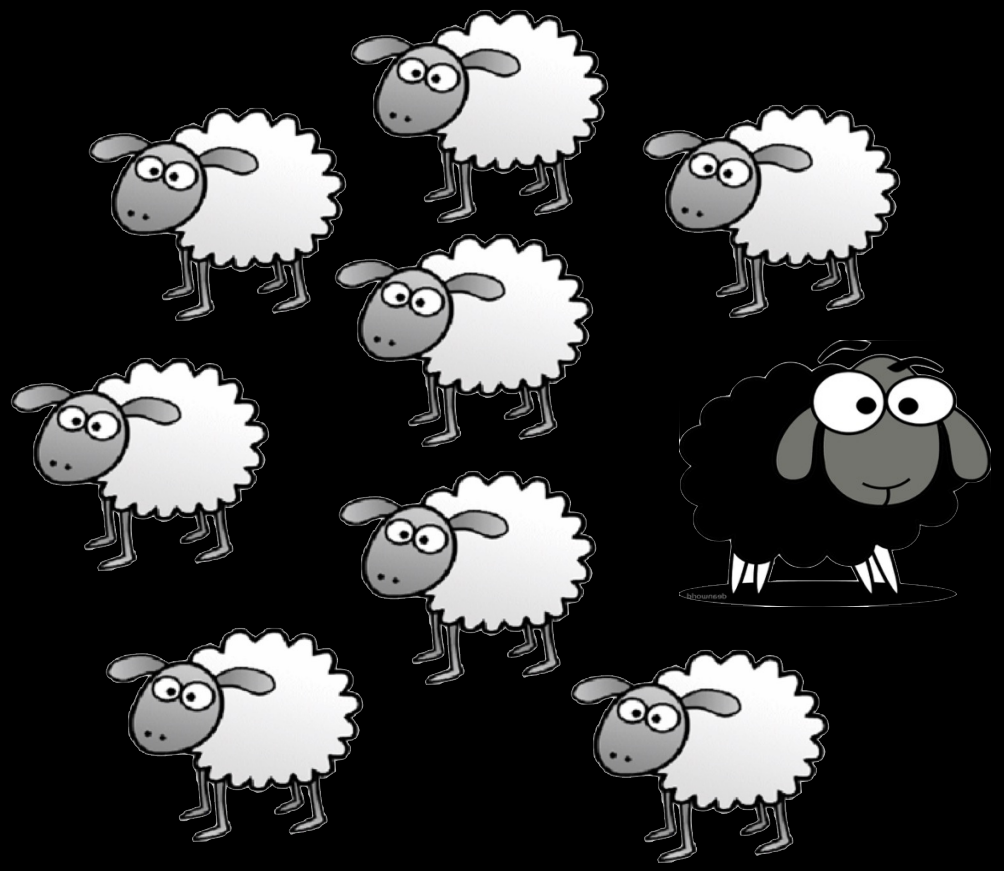
disruption

dɪs'rʌpʃn/

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

dɪs'rʌpʃn/

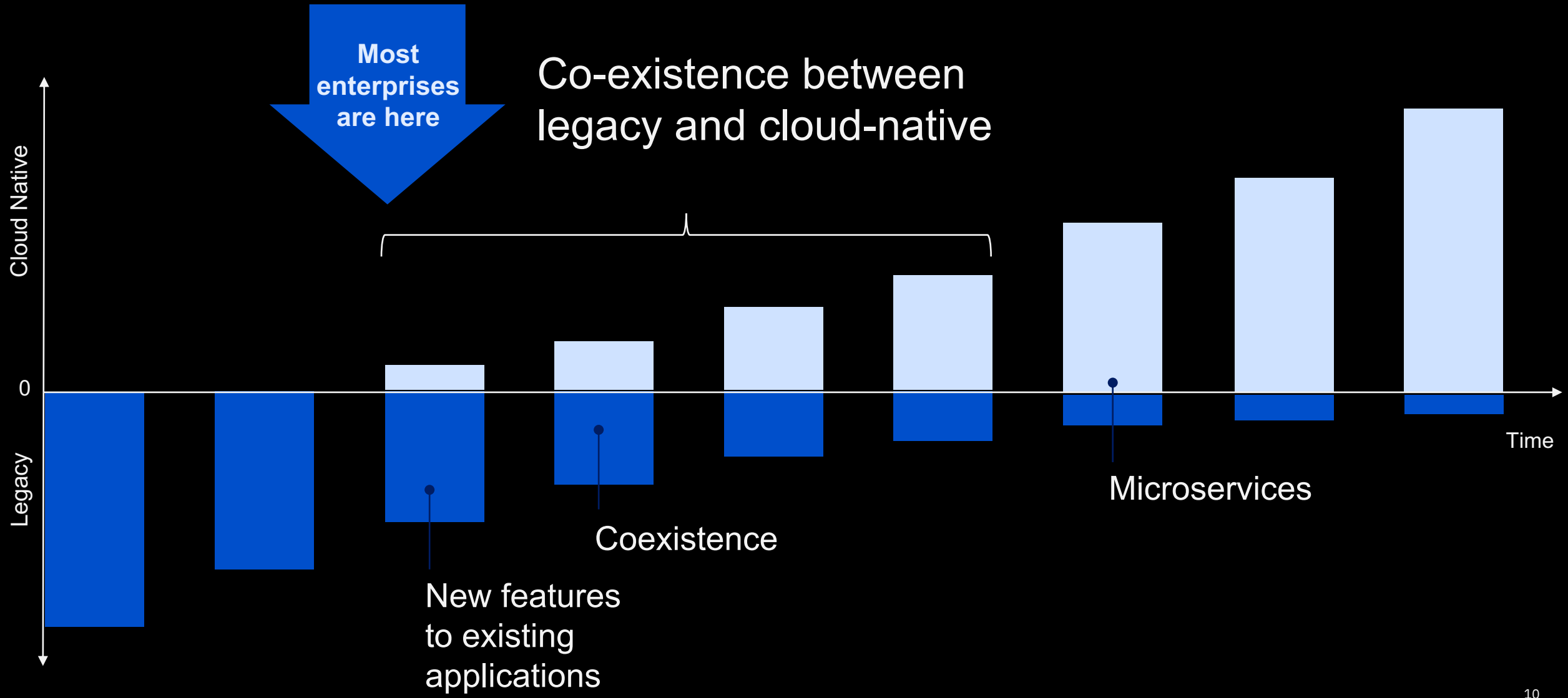
noun

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

- Continuous Integration
- Continuous Build
- Continuous Deploy
- Agile Dev Models
- DevOps

Adopt new Technology

- Architectural patterns (Microservices)
- Frameworks (Java MicroProfile, Spring ...)
- Node.js , Swift, GoLang, Python(!)

Adopt new Tools

- Git, Github, Gitlab
- CI/CD (ArgoCD, ...)
- AIOps

Adopt Cloud-Native

- Cloud
- Containers
- Kubernetes

How do you achieve digital transformation?

Adopt new Processes

- Continuous Integration
- Continuous Build
- Continuous Deploy
- Agile Dev Models
- DevOps

Adopt new Technology

- Architectural patterns (Microservices)
- Frameworks (Java MicroProfile, Spring ...)
- Node.js , Swift, GoLang, Python(!)

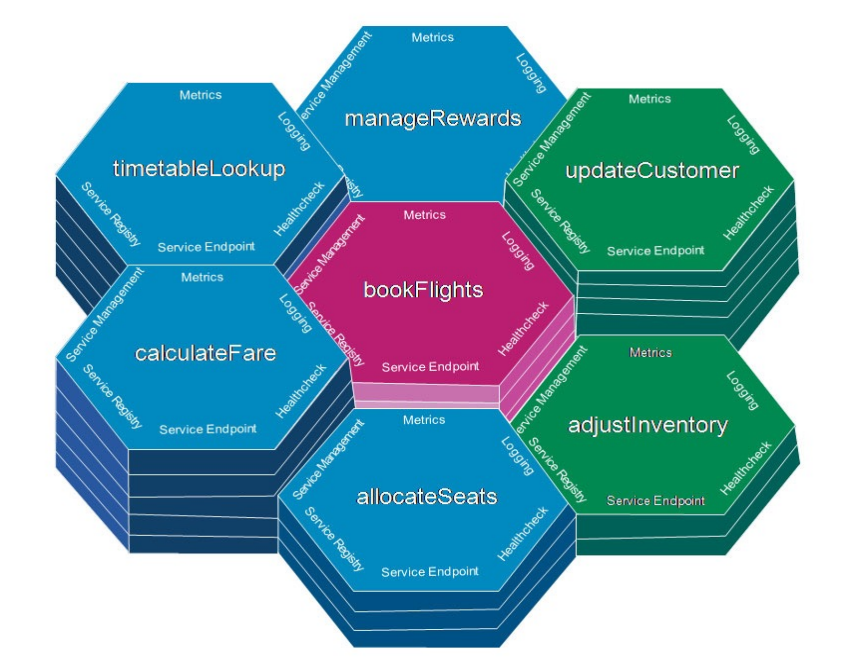
Adopt new Tools

- Git, Github, Gitlab
- CI/CD (ArgoCD, ...)
- AIOps

Adopt Cloud-Native

- Cloud
- Containers
- Kubernetes

Microservices

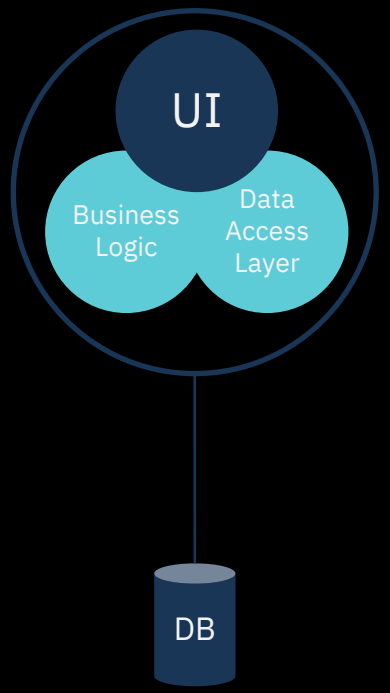


Decomposing an application into single function modules which are independently deployed and operated

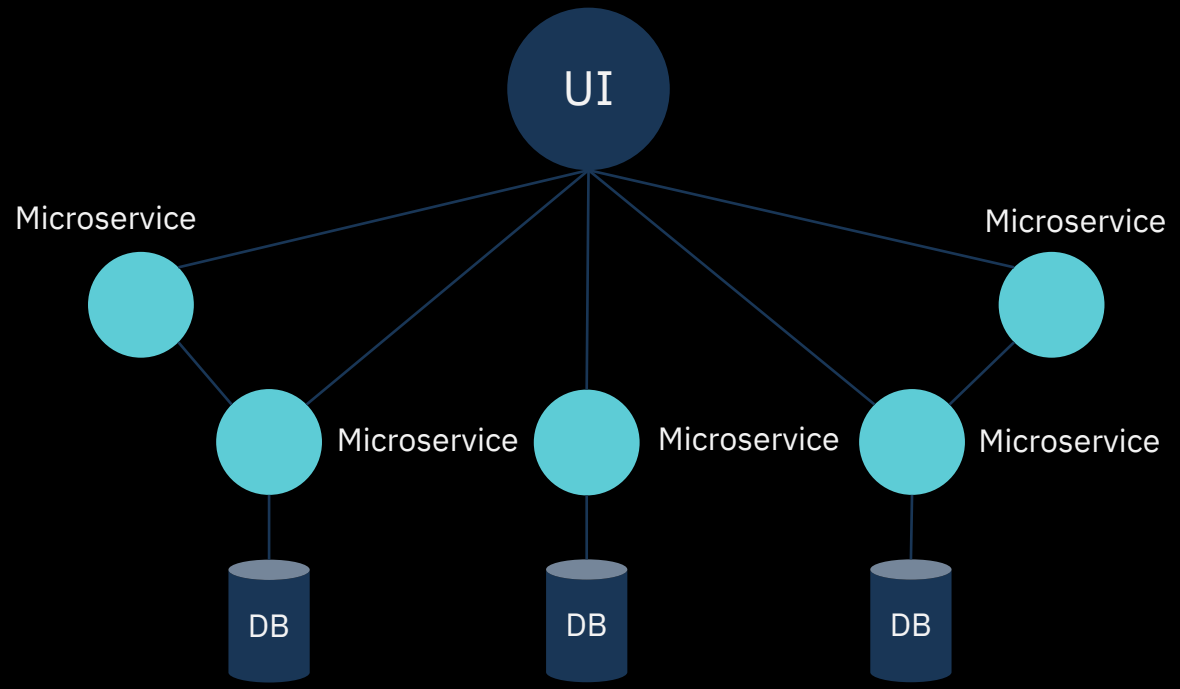
Accelerate delivery by minimizing communication and coordination between people

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

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 scaling 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 better isolates problems
- Can be developed in a different language
- Manages its own data to select the best technology and schema

Source: IBM Architecture Center

That said....

Microservices are hard

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

QUESTIONS?

Kubernetes Workshop Series

Let‘s get real

02

Microservices – The Pitfall

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 – breaking down the ~~monolith~~ beast

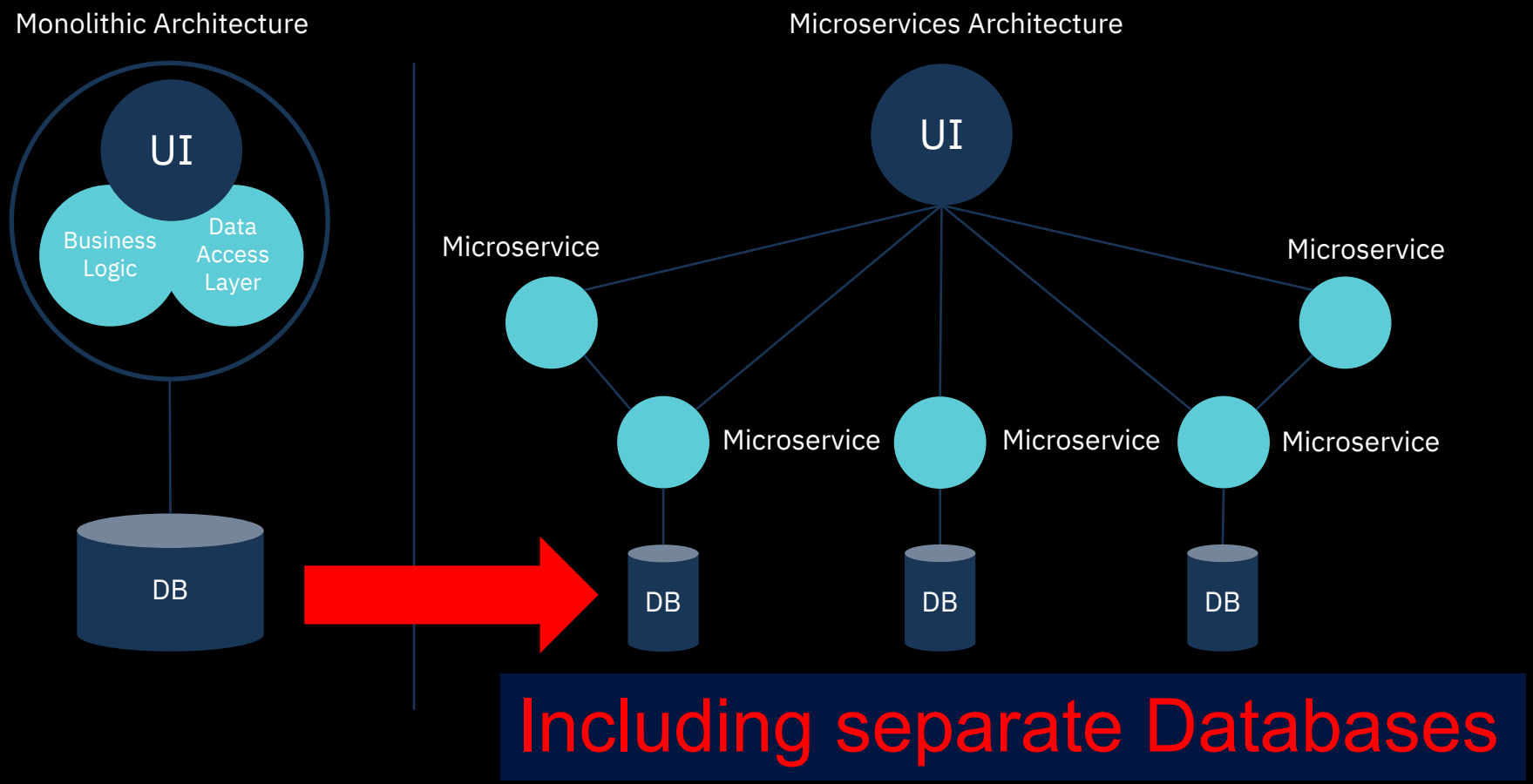
How to identify candidates

- 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/

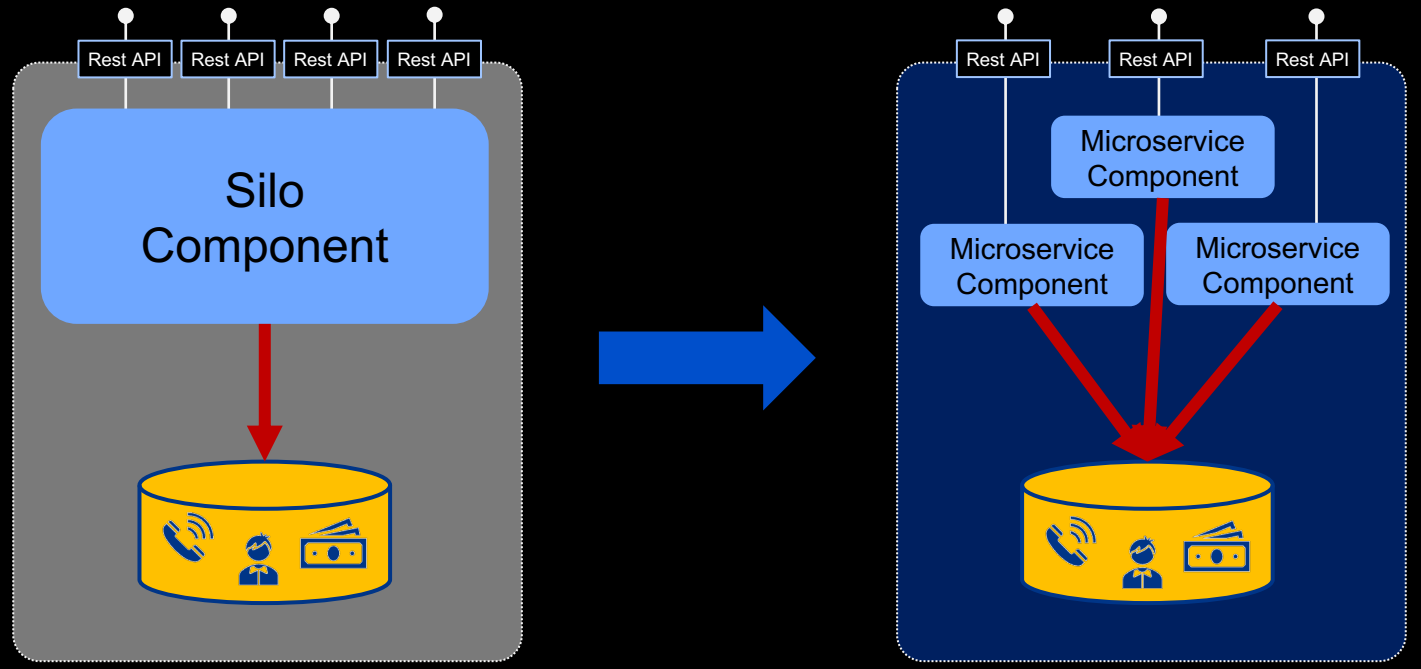
Microservices – Database refactoring

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



Microservices – Database refactoring - Antipattern

Common mistake

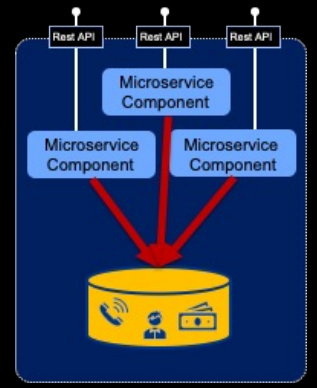


Source: IBM Architecture Center

Microservices – monolithic database

Challenges

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

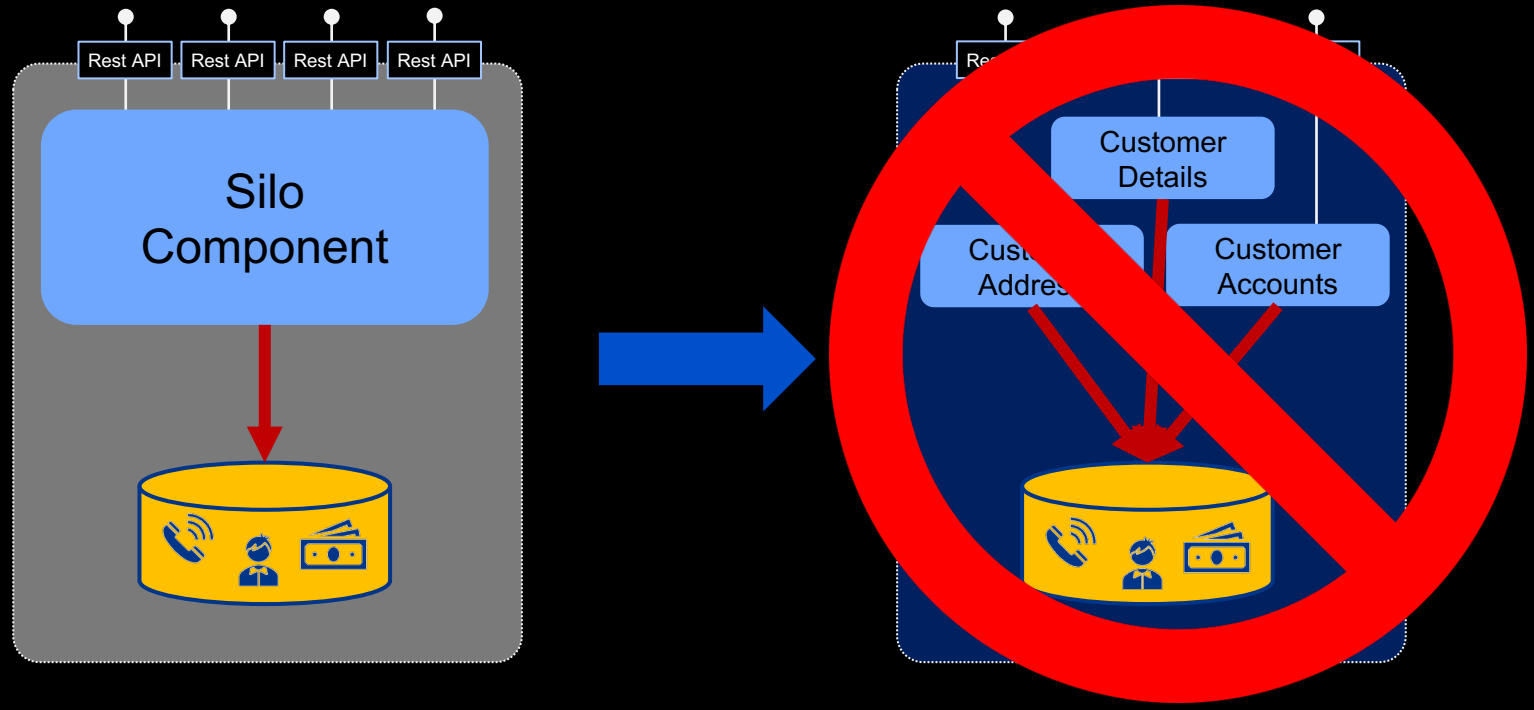


Source: https://dzone.com/articles/breaking-the-monolithic-database-in-your-microserv

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.

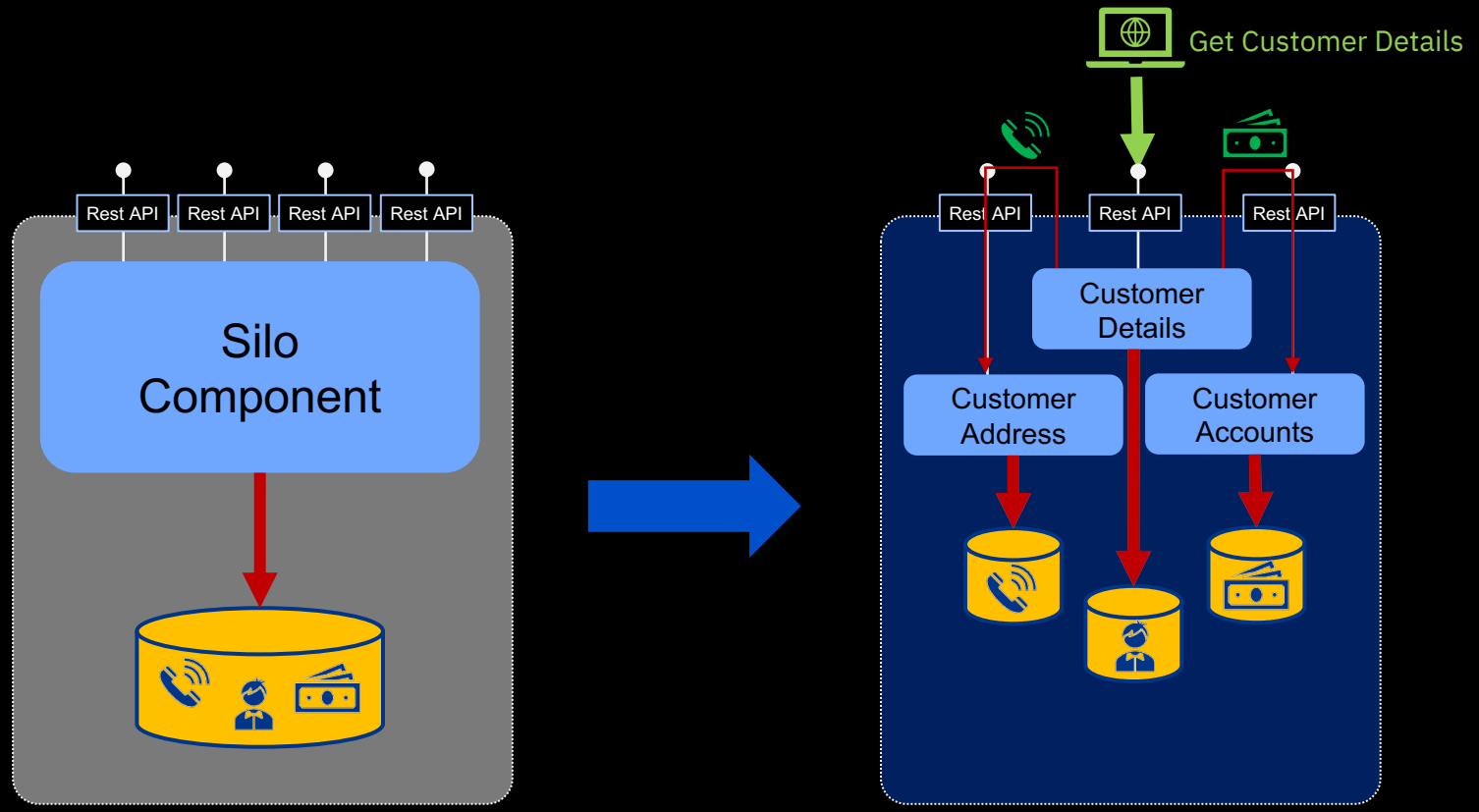


Source: IBM Architecture Center

Microservices – Database refactoring

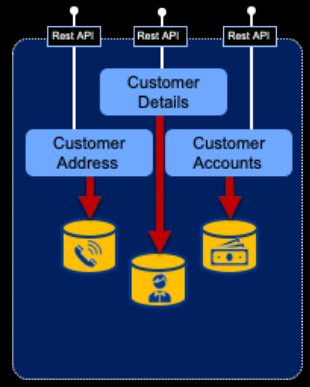
Data Isolation

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

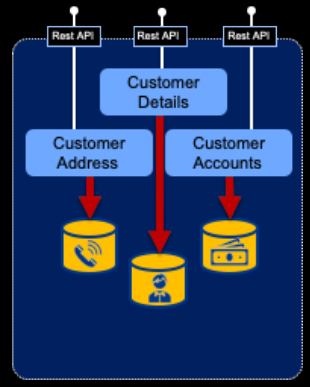


Benefits

- Avoid unexpected data modification – the API enforces consitency
- Make changes to our system without blocking progress of other parts of the system
- Make changes to the schema/databases at our leisure
- Store the data in a project-owned databases, using the appropriate technology
- 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
- Introduces latency
- 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/

Microservices
Good or bad idea?

Microservices – reasons not to use them

Some thoughts

- If speed is your goal, microservices might not be the solution (real-time processing, …)
- Applications that require tight integration between individual components and services
- At what point is it in its lifecycle? Mission-critical?
- Not all applications are large enough to break down into microservices

Ask ourself: What **Business Problem** does this solve?

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

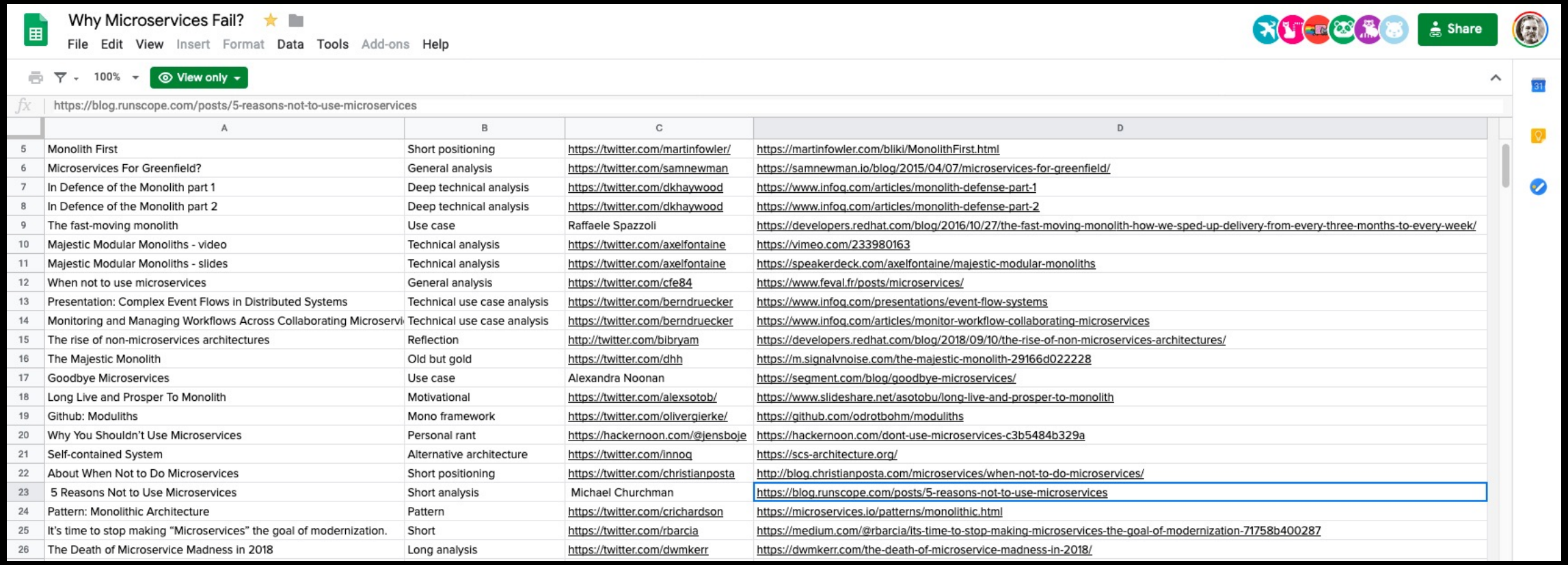
Microservices – reasons not to use them

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 – the Monolith is NOT dead



Why Microservices Fail?

File Edit View Insert Format Data Tools Add-ons Help

Share

100% View only

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

	A	B	C	D
5	Monolith First	Short positioning	https://twitter.com/martinfowler/	https://martinfowler.com/bliki/MonolithFirst.html
6	Microservices For Greenfield?	General analysis	https://twitter.com/samnewman	https://samnewman.io/blog/2015/04/07/microservices-for-greenfield/
7	In Defence of the Monolith part 1	Deep technical analysis	https://twitter.com/dkhaywood	https://www.infoq.com/articles/monolith-defense-part-1
8	In Defence of the Monolith part 2	Deep technical analysis	https://twitter.com/dkhaywood	https://www.infoq.com/articles/monolith-defense-part-2
9	The fast-moving monolith	Use case	Raffaele Spazzoli	https://developers.redhat.com/blog/2016/10/27/the-fast-moving-monolith-how-we-sped-up-delivery-from-every-three-months-to-every-week/
10	Majestic Modular Monoliths - video	Technical analysis	https://twitter.com/axelfontaine	https://vimeo.com/233980163
11	Majestic Modular Monoliths - slides	Technical analysis	https://twitter.com/axelfontaine	https://speakerdeck.com/axelfontaine/majestic-modular-monoliths
12	When not to use microservices	General analysis	https://twitter.com/cfe84	https://www.feval.fr/posts/microservices/
13	Presentation: Complex Event Flows in Distributed Systems	Technical use case analysis	https://twitter.com/berndruecker	https://www.infoq.com/presentations/event-flow-systems
14	Monitoring and Managing Workflows Across Collaborating Microservi	Technical use case analysis	https://twitter.com/berndruecker	https://www.infoq.com/articles/monitor-workflow-collaborating-microservices
15	The rise of non-microservices architectures	Reflection	http://twitter.com/bibryam	https://developers.redhat.com/blog/2018/09/10/the-rise-of-non-microservices-architectures/
16	The Majestic Monolith	Old but gold	https://twitter.com/dhh	https://m.signalvnoise.com/the-majestic-monolith-29166d022228
17	Goodbye Microservices	Use case	Alexandra Noonan	https://segment.com/blog/goodbye-microservices/
18	Long Live and Prosper To Monolith	Motivational	https://twitter.com/alexsotob/	https://www.slideshare.net/asotobu/long-live-and-prosper-to-monolith
19	Github: Moduliths	Mono framework	https://twitter.com/olivergierke/	https://github.com/odrotbohm/moduliths
20	Why You Shouldn't Use Microservices	Personal rant	https://hackernoon.com/@jensboje	https://hackernoon.com/dont-use-microservices-c3b5484b329a
21	Self-contained System	Alternative architecture	https://twitter.com/innoq	https://scs-architecture.org/
22	About When Not to Do Microservices	Short positioning	https://twitter.com/christianposta	http://blog.christianposta.com/microservices/when-not-to-do-microservices/
23	5 Reasons Not to Use Microservices	Short analysis	Michael Churchman	https://blog.runscope.com/posts/5-reasons-not-to-use-microservices
24	Pattern: Monolithic Architecture	Pattern	https://twitter.com/crichardson	https://microservices.io/patterns/monolithic.html
25	It's time to stop making "Microservices" the goal of modernization.	Short	https://twitter.com/rbarcia	https://medium.com/@rbarcia/its-time-to-stop-making-microservices-the-goal-of-modernization-71758b400287
26	The Death of Microservice Madness in 2018	Long analysis	https://twitter.com/dwmkerr	https://dwmkerr.com/the-death-of-microservice-madness-in-2018/

https://docs.google.com/spreadsheets/d/1vjnjAII_8TZBv2XhFHra7kEQzQpOHSZpFIWDjynYYf0

QUESTIONS?

Kubernetes Workshop Series

~~Docker~~ **Containers**

03

Everybody Loves Containers

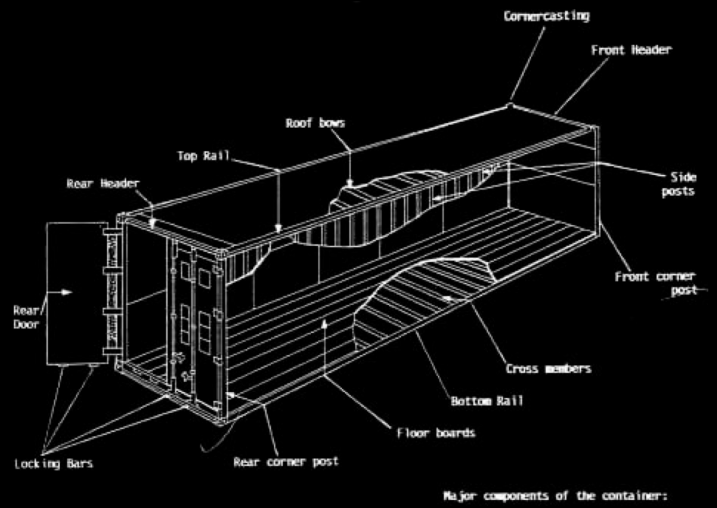
A standard way to package an application and all its dependencies so that it can be moved between environments and run without changes.

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



Major components of the container:

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

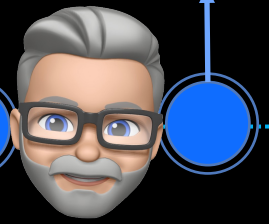
1979 chroot

2007 cgroups

2008 LXC

2011 warden

2012 docker



2014 kubernetes

2015 CLOUD NATIVE COMPUTING FOUNDATION

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".

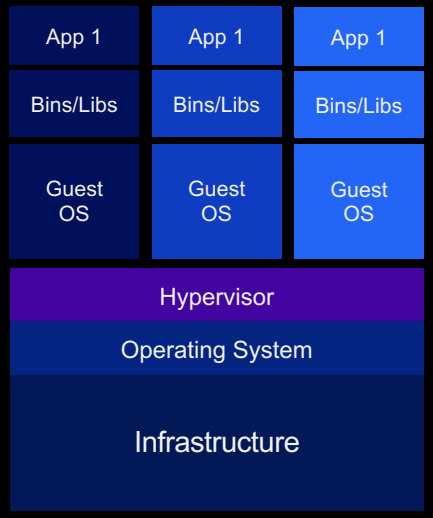
https://en.wikipedia.org/wiki/ChrootChr

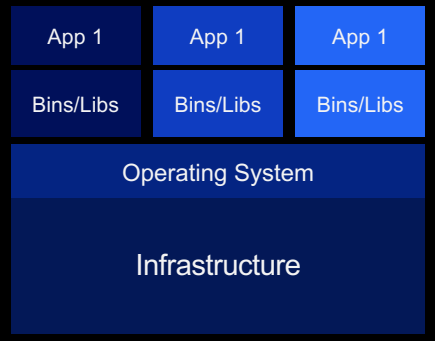
LXC (Linux Containers) is an operating-system-level virtualization environment for running multiple isolated Linux systems (containers) on a single Linux control host.

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

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

VMs vs. Containers

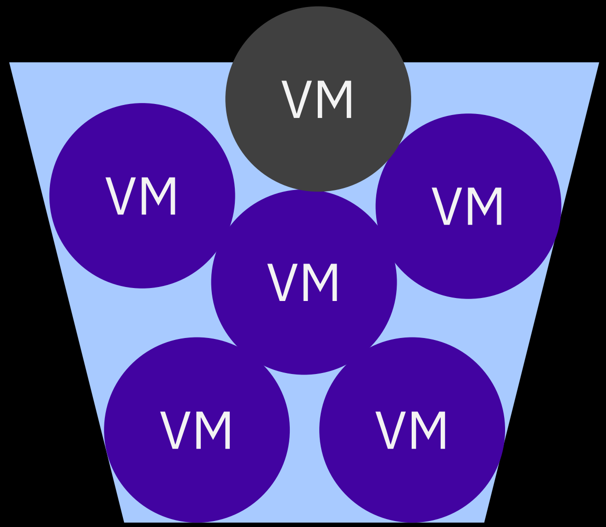


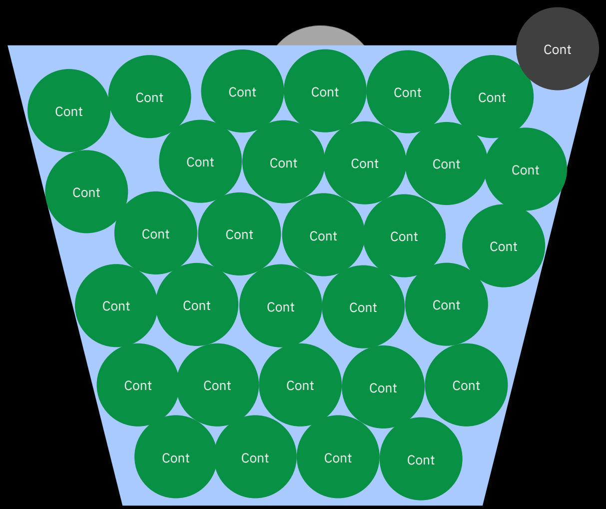


++ VM Isolation (OS)
- Complete OS
- Static Compute
- Static Memory

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

VMs vs. Containers





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

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

Advantages of Containers

Containers provide “just enough” isolation

Containers use hardware more efficiently

Containers are immutable

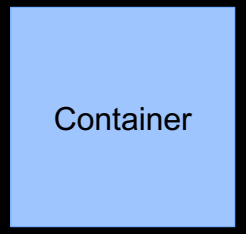
Containers are portable

Container Universe

Components

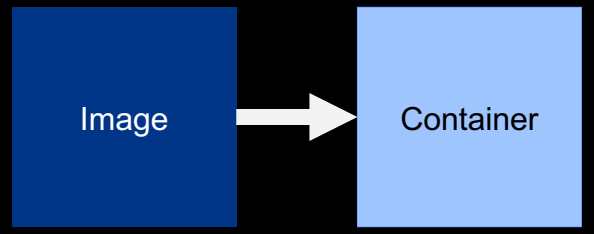
Container

Smallest compute unit



Components

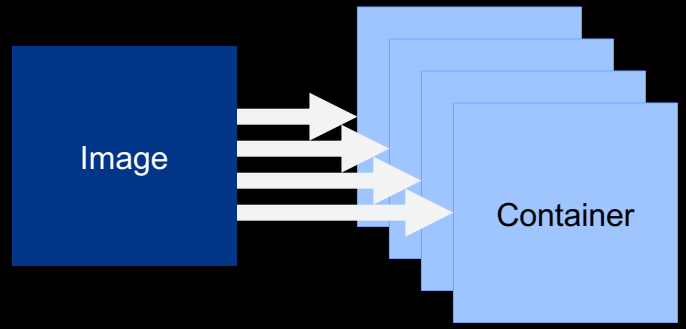
Containers
are created from
Images



one process per container

Components

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



Components

Images are versioned

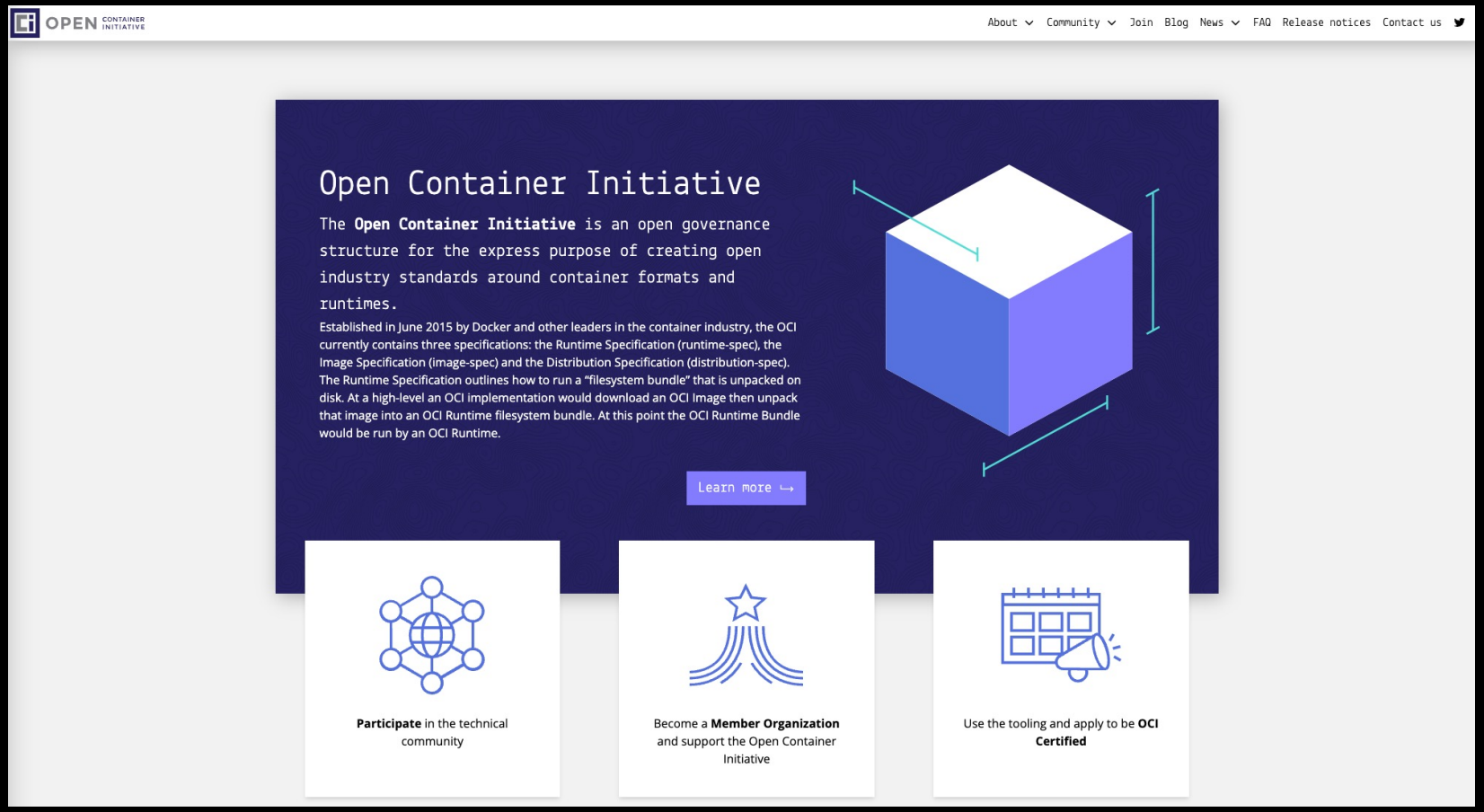
`webapp`:`v2.0`

Image Name — Image Tag

`latest` is a special tag and designates the latest built image. Should be used with caution to ensure immutability.

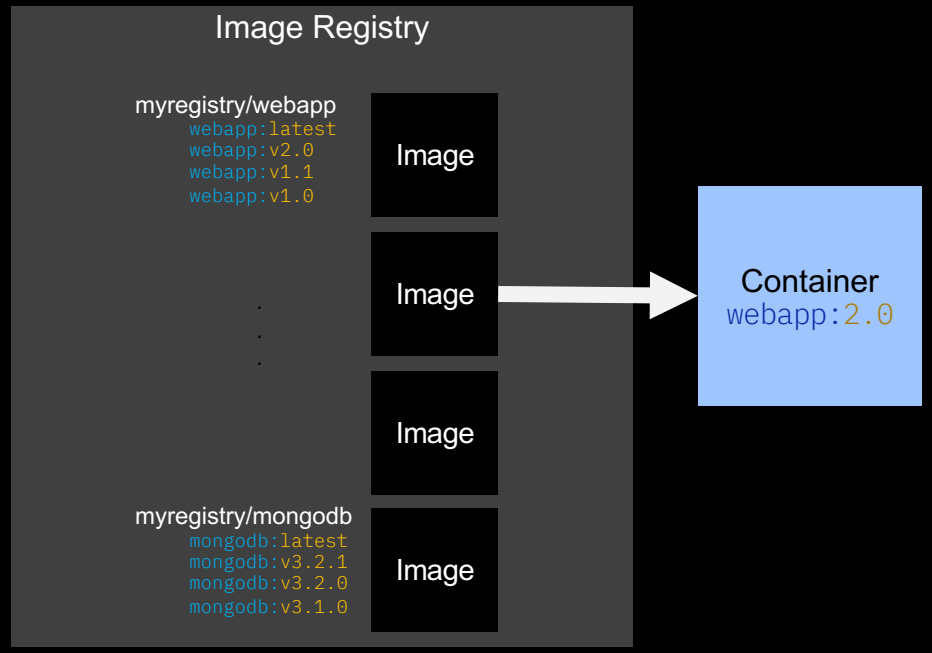
Components

Images are Open Container Initiative - **OCI Compliant**



Components

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



Components

Image

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

Container

The running, standard unit in which the application service resides or transported

Registry

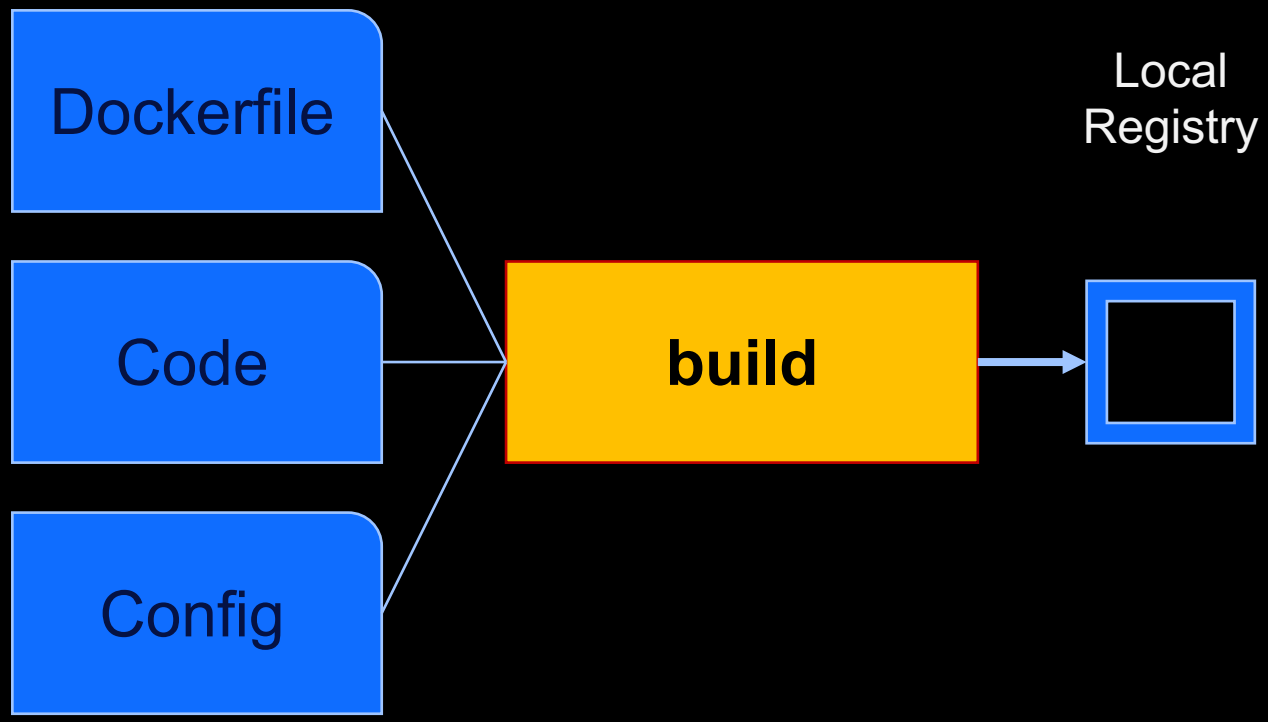
Available in SaaS or Enterprise to deploy anywhere you choose
Stores, distributes, and shares container images

~~Docker~~ Container 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

Create Images

Basics – Build



Basics – Build - Dockerfile

- Recipe - A text file that builds an image using 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/

# Expose the port of the web application
EXPOSE 80

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

Base Image from:

- hub.docker.com
- quay.io
- your own

A base image is the image that is used to create all of your container images.

Start with an empty image:

- FROM scratch

Or use an official image, such as:

- **Ubuntu**: This is one of the most downloaded Docker images in the industry.
- **UBI**: Red Hat Universal Base images allow commercial and open source developers to build containers based on RHEL
- **Alpine**: Extremely small, for systems with constrained resources
- **Nginx**: Tiny web and proxy server
- …

Basics – Build - Dockerfile

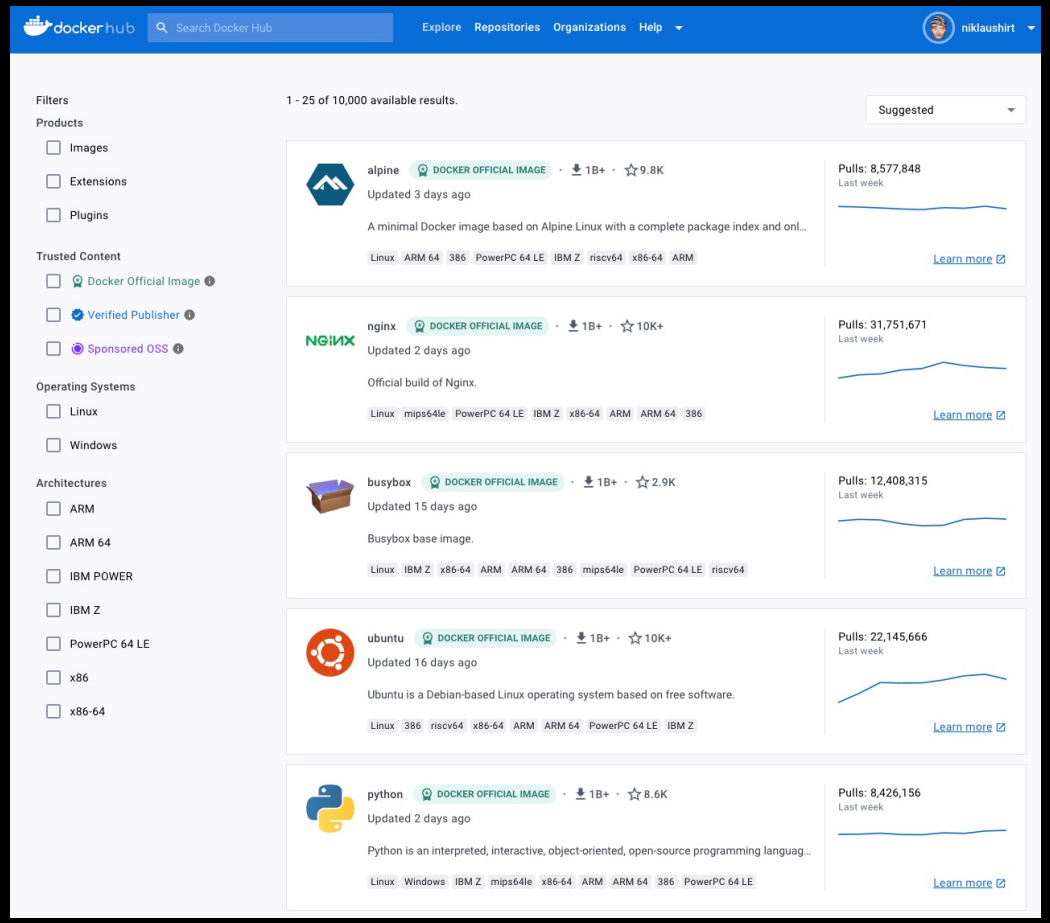
- Recipe - A text file that builds an image using directives

- 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/

# Expose the port of the web application
EXPOSE 80

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

Base Image from:

- hub.docker.com
- quay.io
- your own



Basics – Build - Dockerfile

- Recipe - A text file that builds an image using 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/

# Expose the port of the web application
EXPOSE 80

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

Base Image from:
- hub.docker.com
- quay.io
- your own

Run commands
apt-get, yum, chown, mv, cp, ...

Set environment variables
Can be overriden when running

Add assets to the image
Jar, sources, golang app, ...

Expose a port of the process running in the container

Startup command when image is being run

Basics – Build - Dockerfile – USER

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

# ---------------------------------------------
# Switch to root user (not really needed, you are root by default)
USER root

# Do stuff that requires root rights
# Add an unpriviledged user
RUN groupadd -r mygroup -g 433 && \
      useradd -u 431 -r -g mygroup -s /sbin/nologin myuser

# Do system stuff
RUN apt-get update && apt-get -y upgrade

# ---------------------------------------------
# Switch to non-root user
USER myuser

# Do stuff that doesn't require root rights
RUN apt-get update && apt-get -y upgrade
…

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

By default, containers run as root

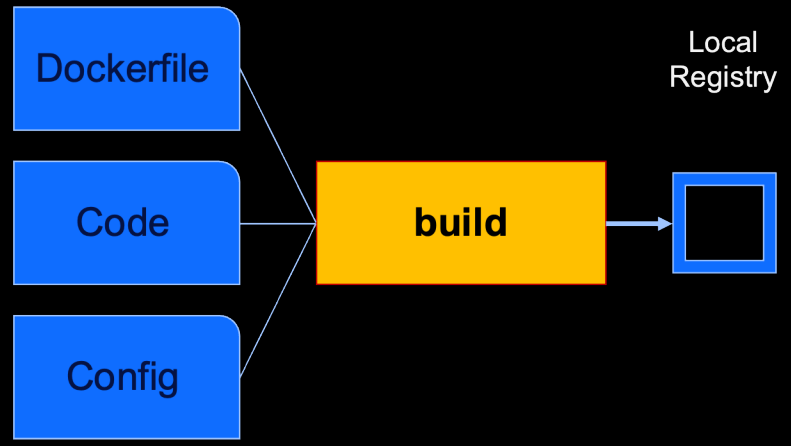
A container running as root has full control of the host system.
Requiring root can be dangerous for others and may not be available in all environments.

Use the USER instruction to specify a non-root user for containers to run as.
If your software does not create its own user, you can create a user and group in the Dockerfile.

Basics – Build - Dockerfile

```
❯ docker/podman build -t myimage:1.0.0 .
```

Dockerfile




```
❯ podman build -t myimage:1.0.0 .
STEP 1/6: FROM tomcat:8-jre8
STEP 2/6: MAINTAINER "youremailaddress"
--> Using cache 360f554b3a76949389ffd63ecfde40bff00fb1705b387406a3fab5294438de47
--> Pushing cache []:f5fbd8f90292956a6fcc07763609d24be78c762ec7eecf448fc094a93322701a
--> 360f554b3a7
STEP 3/6: RUN apt-get update && apt-get -y upgrade
--> Using cache f050872e53e599219ff94c6db56237e6a73369675c7765bc4c953ee5cdd481cf
--> Pushing cache []:96bc0a0287094c44b2251ea2b967e1bcdaae8d499ede59a1db773a0bb8902ccb
--> f050872e53e
STEP 4/6: ENV myName John Doe
--> Using cache 809a87c3671b47d172ebbb5c2d0ca9944006d95a867d31b12edba0fadc4e9880
--> Pushing cache []:005b263c3639ad8a692775141337cff489579082287a1c644679eea02bd7110f
--> 809a87c3671
STEP 5/6: EXPOSE 80
--> Pushing cache []:4fe61da890ff36b64a2cbe9f97929b80d715d938e1a65f04653c02276c1f1011
--> 52a988679c8
STEP 6/6: CMD ["catalina.sh", "run"]
COMMIT myimage:1.0.0
--> Pushing cache []:b2047af1e4593c24be782895a5878f76f1154e83f3be054cdb1a13f13f50419a
--> 9dcc2afafd2
Successfully tagged localhost/myimage:1.0.0
9dcc2afafd2a9fc97ed2daff4df34322f82ac2aabdfeed0ce31d7c0c90bf3b2c
```

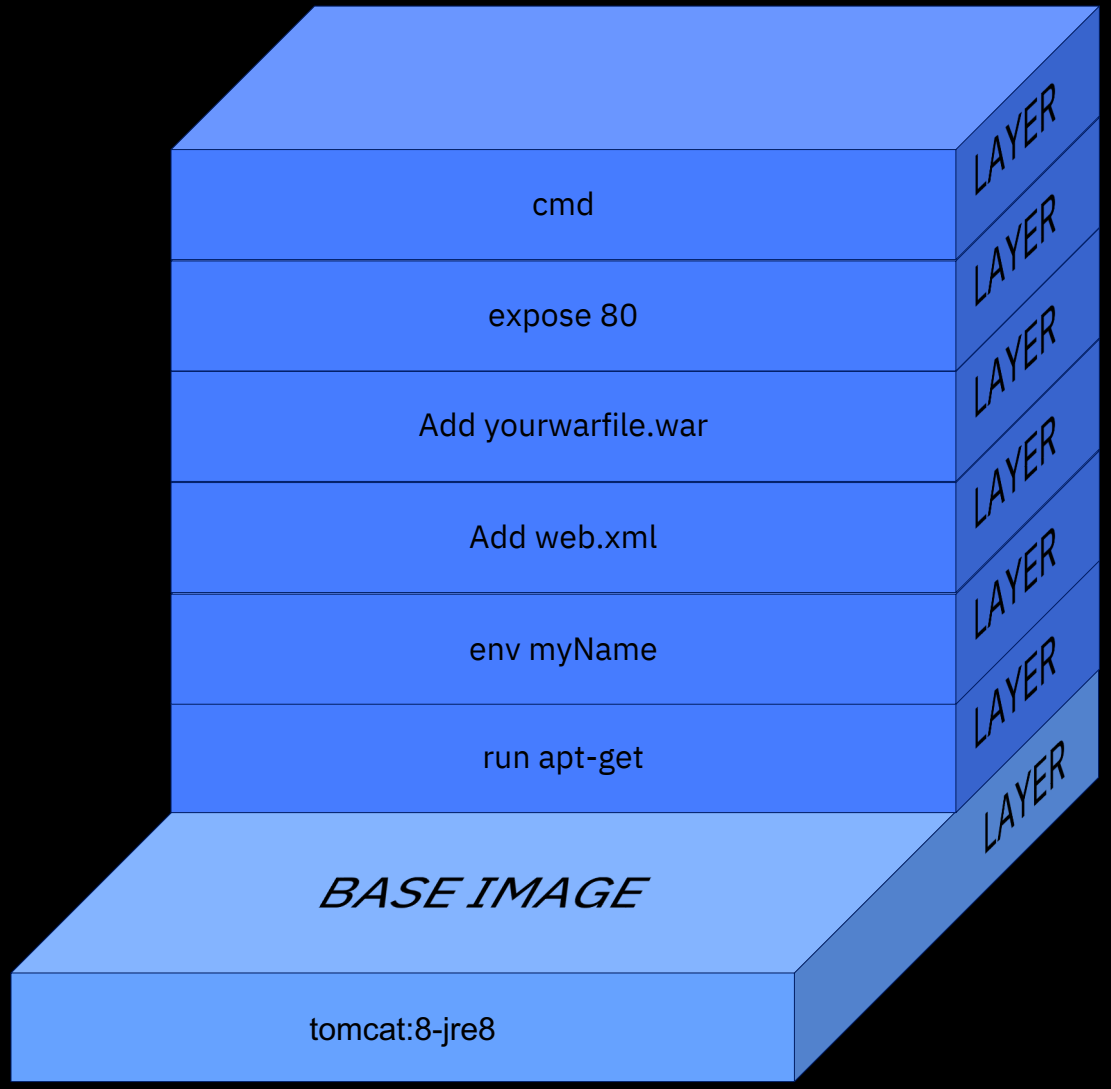
Images Filesystem

Basics – Layered File System

Layers
Inheritance

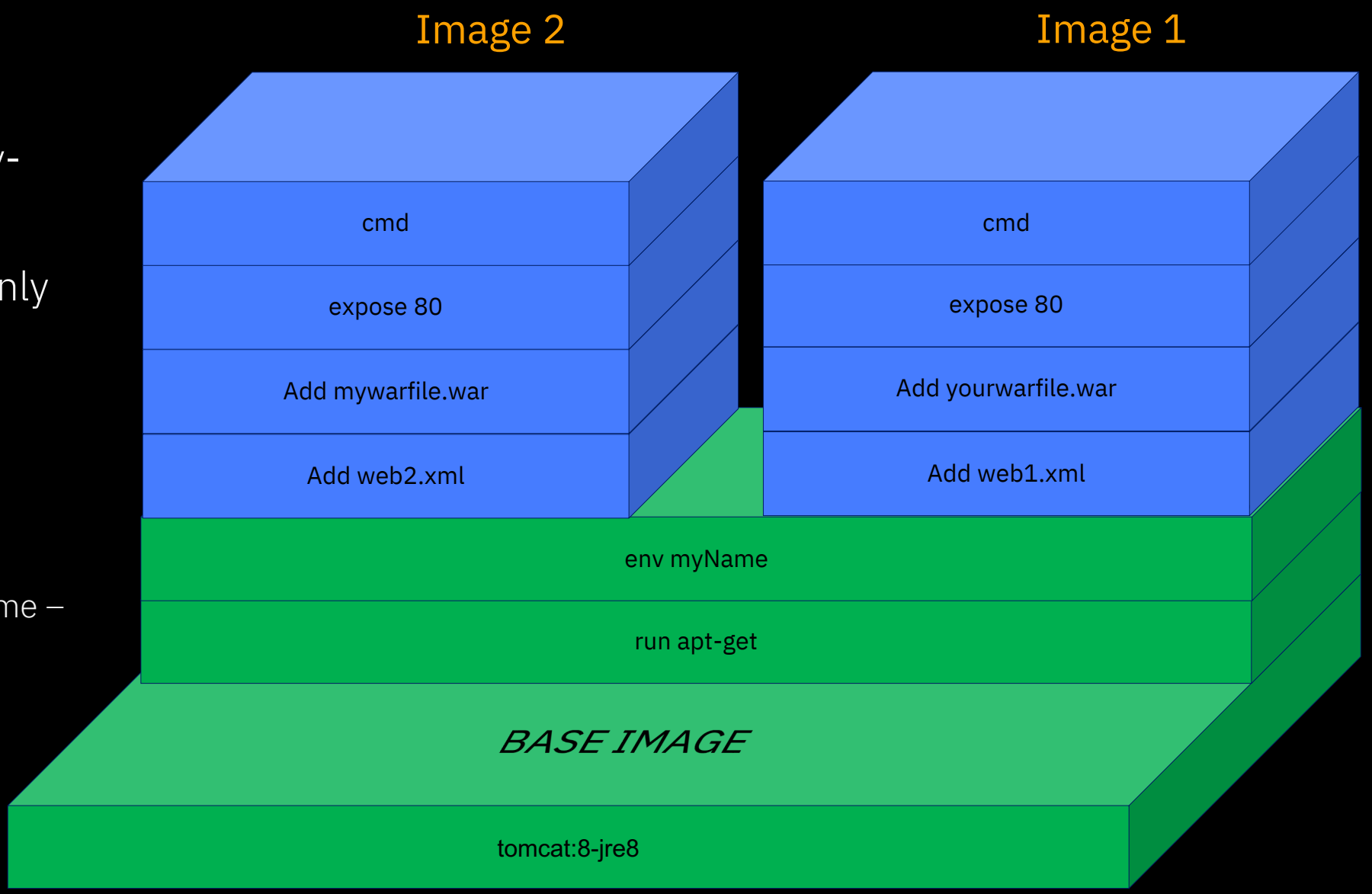
- Build on the work of those who came before you

```
# 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/

# Expose the port of the web application
EXPOSE 80

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



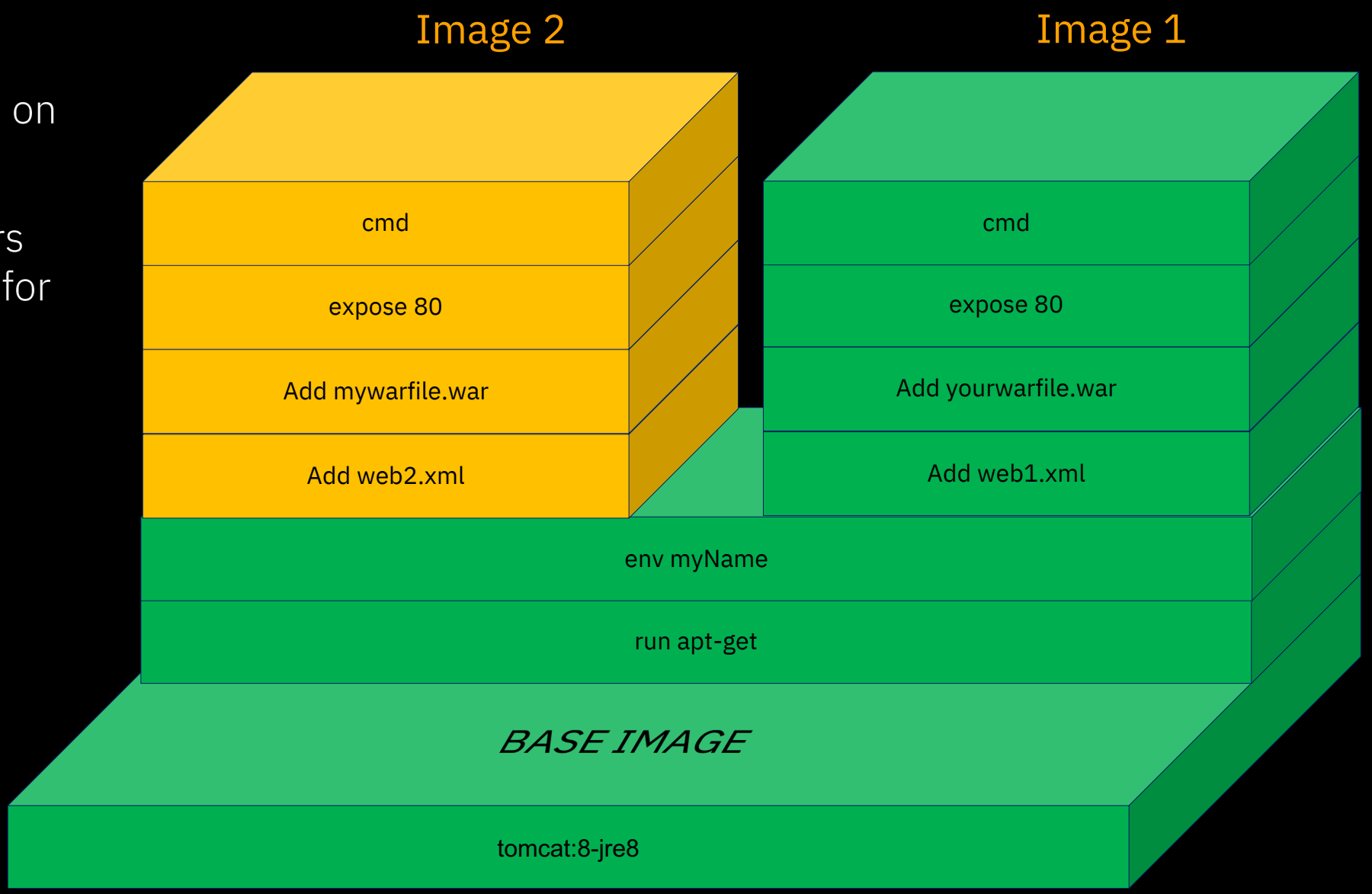
Basics – Layered File System

- Containers (OCI) use 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"



Basics – Layered File System

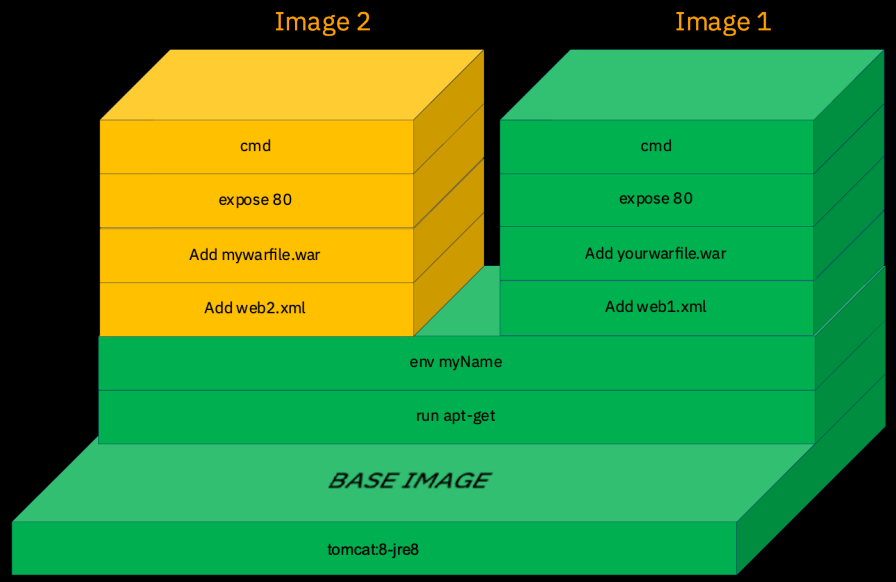
- If I already had Image 1 on my local system
- Only the additional layers need to be downloaded for Image 2



Basics – Build - Dockerfile

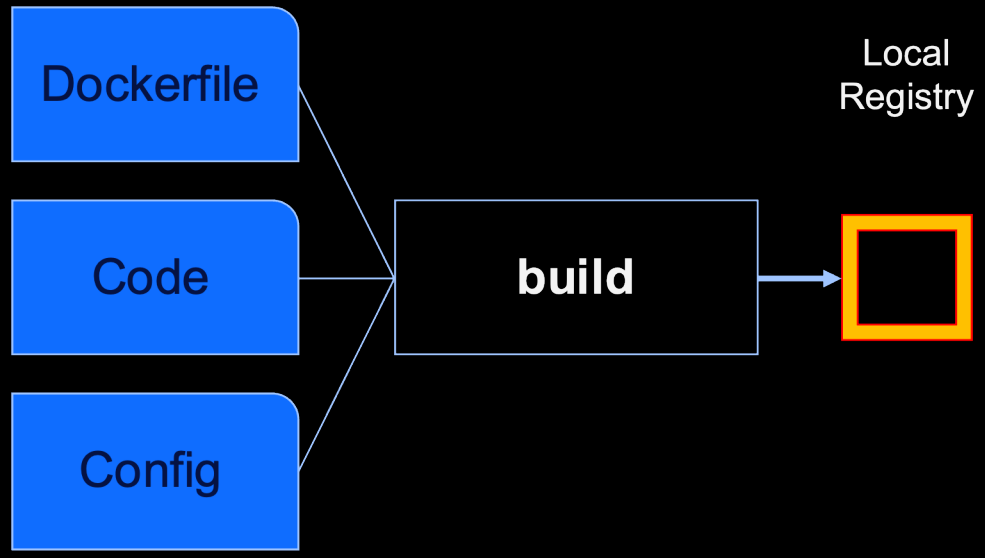
```
❯ podman build -t myimage:1.0.0 .
```

```
❯ podman build -t myimage:1.0.0 .
STEP 1/6: FROM tomcat:8-jre8
STEP 2/6: MAINTAINER "youremailaddress"
--> Using cache 360f554b3a76949389ffd63ecfde40bff00fb1705b387406a3fab5294438de47
--> Pushing cache []:f5fbd8f90292956a6fcc07763609d24be78c762ec7eecf448fc094a93322701a
--> 360f554b3a7
STEP 3/6: RUN apt-get update && apt-get -y upgrade
--> Using cache f050872e53e599219ff94c6db56237e6a73369675c7765bc4c953ee5cdd481cf
--> Pushing cache []:96bc0a0287094c44b2251ea2b967e1bcdaae8d499ede59a1db773a0bb8902ccb
--> f050872e53e
STEP 4/6: ENV myName John Doe
--> Using cache 809a87c3671b47d172ebbb5c2d0ca9944006d95a867d31b12edba0fadc4e9880
--> Pushing cache []:005b263c3639ad8a692775141337cff489579082287a1c644679eea02bd7110f
--> 809a87c3671
STEP 5/6: EXPOSE 80
--> Pushing cache []:4fe61da890ff36b64a2cbe9f97929b80d715d938e1a65f04653c02276c1f1011
--> 52a988679c8
STEP 6/6: CMD ["catalina.sh", "run"]
COMMIT myimage:1.0.0
--> Pushing cache []:b2047af1e4593c24be782895a5878f76f1154e83f3be054cdb1a13f13f50419a
--> 9dcc2afafd2
Successfully tagged localhost/myimage:1.0.0
9dcc2afafd2a9fc97ed2daff4df34322f82ac2aabdfeed0ce31d7c0c90bf3b2c
```



Basics – Build – Local Images

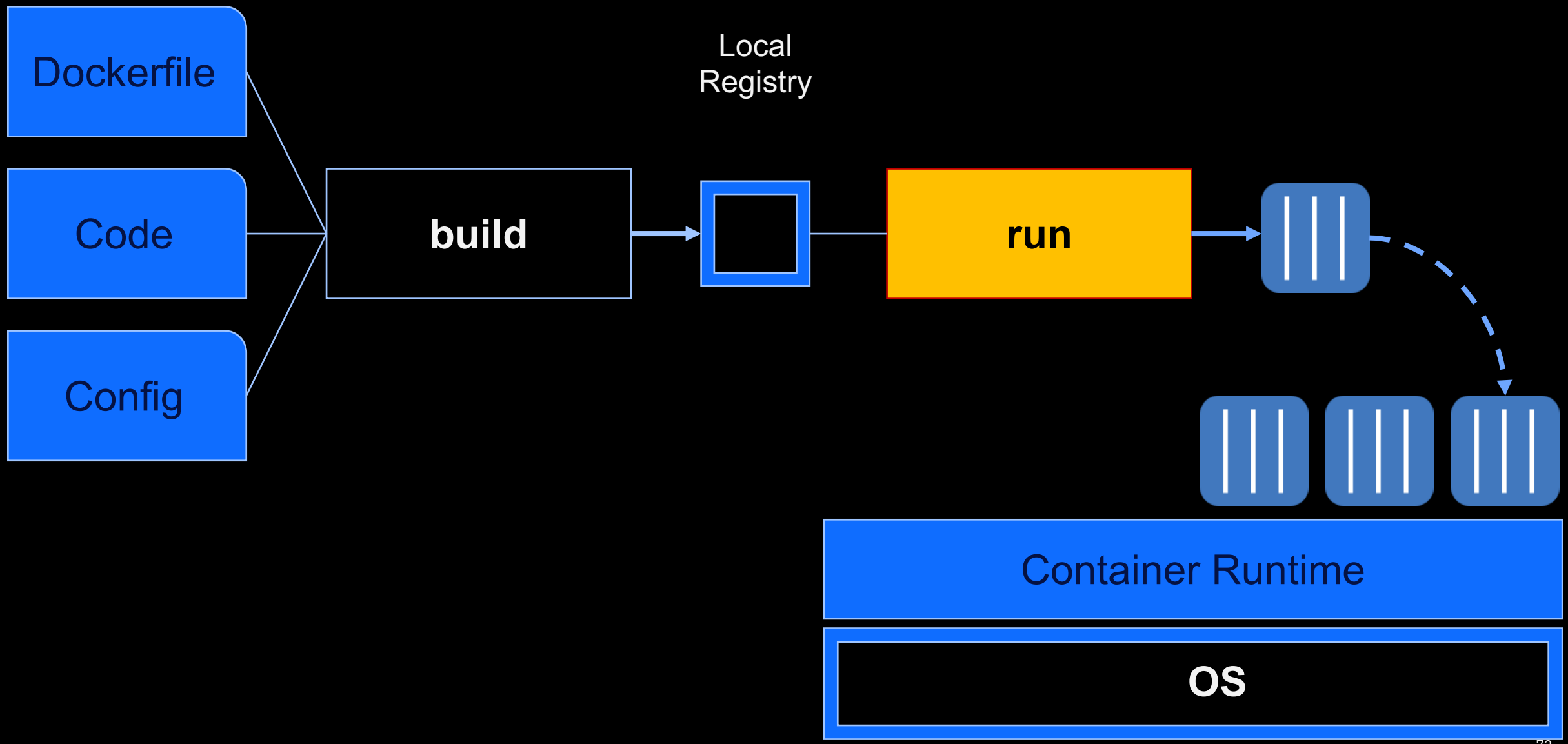
```
❯ podman images

❯ podman rmi 569118d95a3e
```




```
❯ podman images
REPOSITORY                      TAG         IMAGE ID      CREATED         SIZE
localhost/myimage               1.0.0       569118d95a3e  16 minutes ago  276 MB
localhost/niklaushirt/myimage   1.0.0       569118d95a3e  16 minutes ago  276 MB
localhost/k8sdemo-backend       lab         a1e30bbb2f45  41 minutes ago  1.36 GB
docker.io/library/tomcat        8-jre8      6802686d7abd  33 hours ago    235 MB
docker.io/library/node          8-stretch   50c57a9369c7  3 years ago     879 MB
```

Run Containers

Basics – Run

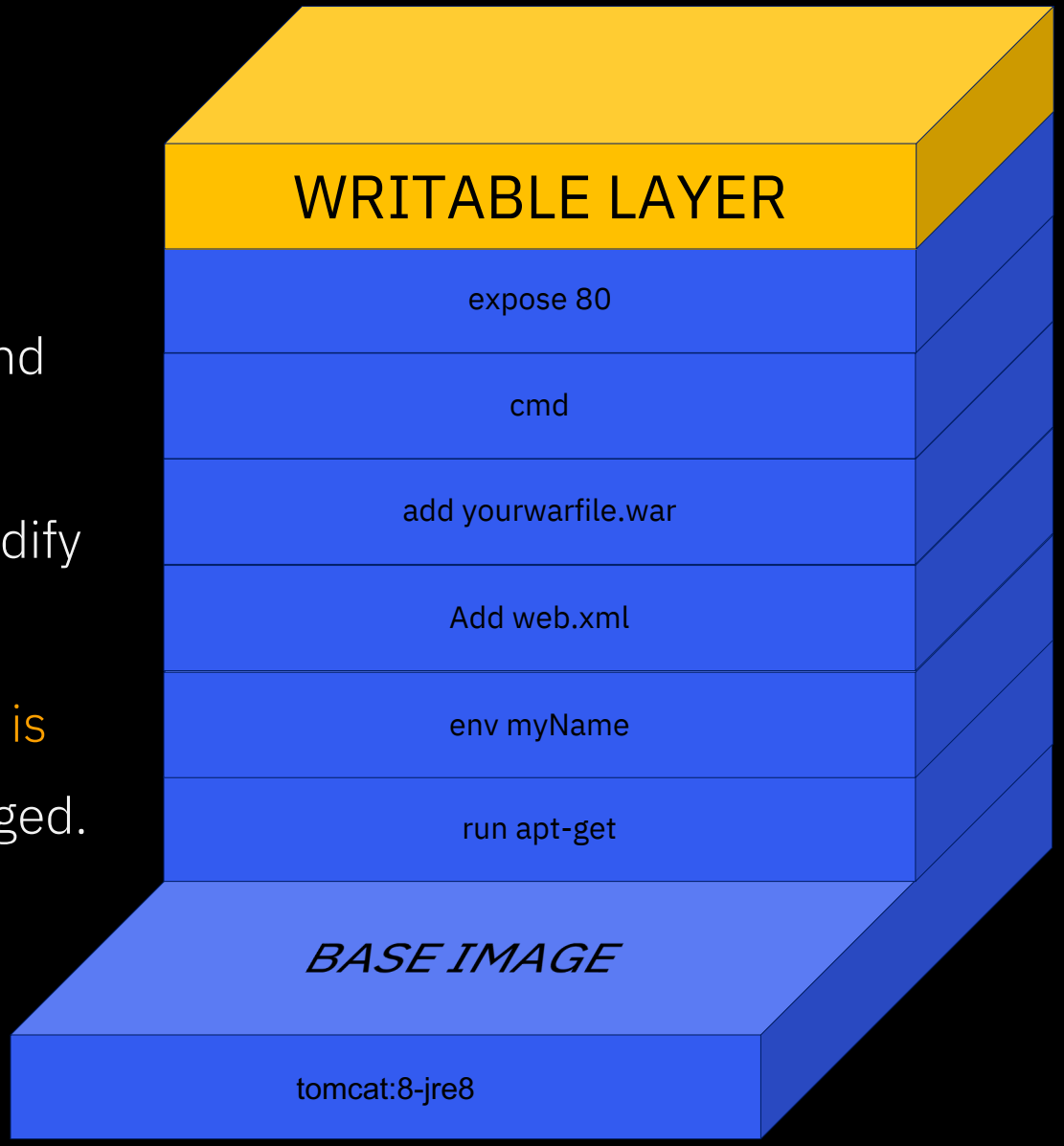

Dockerfile
Code
Config
build
Local
Registry
run
Container Runtime
OS

Basics – Run

Layers

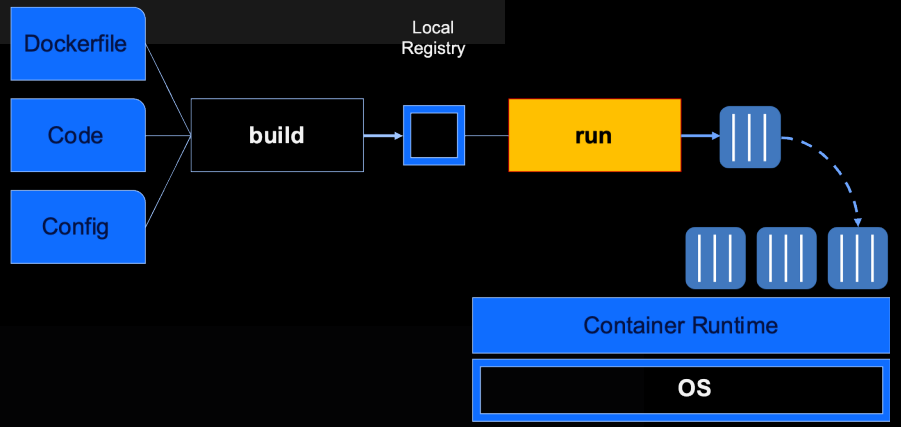
Inheritance

- 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.



Basics – Run

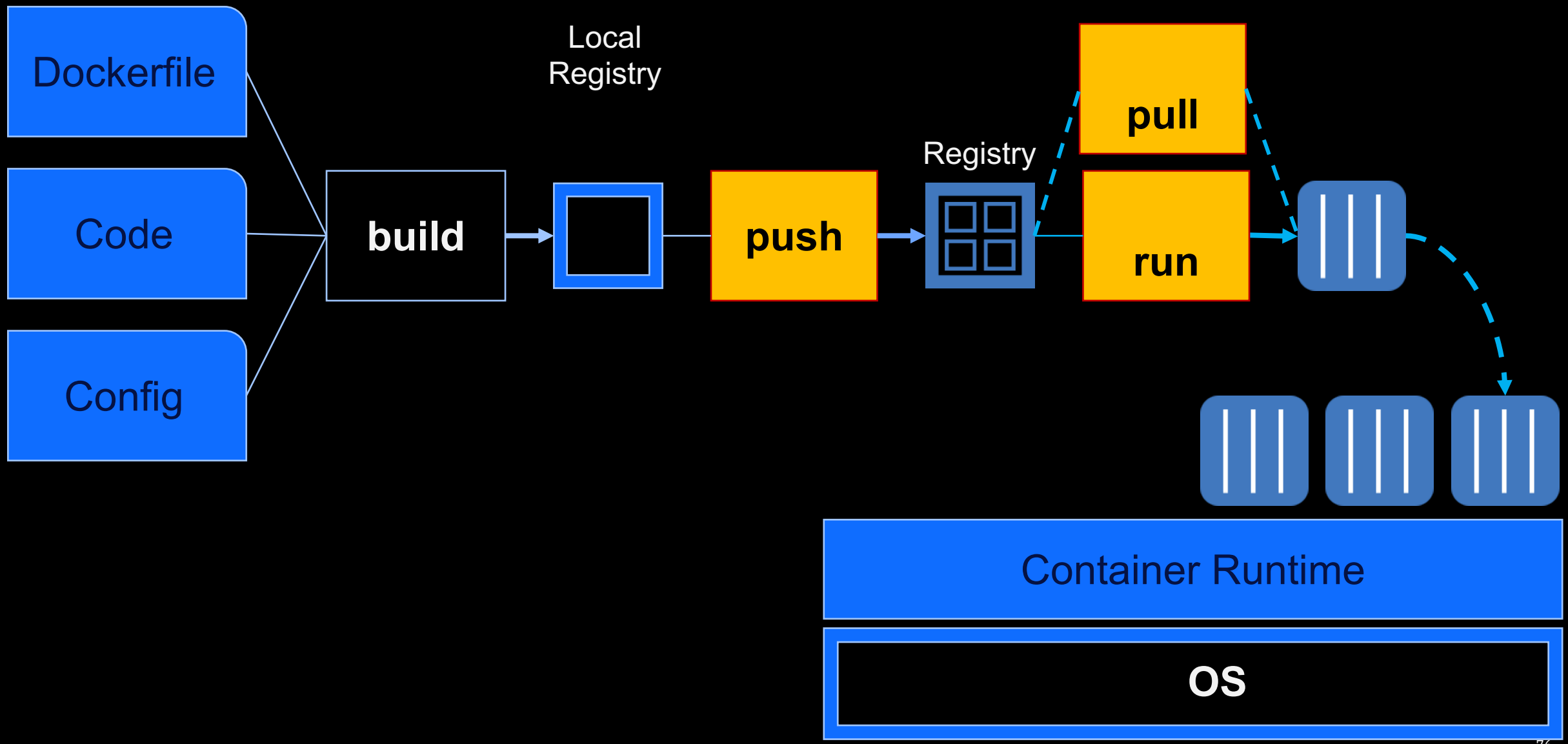
```
❯ podman run --rm -ti myimage:1.0.0 /bin/bash

❯ podman run --rm -d -e MYVAR=foo -p 8080:80 myimage:1.0.0

❯ podman ps [-a]

❯ podman stop/kill <CONTAINER_ID>
```




```
❯ podman run --rm -d --name myimage -p 8080:8080 myimage:1.0.0

b3812ee218eb64bde9c8cd06a4a8cf89b0c499af1a62004ccb109f0d831934a8
❯ podman ps

CONTAINER ID  IMAGE                     COMMAND          CREATED         STATUS         PORTS                   NAMES
b3812ee218eb  localhost/myimage:1.0.0  catalina.sh run  4 seconds ago  Up 4 seconds  0.0.0.0:8080->8080/tcp  myimage
❯ podman kill b3812ee218eb
b3812ee218eb
```

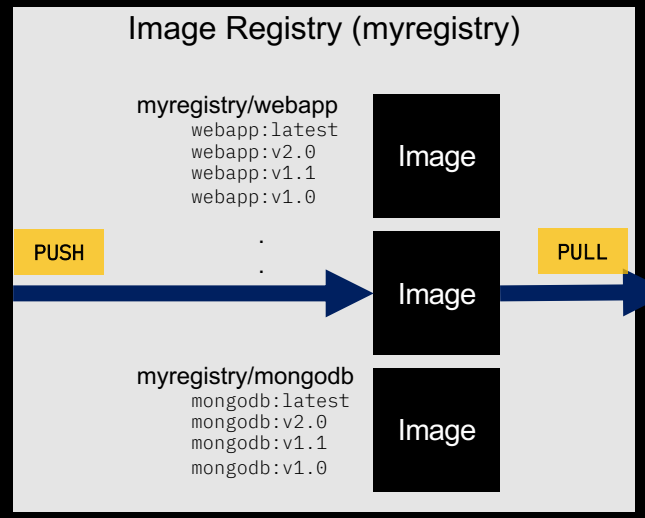
Store Images

Basics – Store, Retrieve & Run with registry



Basics – Store & Retrieve with registry

```
❯ podman login docker.io/quay.io
❯ podman tag myimage:1.0.0 myregistry/myimage:1.0.0
❯ podman push myregistry/myimage:1.0.0

❯ podman pull myregistry/myimage:latest
```




```
❯ podman login docker.io
Authenticating with existing credentials for docker.io
Existing credentials are valid. Already logged in to docker.io
❯ podman tag myimage:1.0.0 niklaushirt/myimage:1.0.0
❯ podman images
REPOSITORY                      TAG         IMAGE ID      CREATED         SIZE
localhost/myimage               1.0.0       569118d95a3e  20 minutes ago  276 MB
localhost/niklaushirt/myimage   1.0.0       569118d95a3e  20 minutes ago  276 MB
localhost/k8sdemo-backend       lab         a1e30bbb2f45  46 minutes ago  1.36 GB
docker.io/library/tomcat        8-jre8      6802686d7abd  33 hours ago    235 MB
docker.io/library/node          8-stretch   50c57a9369c7  3 years ago     879 MB
❯ podman push niklaushirt/myimage:1.0.0
Getting image source signatures
Copying blob sha256:27951c26c895e6b2d302273f91369f3e7bd21430d5d6678ba7c322bc08fdfb60
Copying blob sha256:874b048c963ab55b06939c39d59303fb975d323822a4ea48a02ac8dc635ea371
```

Basics – Registries

Hosting image repositories

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

Public and Private registries

- Public Registry like Quay or Docker Hub
- https://quay.io
- https://hub.docker.com

Login into the registry

- docker login domain:port
- podman login domain:port

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 containers
  - Manage containers when you get to have more than a handful

Control Groups

For limiting resources (cgroups)

+ Namespaces

For filesystem isolation (chroot)

= Containers

Notes from last course

Build images for other platforms (on M1 for example)

```
❯ podman buildx build --platform linux/amd64 -t myimage:x86 --load .
```

Build multi architecture images

```
❯ podman manifest rm myimage-mf

❯ podman build --platform linux/arm64/v8 --platform linux/amd64
        -t myimage:1.0.0 --manifest myimage-mf .

❯ podman manifest inspect myimage-mf
❯ podman manifest push myimage-mf myimage:1.0.0
```

Notes from last course

Run Portainer

```
❯ podman run -d --rm
  -p 8000:8000 -p 9001:9000 --name=portainer
  -v portainer_data:/data
  -v /run/podman/podman.sock:/var/run/docker.sock:Z
  --privileged
  portainer/portainer-ce:2.17.0-alpine
```

QUESTIONS?

Kubernetes Workshop Series

Kubernetes - *Basics*

04

What happened so far...
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.

Everybody Loves Containers

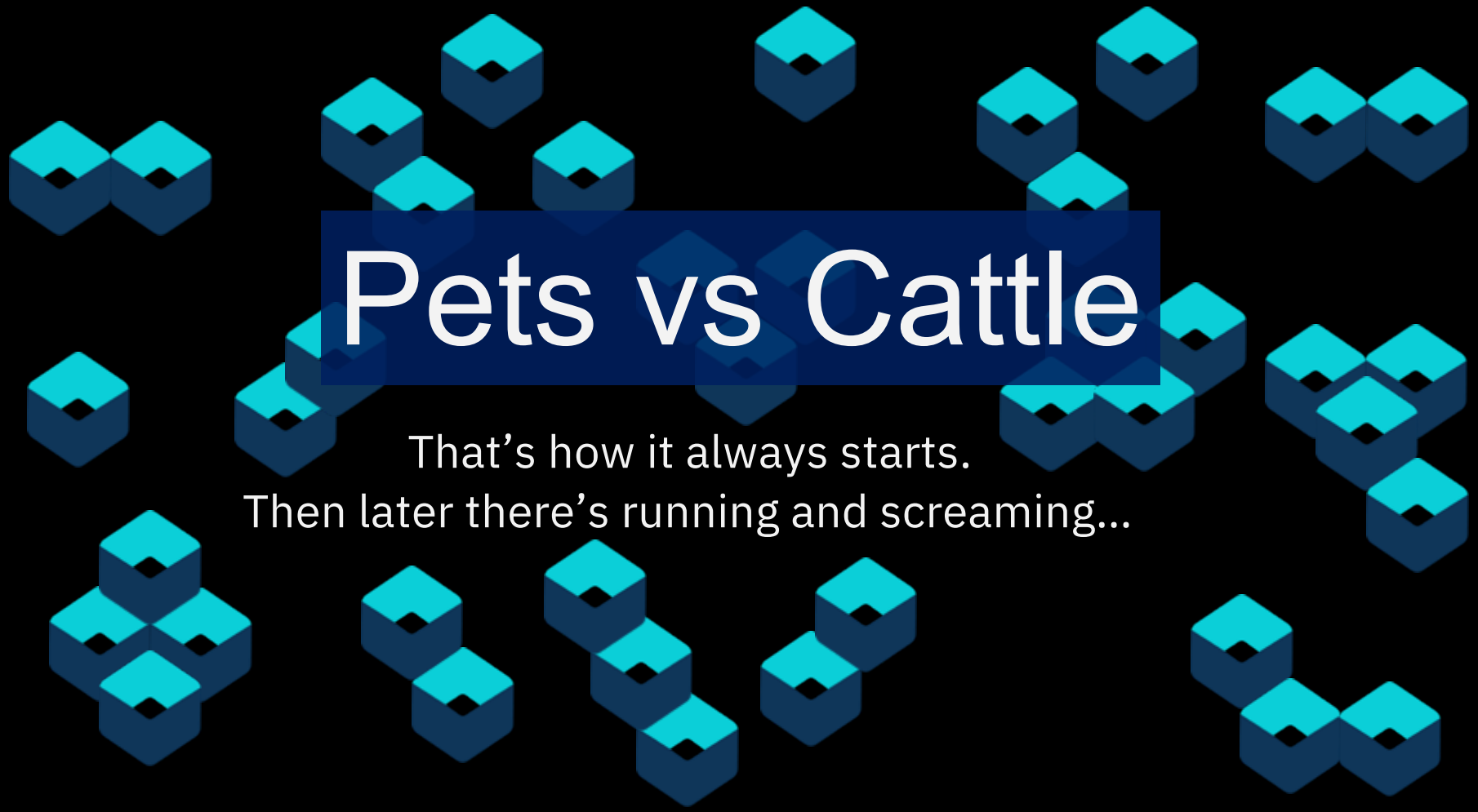
But when you go

From this....

To this....

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

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



Pets vs Cattle

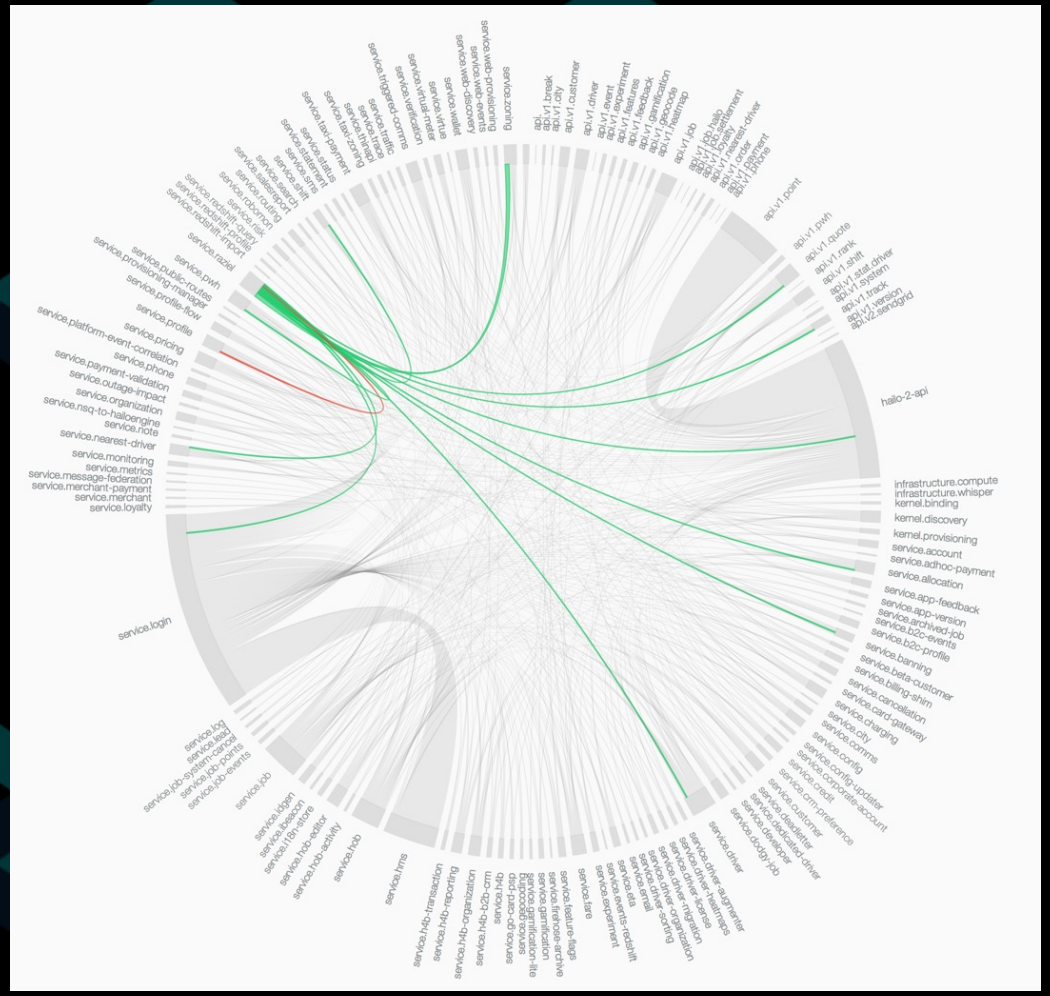
That’s how it always starts.
Then later there’s running and screaming...

The trade off

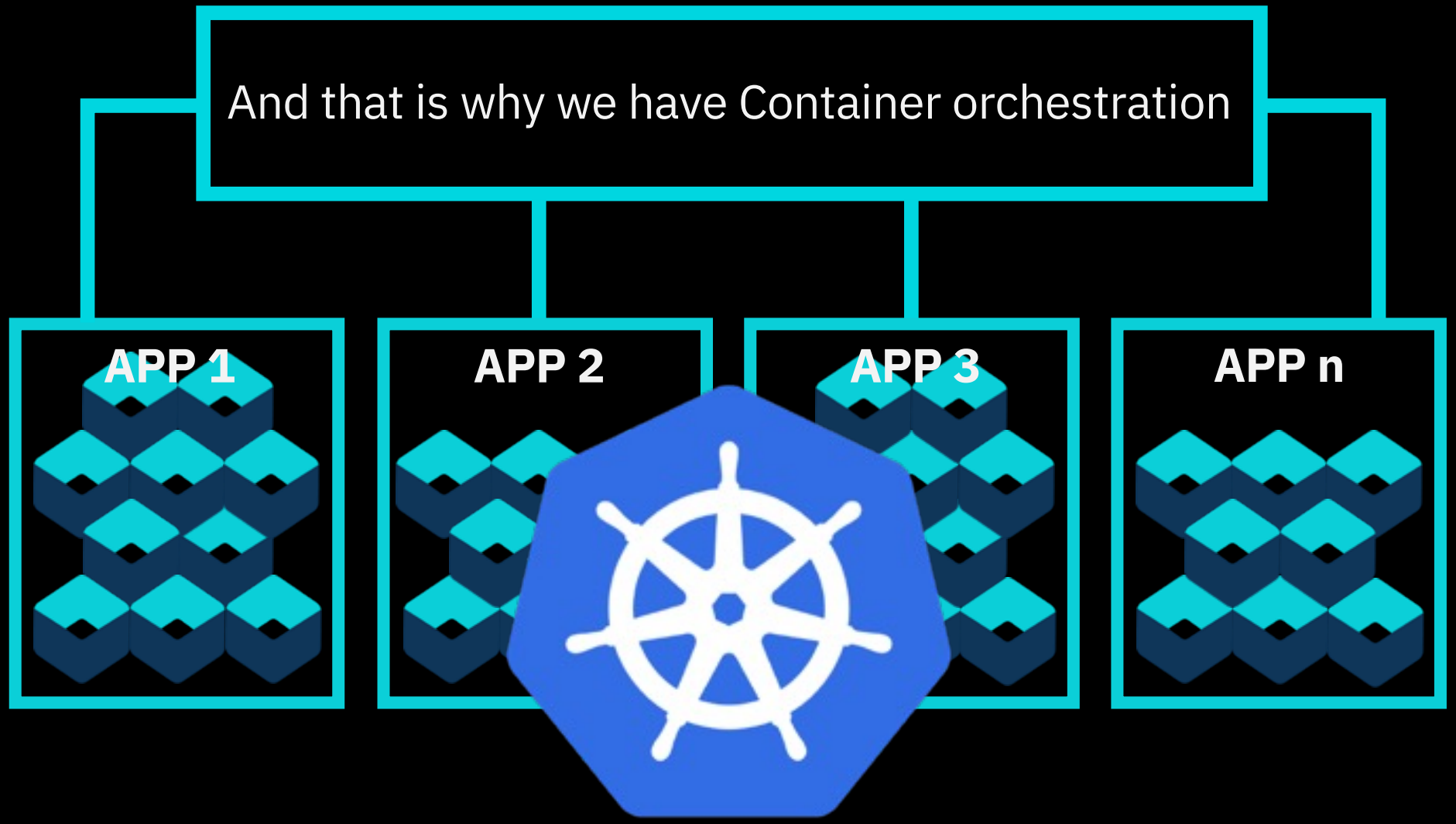
Improved delivery velocity

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.

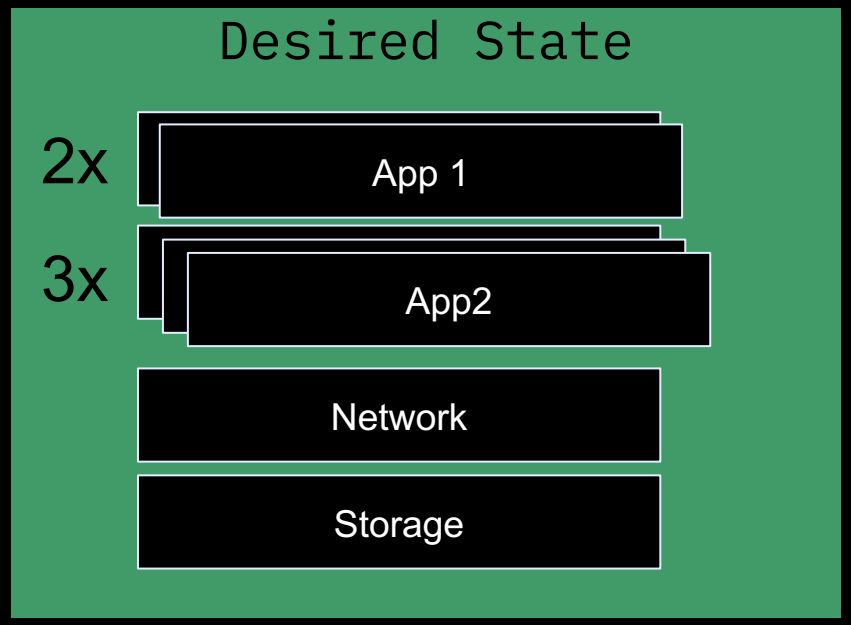
Declarative Systems

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.

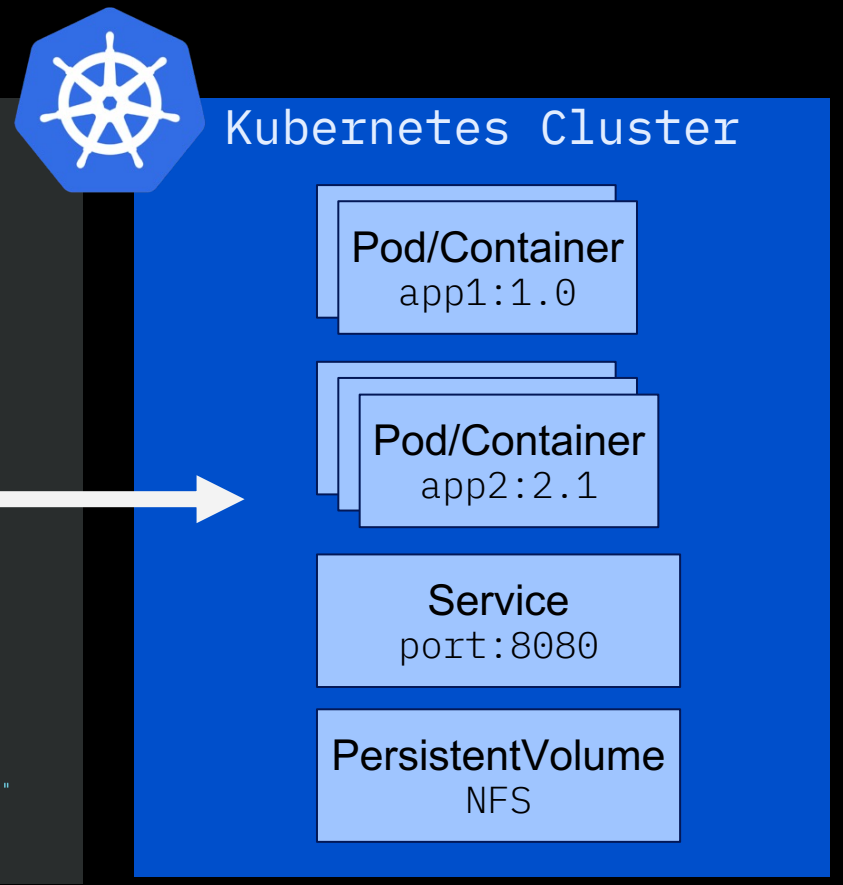
Kubernetes – Declarative System

The Desired State

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




```yaml
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: mcmk-ibm-mcmk-prod-klusterlet
  labels:
    app: ibm-mcmk-prod
    chart: ibm-mcmk-prod-3.1.2
    component: "klusterlet"
    release: mcmk
    heritage: Tiller
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ibm-mcmk-prod
      release: mcmk
  template:
    metadata:
      labels:
        app: ibm-mcmk-prod
        component: "klusterlet"
        release: mcmk
        heritage: Tiller
        chart: ibm-mcmk-prod-3.1.2
      annotations:
        productName: "IBM Multi-cloud Manager - Klusterlet"
        productID: "354b8990aab44c9988a0edfda101b128"
        productVersion: "3.1.2"
    spec:
```



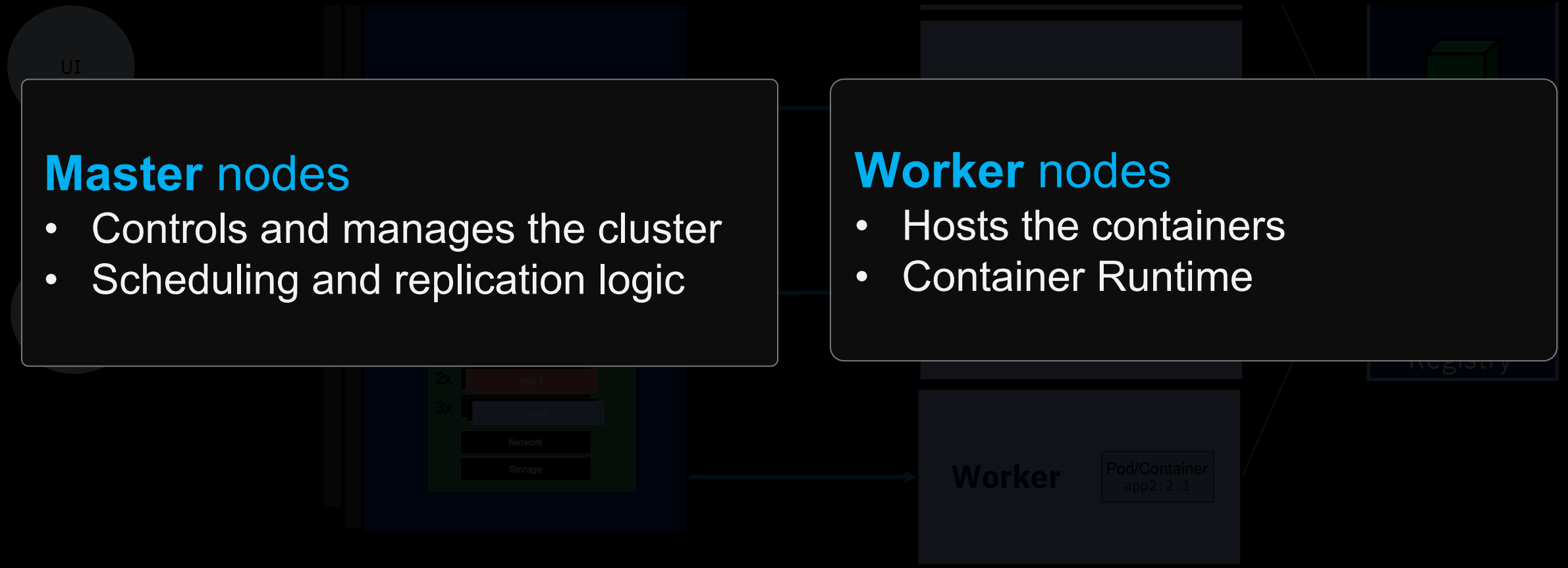
Kubernetes Strengths

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

K8s Architecture

Kubernetes Management Architecture

Nodes – hosts (Baremetal/VMs) that run Kubernetes applications



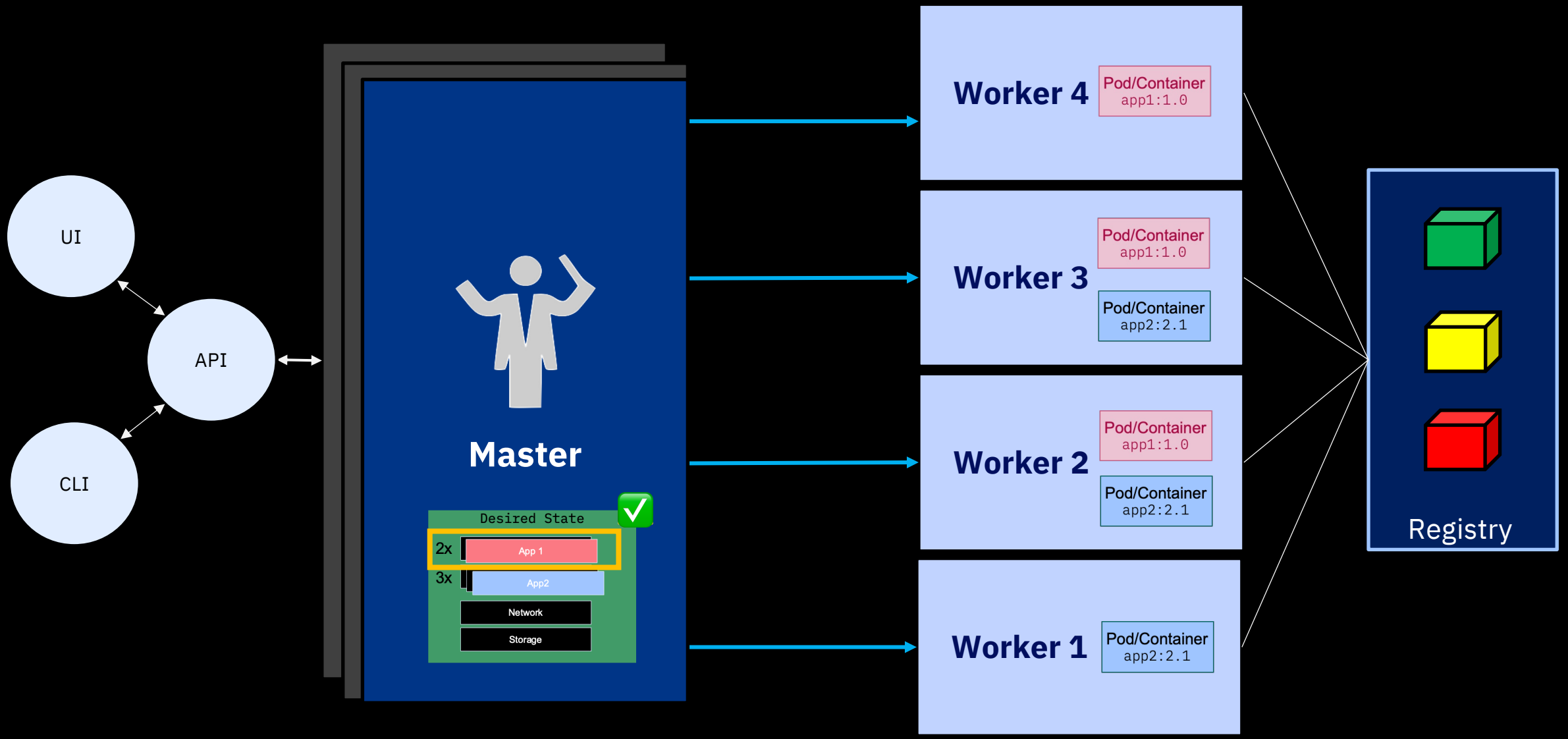
Master nodes

- Controls and manages the cluster
- Scheduling and replication logic

Worker nodes

- Hosts the containers
- Container Runtime

Kubernetes Management Architecture - Example



Kubernetes - Basic Objects

Deployment

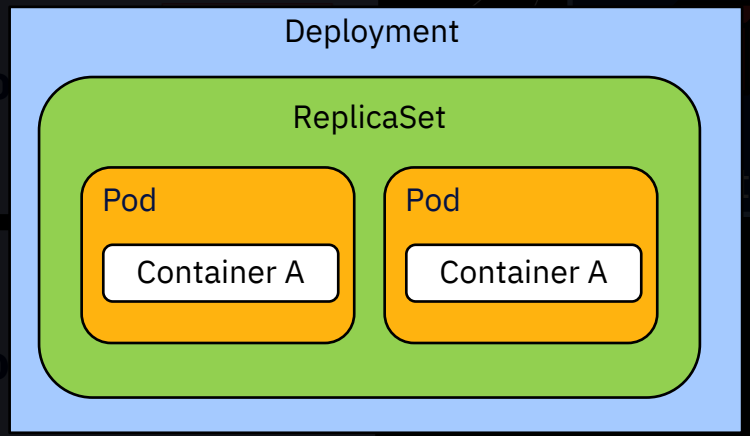
- A set of pods to be deployed together
- Declarative - creates a ReplicaSet describing the desired state
- Scale: A Deployment can be scaled

Pods

- Smallest deployment unit in K8s
- Collection of Containers that run on a worker node

ReplicaSet

- Ensures availability and scalability
- Maintains the number of Pods as requested by user



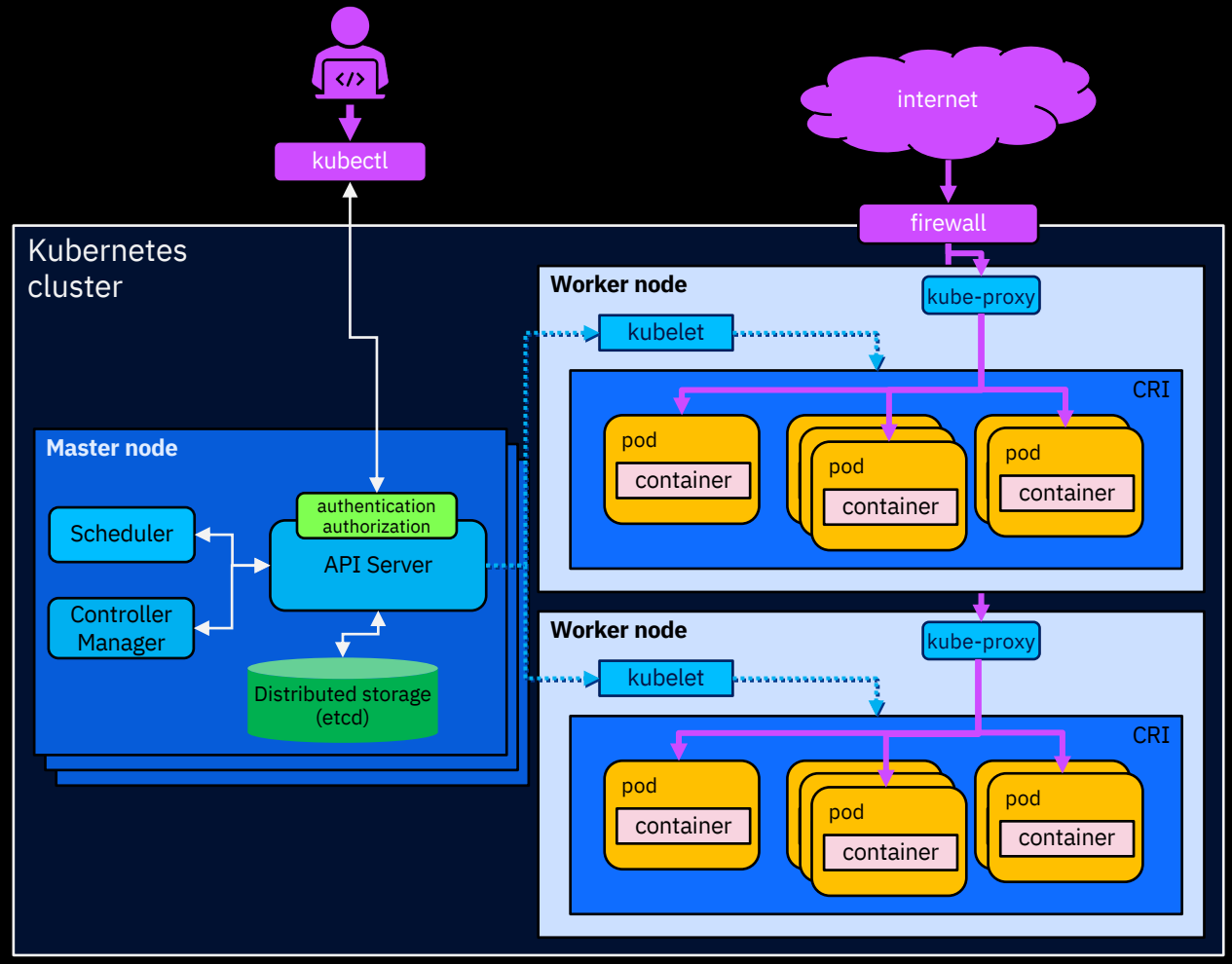
Kubernetes Cluster Architecture

Master node

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

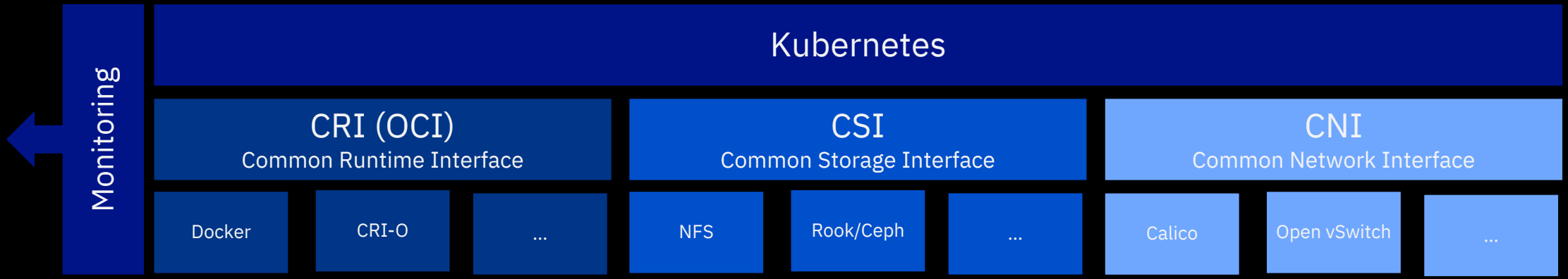
Worker nodes

- 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



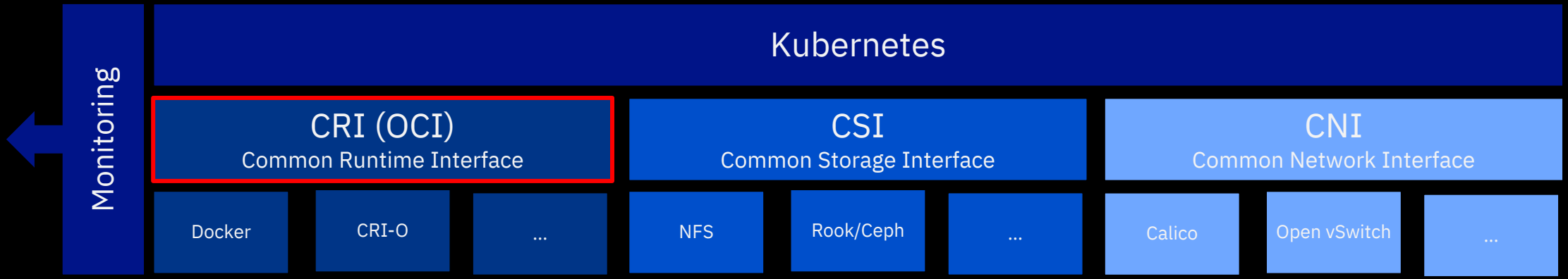
Kubernetes – Common Interfaces

CLOUD NATIVE COMPUTING FOUNDATION



Kubernetes – Common Interfaces

CLOUD NATIVE COMPUTING FOUNDATION



Container History

1979 chroot

2007 cgroups

2008 LXC

2011 warden

2012 docker

2014 kubernetes

2015 CLOUD NATIVE COMPUTING FOUNDATION

https://containerd.io/

2018 containerd

2019 cri-o

2020 docker + kubernetes

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 (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

Kubernetes – Why CRI-O?

cri-o

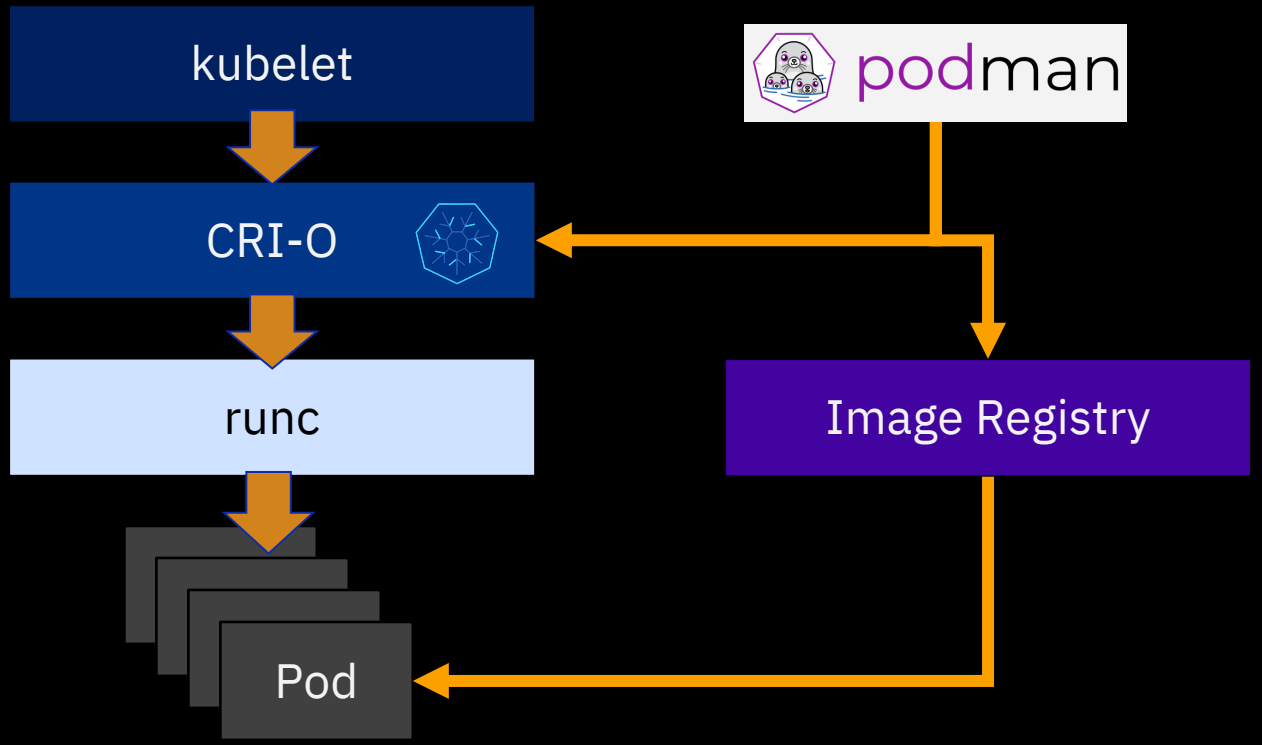
- **A Truly Open Project:** Operated as part of the broader Kubernetes community. Contributors from companies including Red Hat, SUSE, Intel, Google, Alibaba, IBM and more.

- **Lightweight:** CRI-O is made of lots of small components. In comparison, the Docker Engine is a heavyweight daemon which is communicated to using the docker CLI tool in a client/server fashion.

- **More Securable:** As CRI-O containers are children of the process that spawned it (not the daemon) they're fully compatible with tooling like cgroups & security constraints to provide an extra layer of protection to your containers. When using the Docker, every container is a 'child' of that large Docker Daemon. This complicates or outright prevents the use of this tooling.

- **Aligned with Kubernetes:** As an official Kubernetes project, CRI-O releases in lock step with Kubernetes, with similar version numbers. ie. CRI-O 1.11.x works with Kubernetes 1.11.x.

Kubernetes – CRI-O



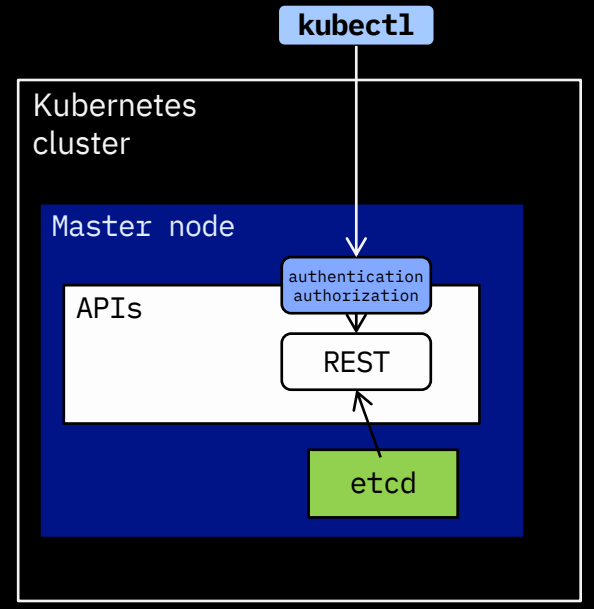
Accessing the Cluster

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.
It contains all the information needed to communicate with your cluster.

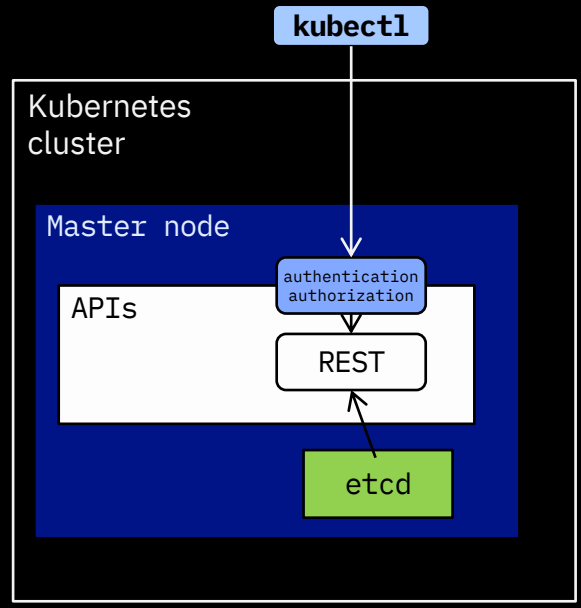
```
training@ubuntu:~/training/demo-app/k8sdemo$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /home/training/.minikube/ca.crt
    server: https://172.17.0.4:8443
  name: minikube
contexts:
- context:
    cluster: minikube
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: /home/training/.minikube/profiles/minikube/client.crt
    client-key: /home/training/.minikube/profiles/minikube/client.key
```

kubectl – talking to the Cluster

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

```
❯ 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
```

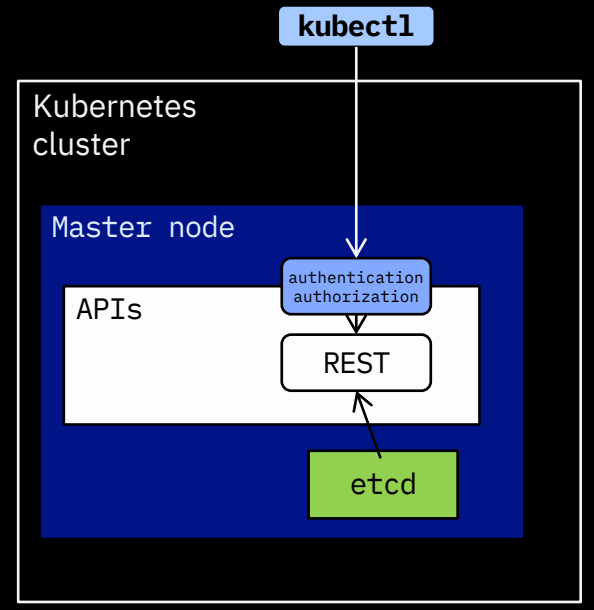


kubectl – talking to the Cluster

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

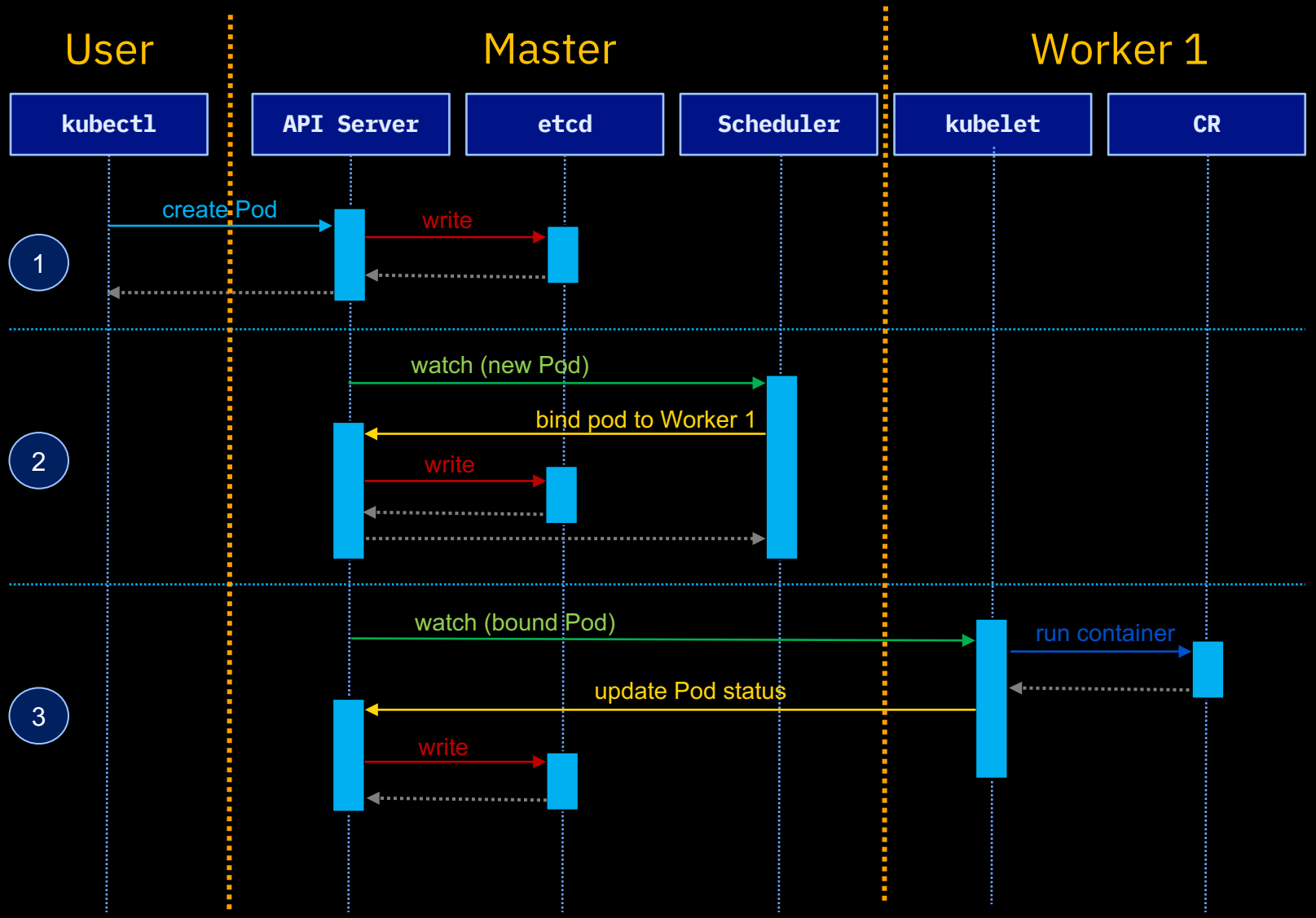
- Kubernetes uses etcd as a key-value database store.
- It stores the configuration of the Kubernetes cluster in etcd.
- It also stores the actual state of the system and the desired state of the system in etcd.

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



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

K8s Objects

Kubernetes Objects with YAML Manifests

`apiVersion: v1` **`kind:`** `Pod`	What kind of object you want to create

`metadata:`
`  name: nginx`
`  namespace: default`
`  labels:`
`  app: k8s-demo`
	Data that helps uniquely identify the object (Name, Namespace, Labels, ...)

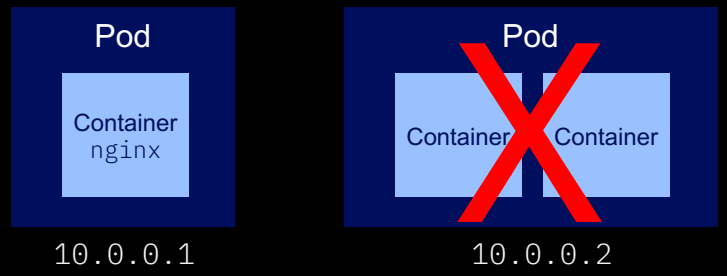
`spec:`
`  containers:`
`  - name: nginx`
`  image: nginx:1.7.9`
`  ports:`
`  - containerPort: 80`
`  env:`
`  - name: APPLICATION_NAME`
`  value: k8sdemo`
	Description of the characteristics you want the resource to have: its DESIRED STATE

`status:`
`  phase: Running`
`  hostIP: 192.168.252.150`
`  podIP: 10.131.1.51`
	Describes the CURRENT STATE of the object.
	This is generated once the object has been deployed.

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
  - IP Address can change when restarting the Pod



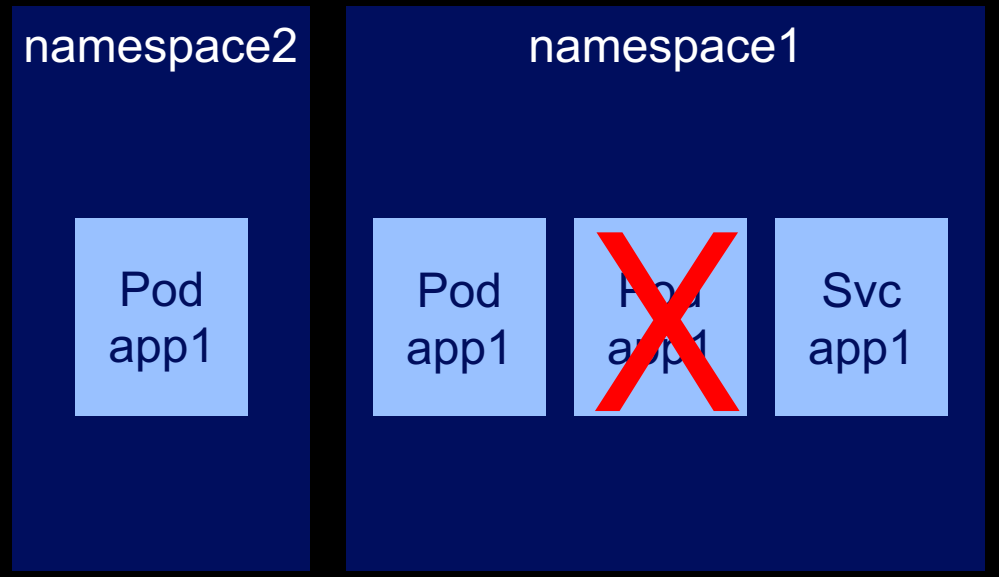
Naming and Structuring

Name

- Each resource object by type has a unique name

Namespace

- **Logical grouping:** group objects by application, tier, environment, …
  Example:
  - app1-frontend-prod
  - app2-datalayer-dev
- **Resource isolation**: Each namespace is a virtual cluster within the physical cluster
  - Resource objects are scoped within namespaces
  - Names of resources need to be unique within a namespace, but not across namespaces
  - Low-level resources are not in namespaces: nodes, persistent volumes, and namespaces themselves
- **Resource quotas**: Namespaces can divide cluster resources
- **Network isolation**: Namespaces can isolate 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
  - `openshift-xxx` – The namespaces for objects created by OpenShift

Creating a Pod

```
❯ kubectl run nginx --image= nginx:1.7.9

Or

❯ kubectl create -f example.yaml
```

Imperative

Declarative

example.yaml

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

Creating a Pod

```
❯ kubectl create -f example.yaml
```

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

❯ kubectl get pods nginx-5bd87f76c-vxc79 -o yaml
apiVersion: v1
kind: Pod
metadata:
…
status:
…
```

example.yaml

```
apiVersion: v1
kind: Pod                          # kind
metadata:
  name: nginx                      # id
  namespace: default
spec:                              # desired state
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```

Deleting a Pod

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

❯ kubectl delete pods nginx-5bd87f76c-vxc79
```

```
❯ kubectl delete –f example.yaml
```

example.yaml

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

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

```
❯ kubectl get pods -l release=april2023
NAME                     READY   STATUS    RESTARTS   AGE
nginx-5bd87f76c-vxc79    1/1     Running   0          29s
```

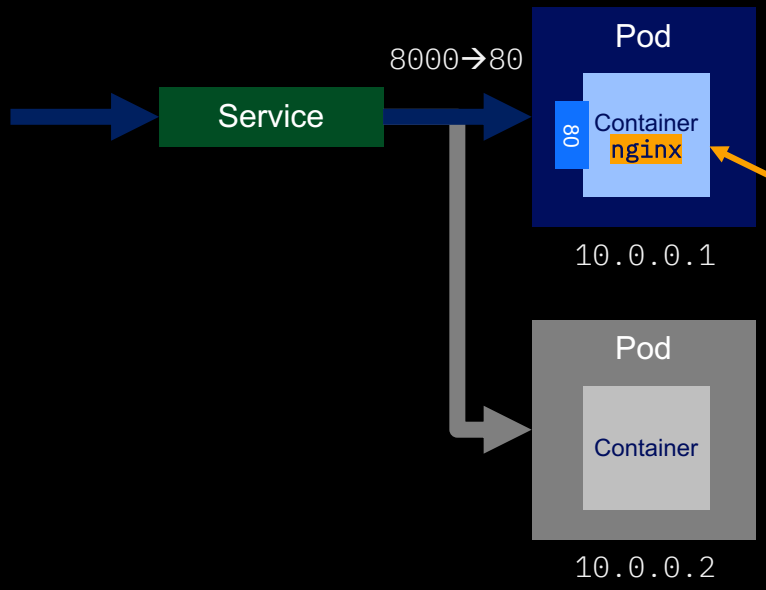
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: default
  labels:
    app: my-nginx
    release: april2023
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```

Basic Networking

Service

- A service provides a way to refer to a set of Pods (selected by labels) with a single static IP address.
- Creates an entry in the Kubernetes DNS

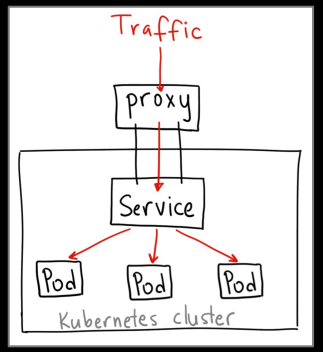
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: my-nginx
    release: april2023
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```



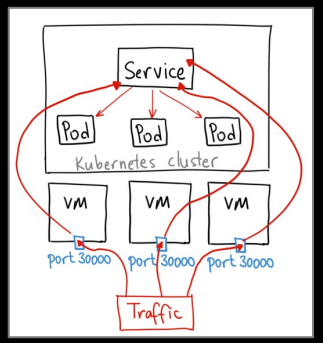

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
  selector:
    app: my-nginx
```

Service

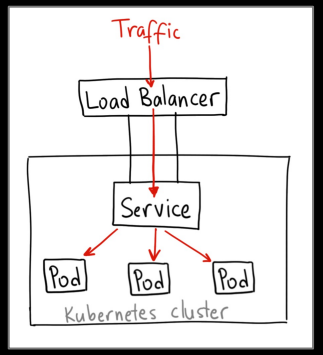
ClusterIP (default): This type exposes the service on the cluster internal IP. This means that the service is only reachable from within the cluster.



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



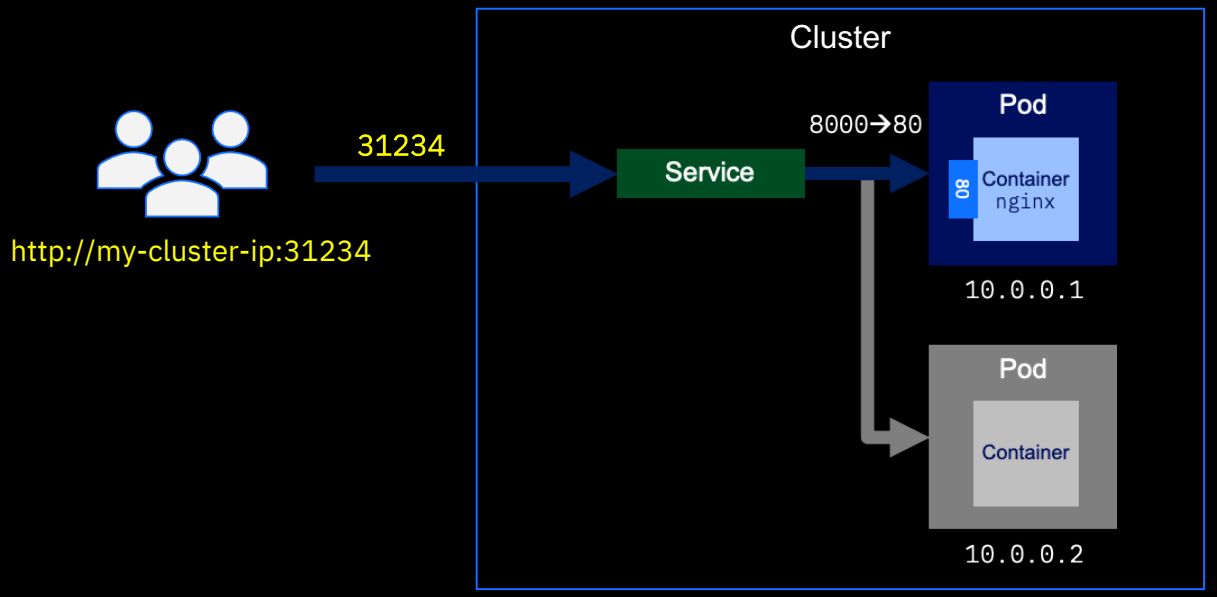
LoadBalancer: This service type exposes a service using a provided external load balancer.



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:
  type: NodePort
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
  selector:
    app: nginx
```

Service Access

So, a Kubernetes service with the name ***k8s-demo-service*** in namespace ***default*** could be addressed:

When referred from the same namespace

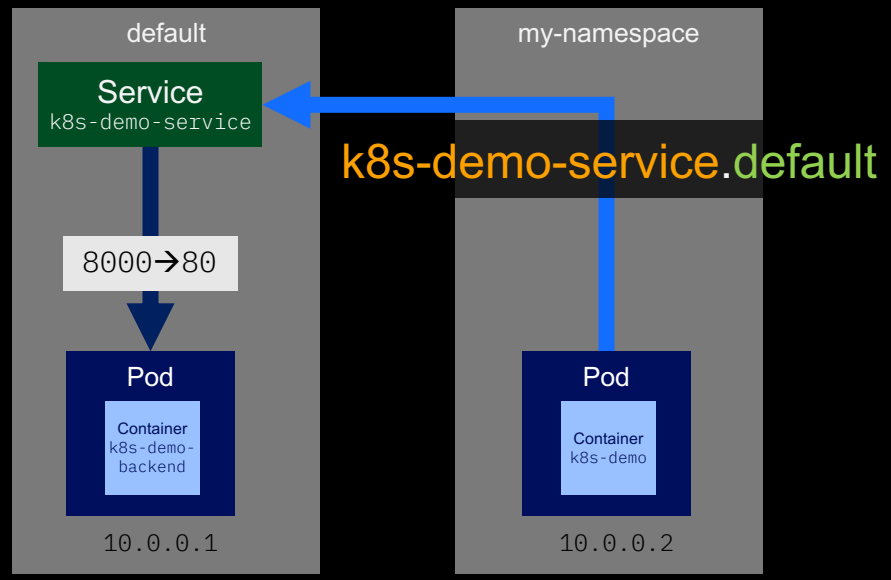
k8s-demo-service

When referred from another namespace

k8s-demo-service.default

Fully qualified name

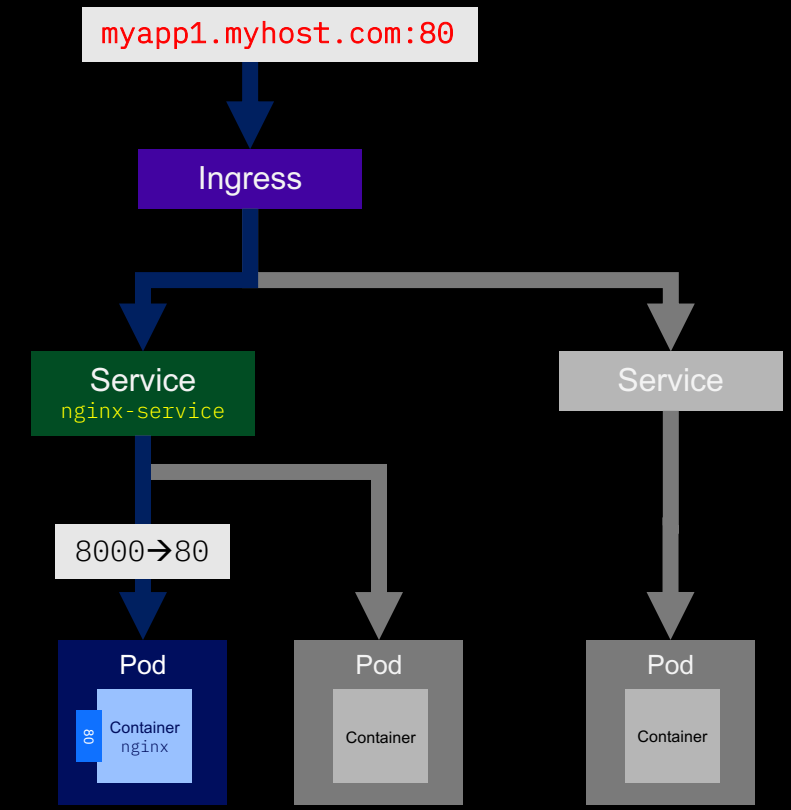
k8s-demo-service.default.svc.cluster.local



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 an additional Loadbalancer.

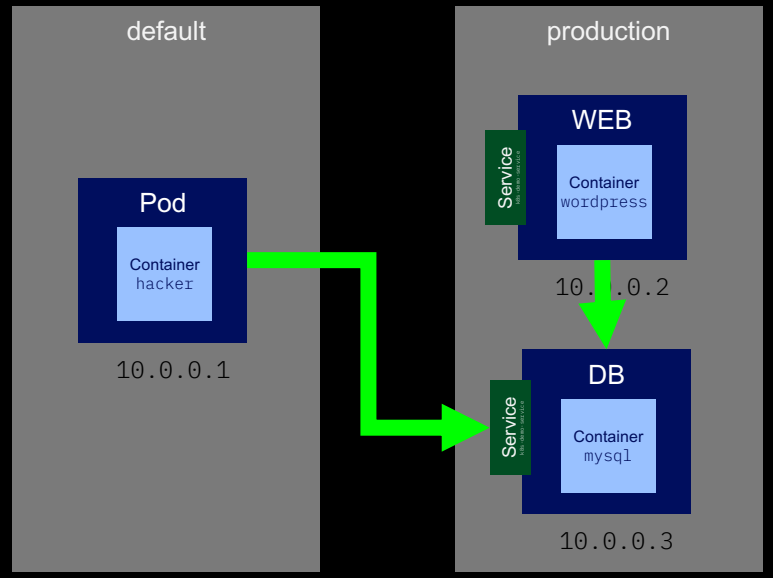
```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: myapp1.myhost.com
    http:
      paths:
      - backend:
          serviceName: nginx-service
          servicePort: 80
```



Securing Service Access

NetworkPolicies allow you to specify how a pod is allowed to communicate with "endpoints" and "services" over the network.

⚠ By default, Pods can access ALL other Pods.

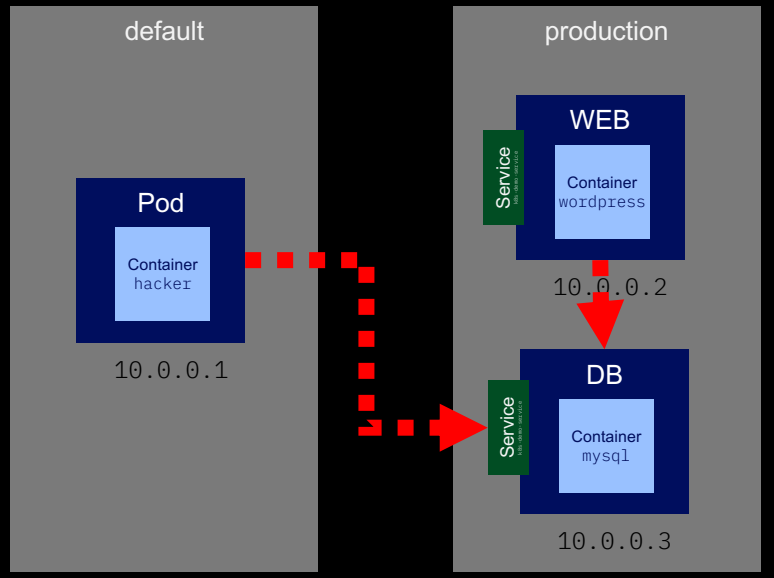


Securing Service Access

NetworkPolicies allow you to specify how a pod is allowed to communicate with "endpoints" and "services" over the network.

⚠ By default, Pods can access ALL other Pods.

```yaml
kind: NetworkPolicy
metadata:
  name: namespace-deny
  namespace: production
spec:
  podSelector:
    matchLabels: {}
```

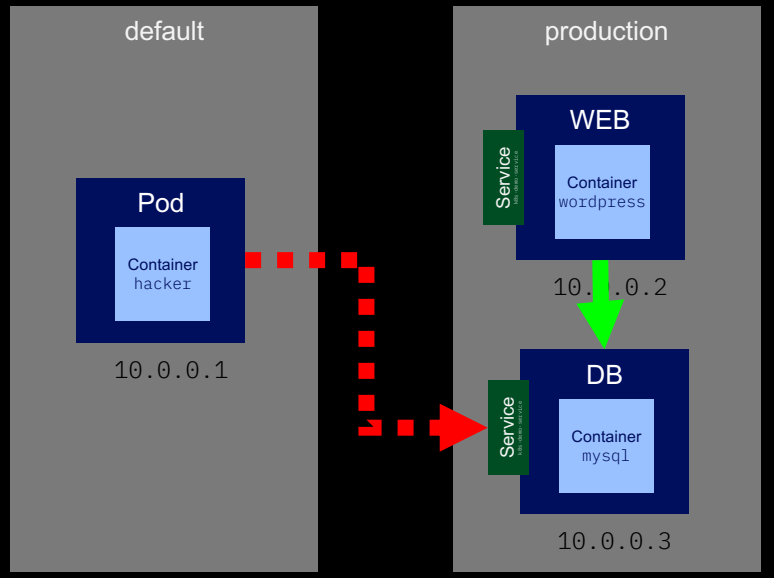


Securing Service Access

NetworkPolicies allow you to specify how a pod is allowed to communicate with "endpoints" and "services" over the network.

⚠ NetworkPoliciers are cumulative!!!

```yaml
kind: NetworkPolicy
metadata:
  name: namespace-allow-db
  namespace: production
spec:
  podSelector:
    matchLabels:
      run: DB
  ingress:
    - from:
      - podSelector:
          matchLabels:
            run: WEB
```



Deployments

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
```

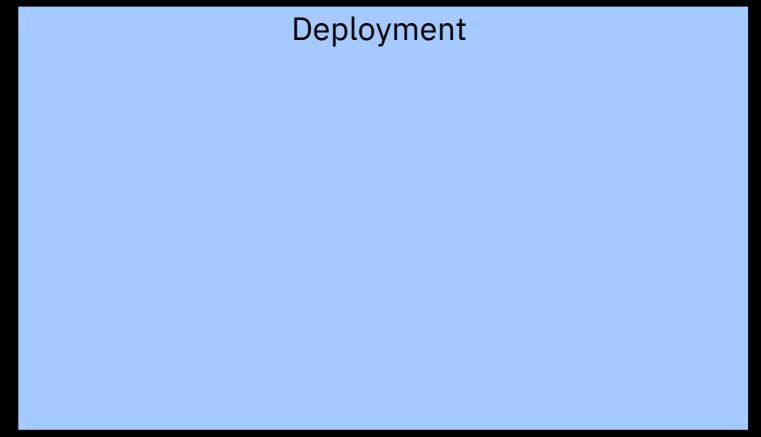
example.yaml

```yaml
apiVersion: app/v1
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
```

Deployment

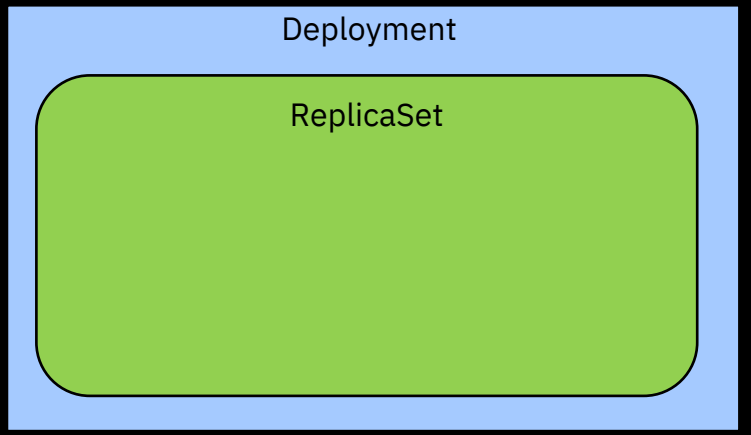
Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
…
  replicas: 2
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        imagePullPolicy: Always
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

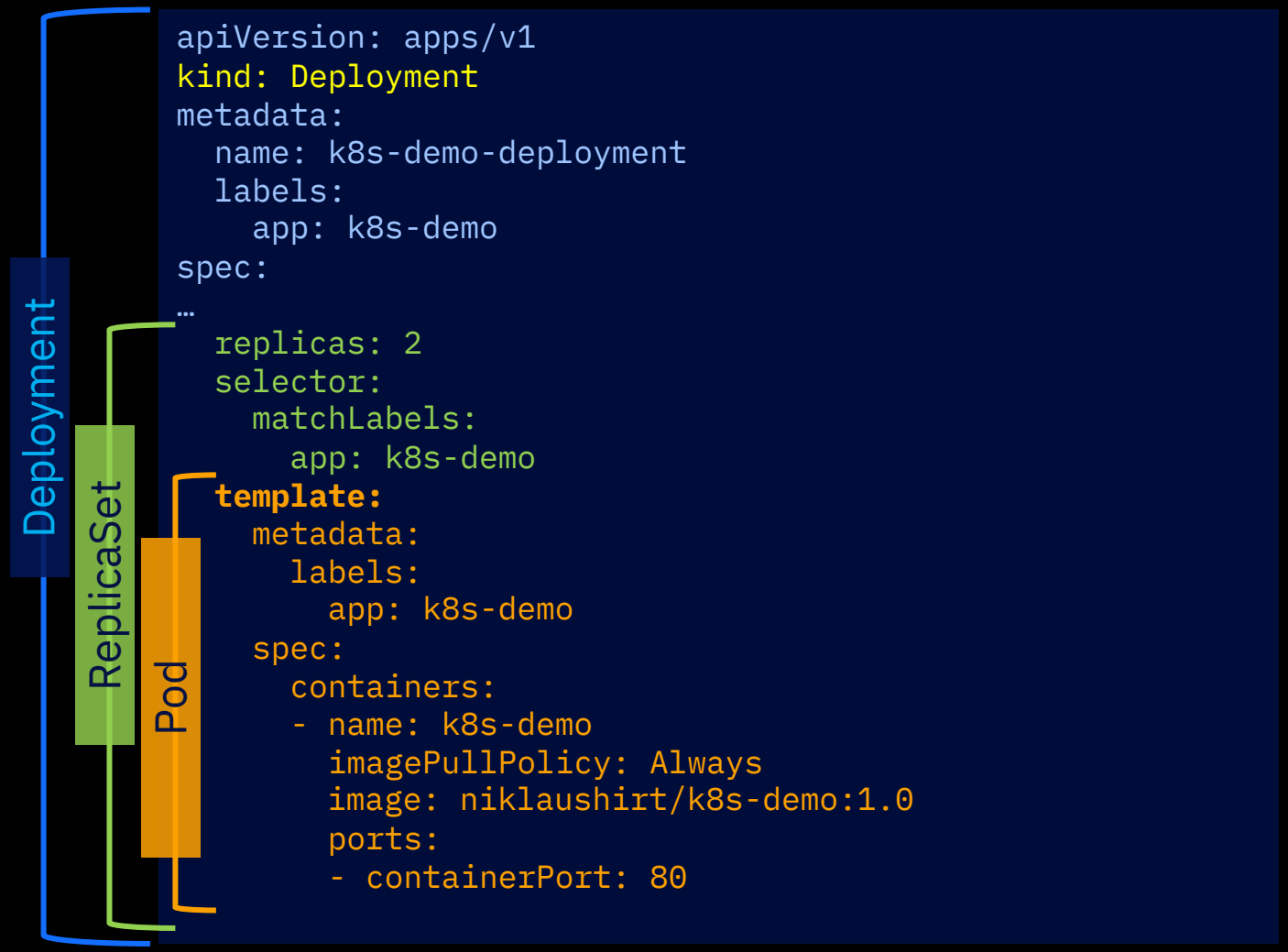


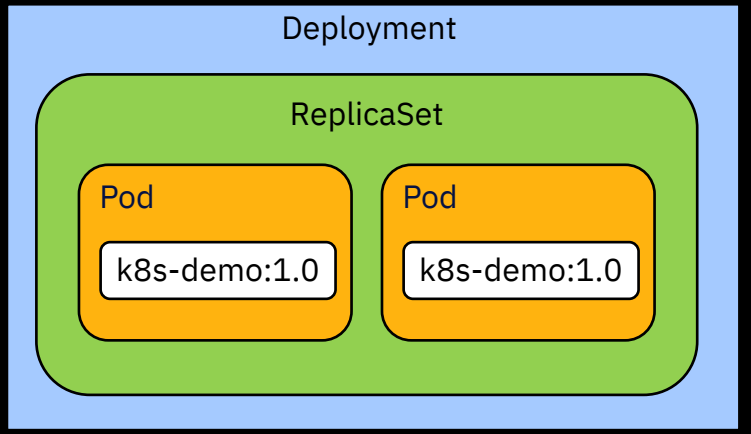
Deployment


Deployment
ReplicaSet
apiVersion: apps/v1
kind: Deployment
metadata:
name: k8s-demo-deployment
labels:
app: k8s-demo
spec:
…
replicas: 2
selector:
matchLabels:
app: k8s-demo
template:
metadata:
labels:
app: k8s-demo
spec:
containers:
- name: k8s-demo
imagePullPolicy: Always
image: niklaushirt/k8s-demo:1.0
ports:
- containerPort: 80


Deployment
ReplicaSet

Deployment


Deployment
ReplicaSet
Pod
apiVersion: apps/v1
kind: Deployment
metadata:
name: k8s-demo-deployment
labels:
app: k8s-demo
spec:
…
replicas: 2
selector:
matchLabels:
app: k8s-demo
template:
metadata:
labels:
app: k8s-demo
spec:
containers:
- name: k8s-demo
imagePullPolicy: Always
image: niklaushirt/k8s-demo:1.0
ports:
- containerPort: 80


Deployment
ReplicaSet
Pod
k8s-demo:1.0
Pod
k8s-demo:1.0

StatefulSet, DaemonSet, Job...

StatefulSet

- Intended to be used with stateful applications and distributed systems.
  - Pods are created sequentially
  - Ordinal index and stable network identity
  - Can create storage for each replica

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

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

Configuration

Configuring Resources and Containers

Implicit Environment Variables

```
allowed: true
enemies: aliens
lives: 3
```

```
kind: Deployment
spec:
  containers:
    - name: test-container
      image: xxx
      ...
      env:
        - name: allowed
          value: true
        - name: enemies
          value: aliens
        - name: lives
          value: 3
```

Can be used as normal environment variable (here `enemies=aliens`)

Configuring Resources and Containers

Environment variables from external sources

ConfigMap

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

Secret *(not so secret)*

- Sensitive info that containers need to read or consume
- Not encrypted! Only base64 encoded!!!
- Must be protected from read access by unauthorized users

Configuring Resources and Containers

Environment variables from ConfigMaps or Secrets

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-config
data:
  allowed: true
  enemies: aliens
  lives: 3
immutable: true
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: my-tls-secret
data:
  tls.crt: >-
    LS0tLS1CRUdJTiBDSUVSakND...
  tls.key: >-
    LS0tLS1CRUdJTi0tLQpNSUlF...
type: Opaque
immutable: true
```

Configuring Resources and Containers

Environment variables from ConfigMaps or Secrets

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-config
data:
  allowed: true
  enemies: aliens
  lives: 3
```

```yaml
kind: Deployment
spec:
  containers:
    - name: test-container
      image: xxx
      ...
```

Can be used as normal environment variable (here `SPECIAL_LEVEL_KEY=aliens`)

Configuring Resources and Containers

Mapping files from ConfigMaps or Secrets

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: my-tls-secret
data:
  tls.crt: >-
    LS0tLS1CRUdJTiBDSUVSakND...
  tls.key: >-
    LS0tLS1CRUdJTi0tLQpNSUlF...
type: Opaque
```

```yaml
kind: Deployment
…
spec:
  containers:
    - name: test-container
      image: xxx

      ...
      volumeMounts:
      - name: my-tls-secret-volume
        mountPath: /workdir/home/certs/public.crt
        subPath: tls.crt

 ...
 volumes:
    - name: my-tls-secret-volume
      secret:
        secretName: my-tls-secret
        items:
          - key: tls.crt
            path: tls.crt
```

The Certificate will be mapped to the `/workdir/home/certs/public.crt` file

Configuring Resources and Containers

Mapping files from ConfigMaps or Secrets

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-config
data:
…
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: my-tls-secret
data:
…
```

```yaml
kind: Deployment
…
spec:
  containers:
    - name: test-container
      image: xxx
      ...
      envFrom:
        - configMapRef:
            name: example-config
            optional: true
        - secretRef:
            name: my-tls-secret
            optional: true
```

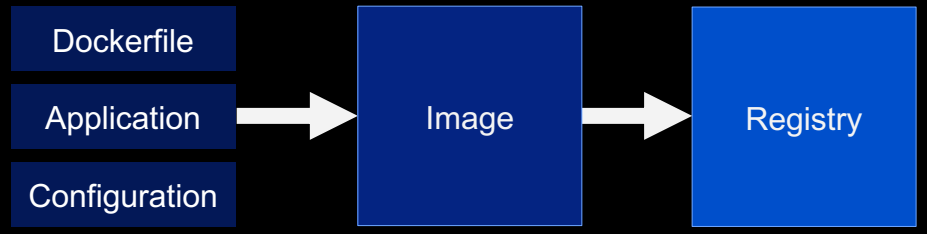
Kubernetes Workshop Series

Kubernetes - *Applied*

05

..but how do I use this for real?

Minimal Application – Container Image



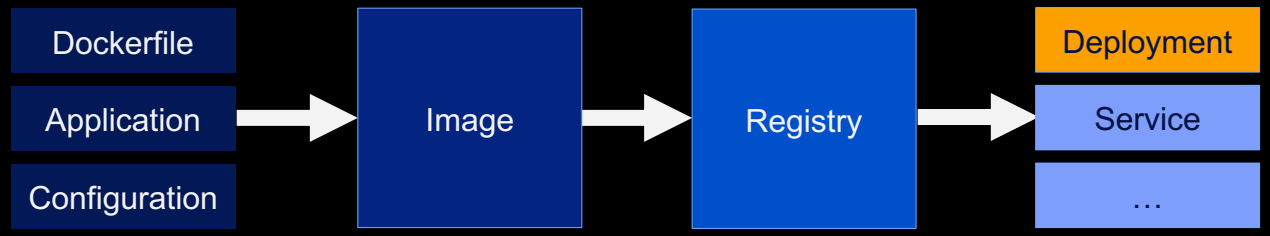

```
FROM node:8-stretch

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
&& apt-get dist-upgrade -y \
&& apt-get clean \
&& echo 'Finished installing dependencies'

# Install npm production packages
COPY package.json /app/
RUN cd /app; npm install --production

COPY ./myapp /app

ENV NODE_ENV production
ENV BACKEND_URL https://api.nasa.gov/planetary/apod\?api_key\=_KEY
ENV PORT 3000

EXPOSE 3000

CMD ["npm", "start"]
```

Minimum components you need:

- Built Application or source
  (NodeJS, Java, Golang, ...)
- Dockerfile
  → Use the right base image for your app

```
❯ podman build -t niklaushirt/k8s-demo:1.0 .
❯ podman push niklaushirt/k8s-demo:1.0
```

Minimal Application – Kubernetes Manifests

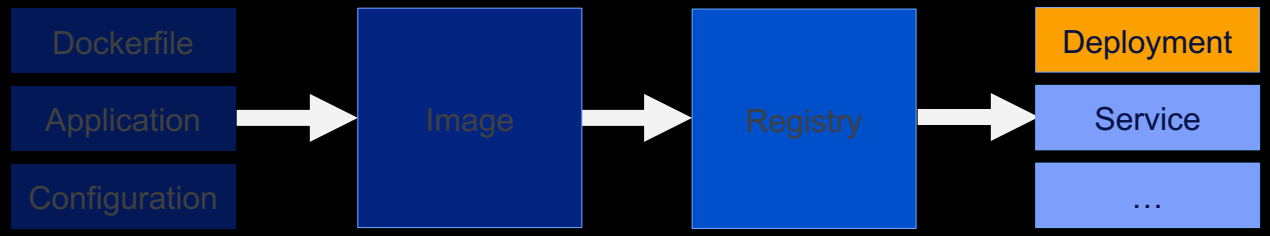



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        imagePullPolicy: Always
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

Minimum components you need:

- Deployment
- Service

Minimal Application – Kubernetes Manifests



Deployment

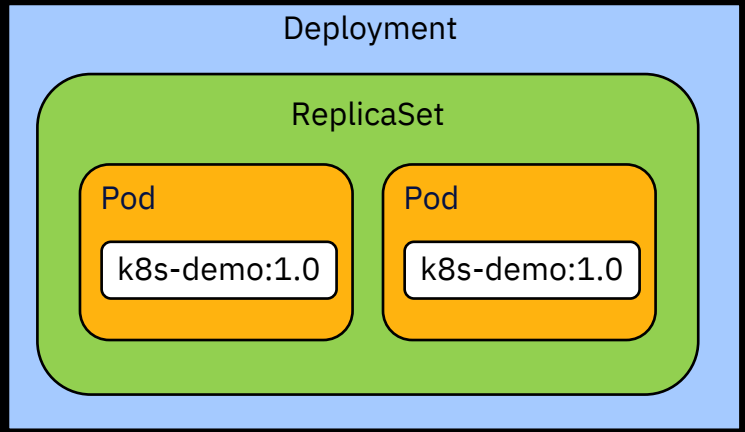
ReplicaSet

Pod

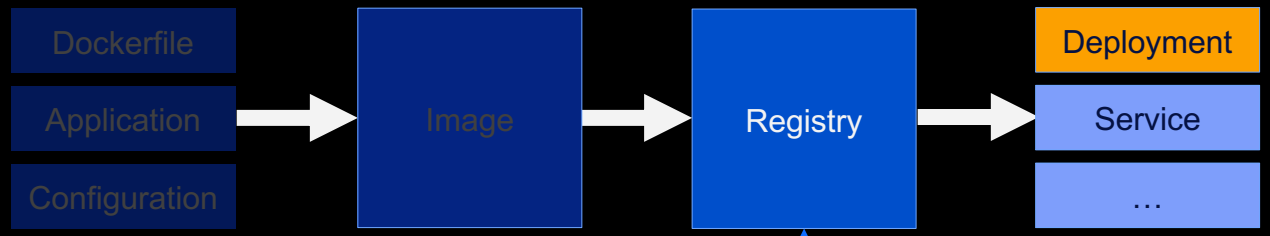
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        imagePullPolicy: Always
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

Minimum components you need:

- Deployment
- Service



Minimal Application – Kubernetes Manifests




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        imagePullPolicy: Always
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

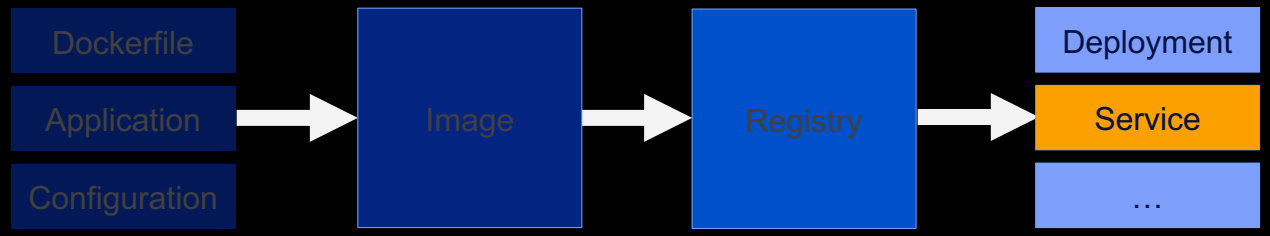
Minimum components you need:

- Deployment
- Service

imagePullPolicy:

IfNotPresent: only pulled if not already present locally
Always: every time the kubelet launches a container

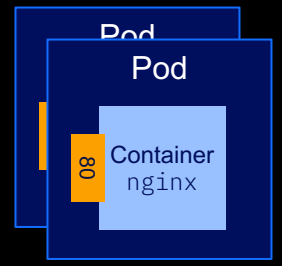
Minimal Application – Kubernetes Manifests



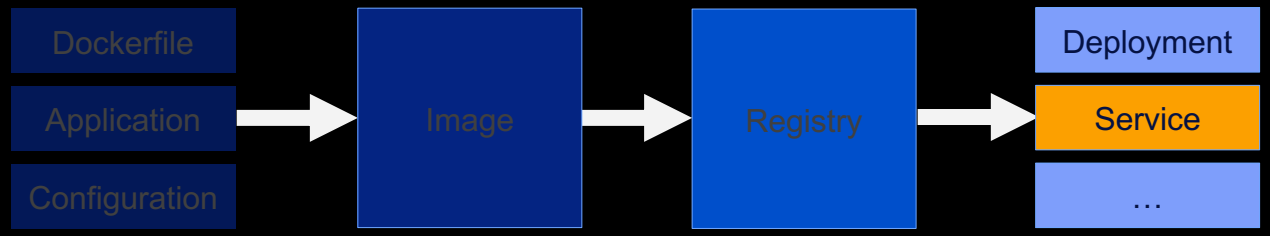

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        imagePullPolicy: Always
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

Minimum components you need:

- Deployment
- Service



Minimal Application – Kubernetes Manifests

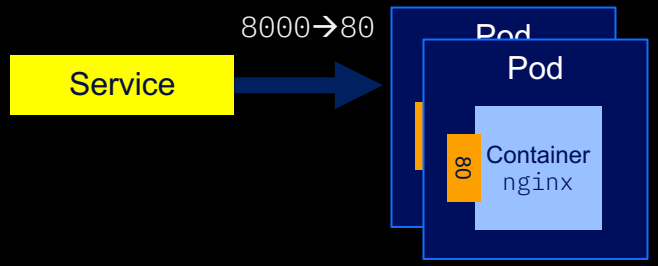



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        imagePullPolicy: Always
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

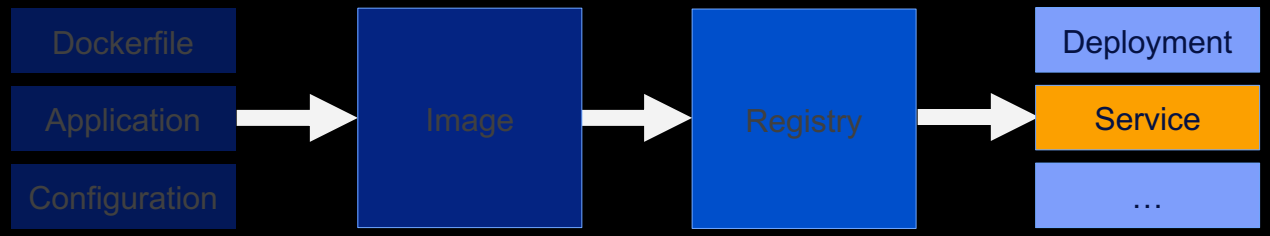
```yaml
apiVersion: v1
kind: Service
metadata:
  name: k8s-demo-service
spec:
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
  selector:
    app: k8s-demo
```

Minimum components you need:

- Deployment
- Service



Minimal Application – Kubernetes Manifests

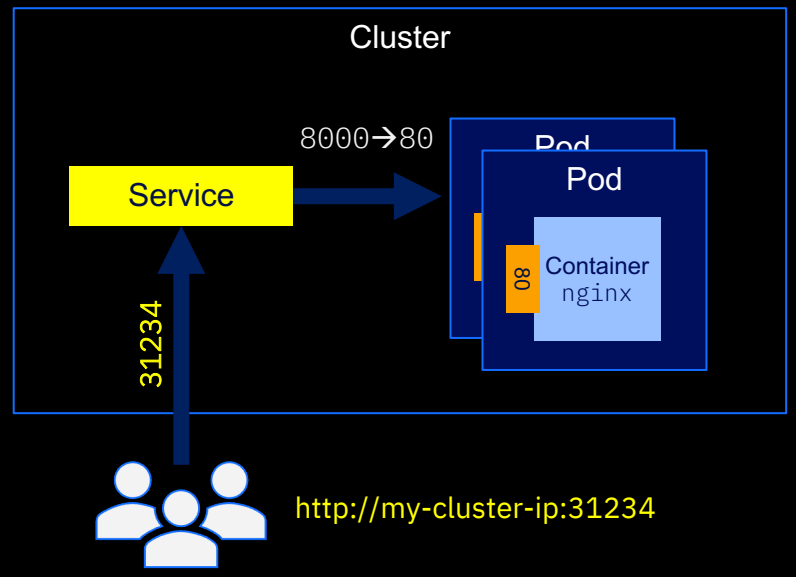



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        imagePullPolicy: Always
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

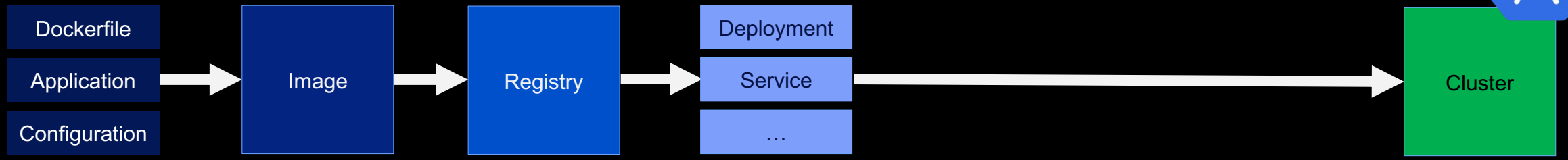
```yaml
apiVersion: v1
kind: Service
metadata:
  name: k8s-demo-service
spec:
  type: NodePort
  ports:
  - port: 8000
    targetPort: 80
    nodePort: 31234
    protocol: TCP
  selector:
    app: k8s-demo
```

Minimum components you need:

- Deployment
- Service



Minimal Application – Deploying




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: k8s-demo-service
spec:
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
  selector:
    app: k8s-demo
```

Minimum components you need:

- Deployment
- Service

```
❯ kubectl apply -f deployment.yaml
❯ kubectl apply -f service.yaml
```

Minimal Application – Helm

The package manager for Kubernetes

Helm helps you manage Kubernetes applications — Helm Charts help you **define, install, and upgrade** even the most complex Kubernetes application.

Minimal Application – Helm


```yaml
apiVersion: v1
name: k8s-demo
description: Demo App for the training.
keywords:
  - Training
maintainers:
  - name: Niklaus Hirt

version: 1.10.0
appVersion: 19.0.0.6

home: https://github.com/niklaushirt
icon: https://github.com/niklaushirt/logo-01.png
sources:
  - https://github.com/niklaushirt/charts/tree/master/stable
```

Minimum components we need:

Chart.yaml

Basic information about the chart

Templates (yaml)

Kubernetes deployment manifests (yaml) with placeholders for external parameters

values.yaml

A set of variables that will be passed on at deploy time to the deployment manifest templates

Minimal Application – Helm


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        image: {{ .Values.image.name }}:{{ .Values.image.tag }}
        ports:
        - containerPort: 80
```

Minimum components we need:

Chart.yaml

Basic information about the chart

Templates (yaml)

Kubernetes deployment manifests (yaml) with placeholders for external parameters

values.yaml

A set of variables that will be passed on at deploy time to the deployment manifest templates

Minimal Application – Helm


```yaml
apiVersion: v1
image:
  name:
    - niklaushirt/k8s-demo
  tag: 1.0
  pullPolicy: IfNotPresent
replicas: 1
```

Minimum components we need:

Chart.yaml

Basic information about the chart

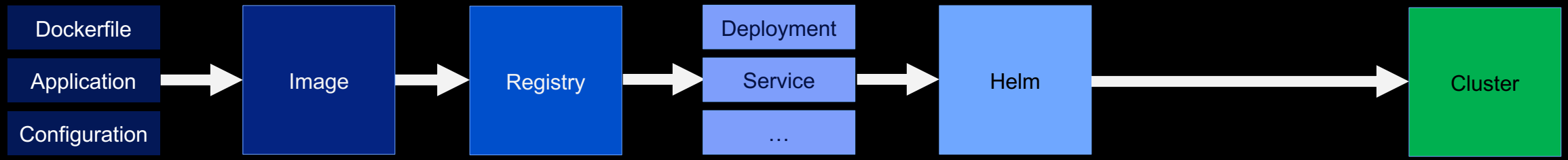
Templates (yaml)

Kubernetes deployment manifests (yaml) with placeholders for external parameters

values.yaml

A set of variables that will be passed on at deploy time to the deployment manifest templates

Minimal Application – Helm



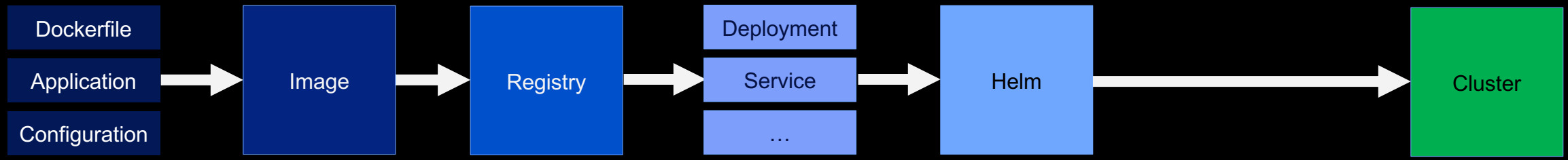

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        image: {{ .Values.image.name }}:{{ .Values.image.tag }}
        ports:
        - containerPort: 80
```

```yaml
apiVersion: v1
image:
  name:
    - niklaushirt/k8s-demo
  tag: 1.0
  pullPolicy: IfNotPresent
replicas: 1
```

```
helm install k8s-demo
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

Minimal Application – Helm



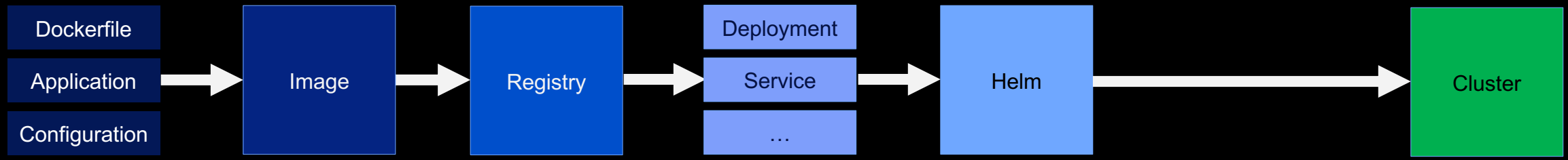

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        image: {{ .Values.image.name }}:{{ .Values.image.tag }}
        ports:
        - containerPort: 80
```

```yaml
apiVersion: v1
image:
  name:
    - niklaushirt/k8s-demo
  tag: 1.1
  pullPolicy: IfNotPresent
replicas: 3
```

```
helm install k8s-demo
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        image: niklaushirt/k8s-demo:1.1
        ports:
        - containerPort: 80
```

Minimal Application – Helm



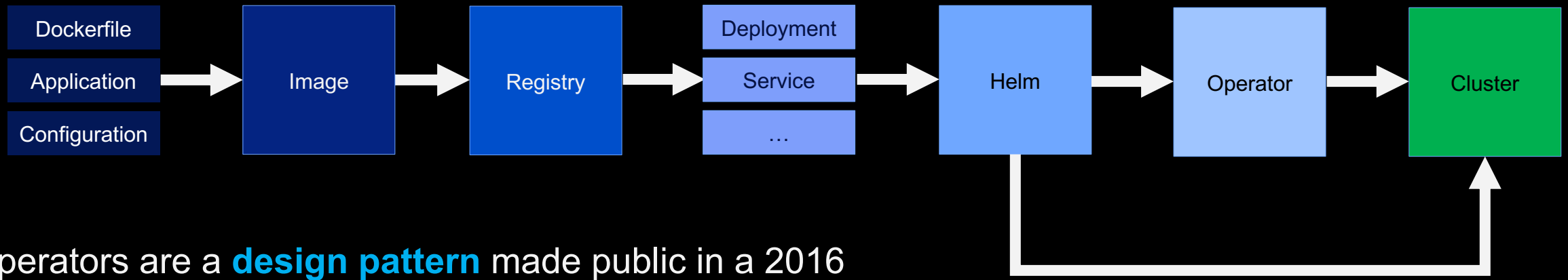

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        image: {{ .Values.image.name }}:{{ .Values.image.tag }}
        ports:
        - containerPort: 80
```

```yaml
apiVersion: v1
image:
  name:
    - niklaushirt/k8s-demo
  tag: 1.0
  pullPolicy: IfNotPresent
~~replicas: 1~~
```

```
helm install --set replicas=3 k8s-demo
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
      - name: k8s-demo
        image: niklaushirt/k8s-demo:1.0
        ports:
        - containerPort: 80
```

Minimal Application – Operators

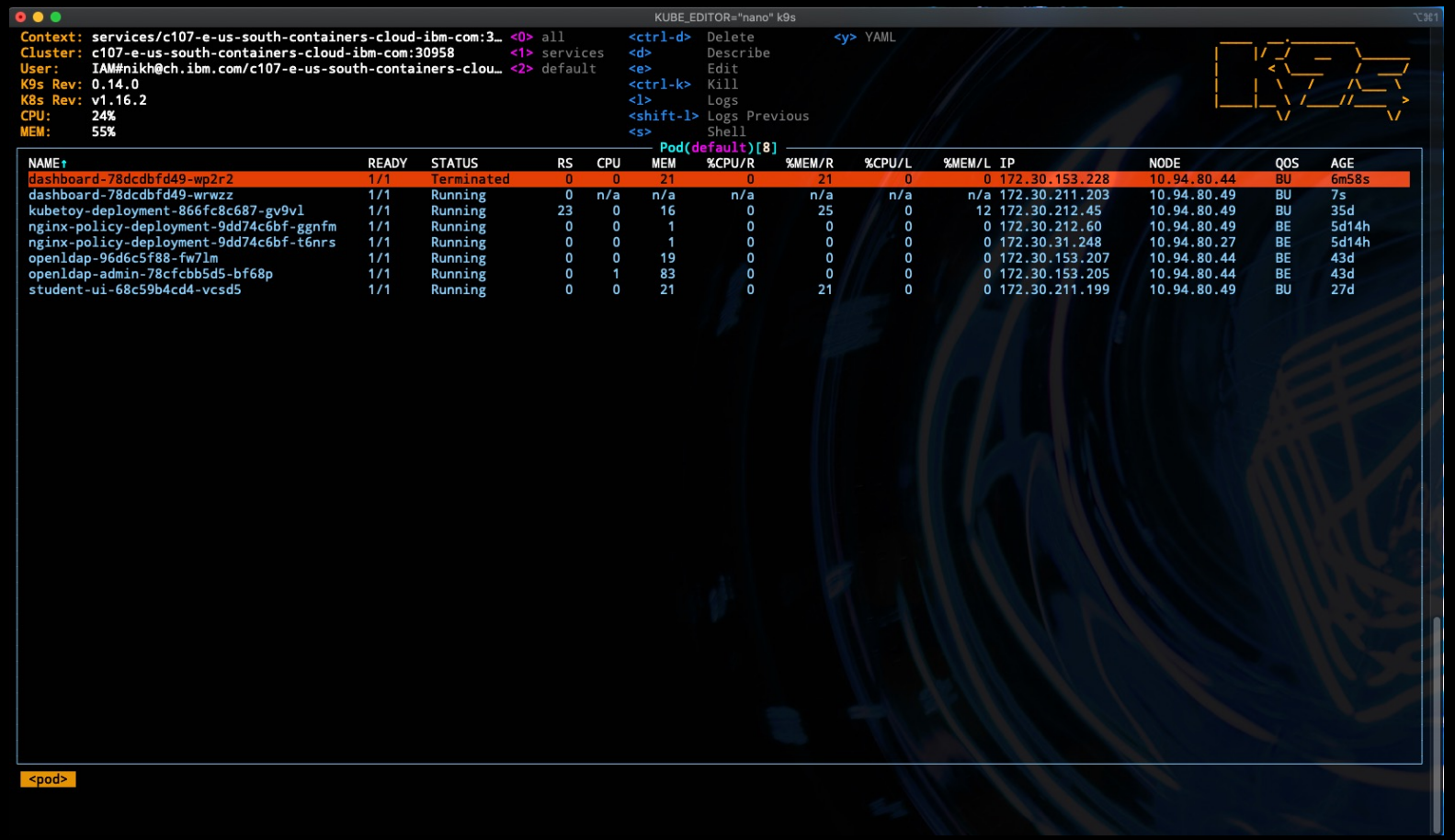


Operators are a **design pattern** made public in a 2016 CoreOS blog post.

The goal of an Operator is to put **operational knowledge into software**.

Operators implement and automate common
Day-1 (installation, configuration, etc) and
Day-2 (re-configuration, update, backup, failover, restore, etc.) **activities**
in a piece of software running inside your Kubernetes cluster, by integrating natively with Kubernetes concepts and APIs.

K9s – the Norton Commander of Kubernetes



Preinstalled in the training VM

For dumb people like me
(who just don't get vi)

```
KUBE_EDITOR="nano" k9s
```

I did create an alias in .bashrc/.zshrc

```
alias k9s='KUBE_EDITOR="nano" k9s'
```

https://github.com/derailed/k9s

QUESTIONS?

Kubernetes Workshop Series

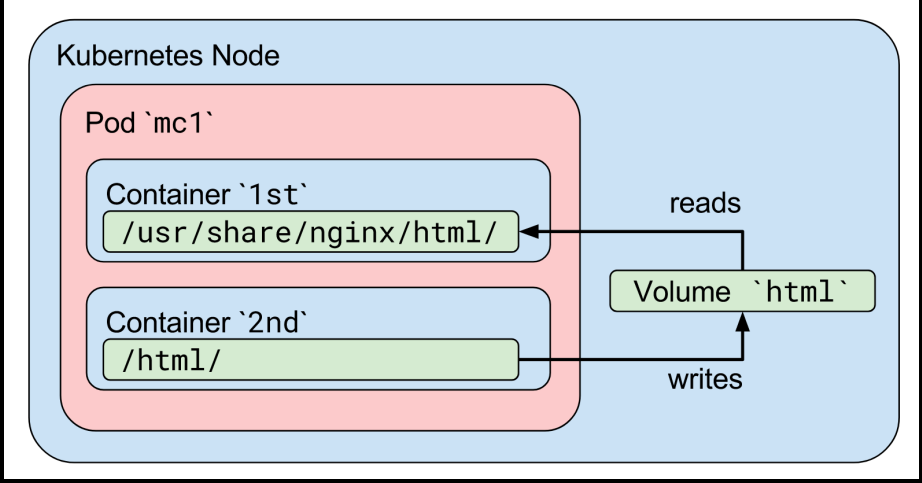
Kubernetes –Advanced concepts

06

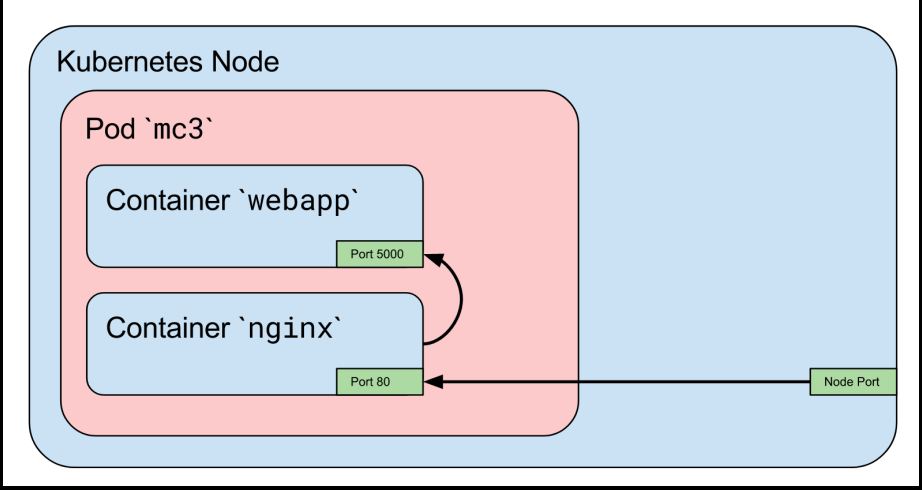
One container per Pod

Multi-Container Pod Design Patterns

Shared Volumes

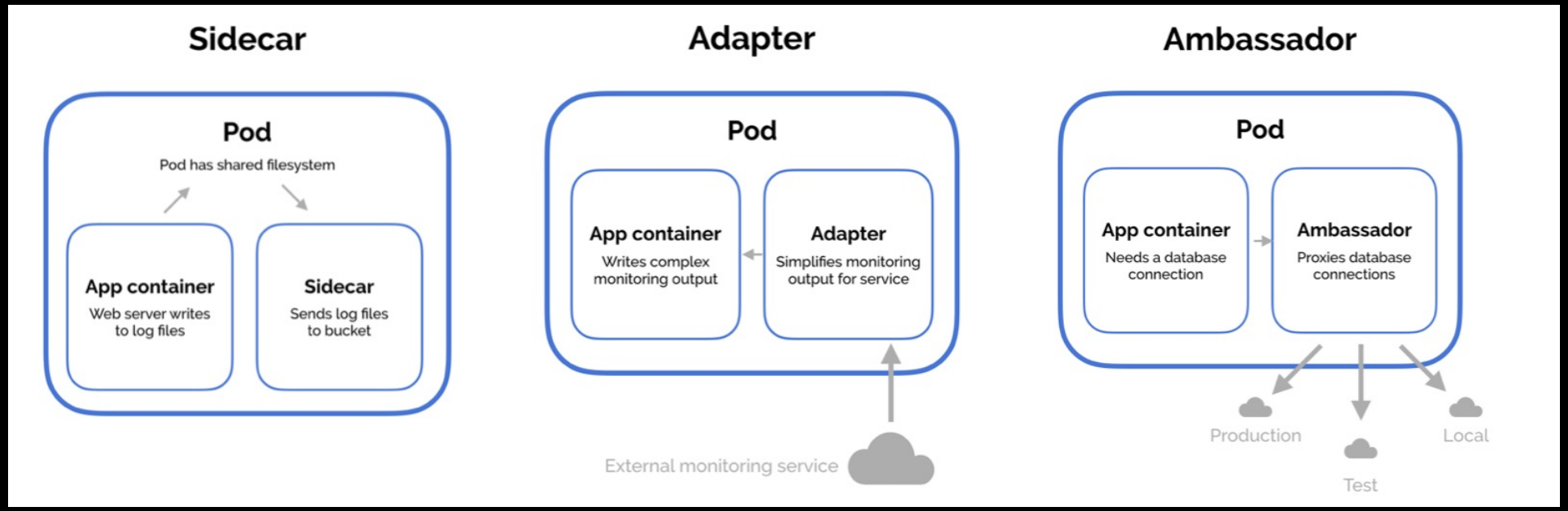


No Network isolation



In what order containers are being started in a Pod?
The containers scale together…

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

Requests and Limits

Resources: Requests and Limits

```
apiVersion: apps/v1
kind: Deployment
metadata:
…
spec:
…
template:
…
  resources:
    requests:
      memory:512Mi
      cpu: 100m
    limits:
      memory: 1900Mi
      cpu: 300m
```

When working with containers in Kubernetes, it’s important to know what are the resources involved and how they are needed.

Some processes will require more CPU or memory than others. Some are critical and should never be starved.
Knowing that, we should configure our containers and Pods properly in order to get the best of both.

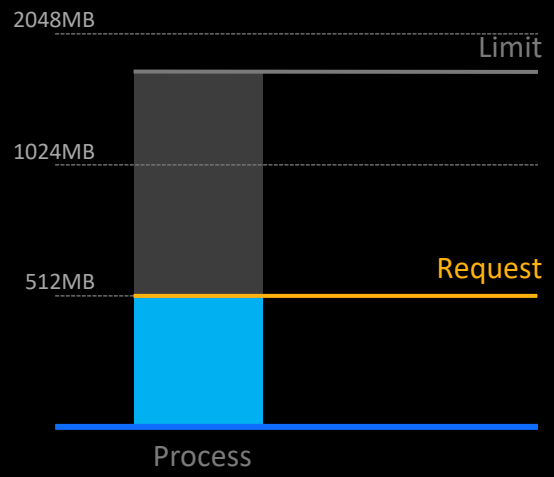
Resources: Requests and Limits

```
apiVersion: apps/v1
kind: Deployment
metadata:
…
spec:
…
template:
…
  resources:
    requests:
      memory: 512Mi
      cpu: 100m
    limits:
      memory: 1900Mi
      cpu: 300m
```

Requests affect how the pods are scheduled in Kubernetes.

When a Pod is created, the scheduler finds the nodes which can accommodate the Pod according to the resource request.

Requests define the minimum amount of resources that containers need.
If you think that your app requires at least 256MB of memory to operate, this is the request value.
The application can use more than 256MB, but Kubernetes guarantees a minimum of 256MB to the container.



Error event: FailedScheduling

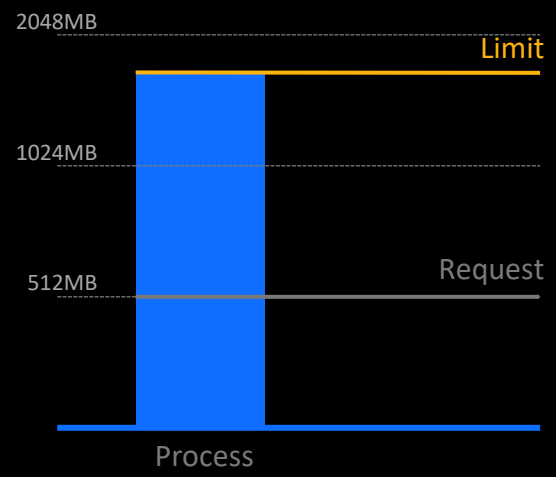
Resources: Requests and Limits

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
…
spec:
…
template:
…
  resources:
    requests:
      memory: 512Mi
      cpu: 100m
    limits:
      memory: 1900Mi
      cpu: 300m
```

Limits define the max amount of resources that the container can consume.
They protect the Cluster from runaway Pods that eat up all CPU and/or Memory.

The **CPU limit** defines how much CPU time the container can use.
The Linux kernel (cgroup) checks to see if this limit is exceeded; if so, the kernel waits before allowing that cgroup to resume execution.

Error event:
CPU: NONE → throttling!



The **memory limit** defines a memory limit for that cgroup.
If the container tries to allocate more memory than this limit, the Linux kernel stops the processes in the container that tried to allocate memory.
If that process is the container's PID 1, and the container is marked as restartable, Kubernetes restarts the container.

Error events:
Memory: OOM Killed (out of memory) – only for PID1, otherwise invisible

Probes

Readiness and Liveness Probes

Running	The Pod has been bound to a node, and all of the containers have been created.

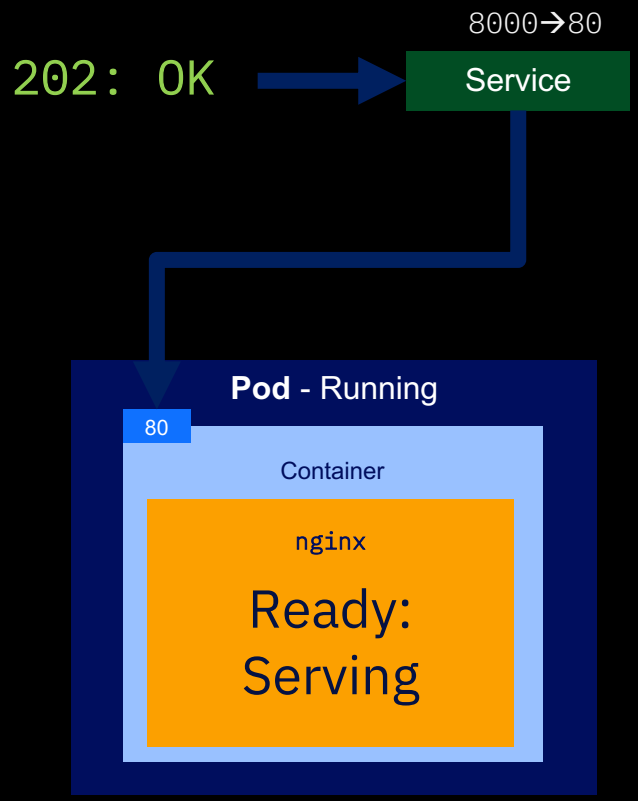
Does not guarantee that the process in the container is fully initialized and Ready to receive traffic.

As soon as a Pod is Running it is also considered Ready and traffic is sent to it.

HTTP Error

Readiness and Liveness Probes

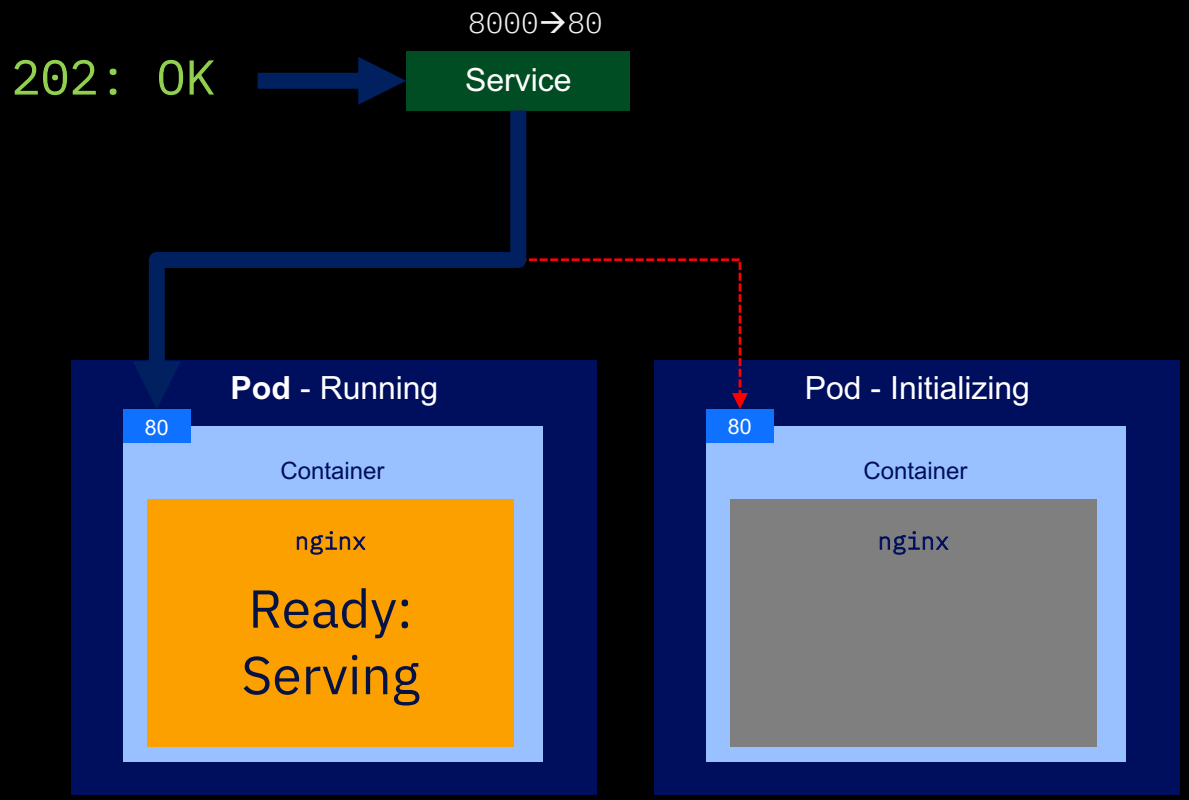
- NGINX Application running




```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  …
```

Readiness and Liveness Probes

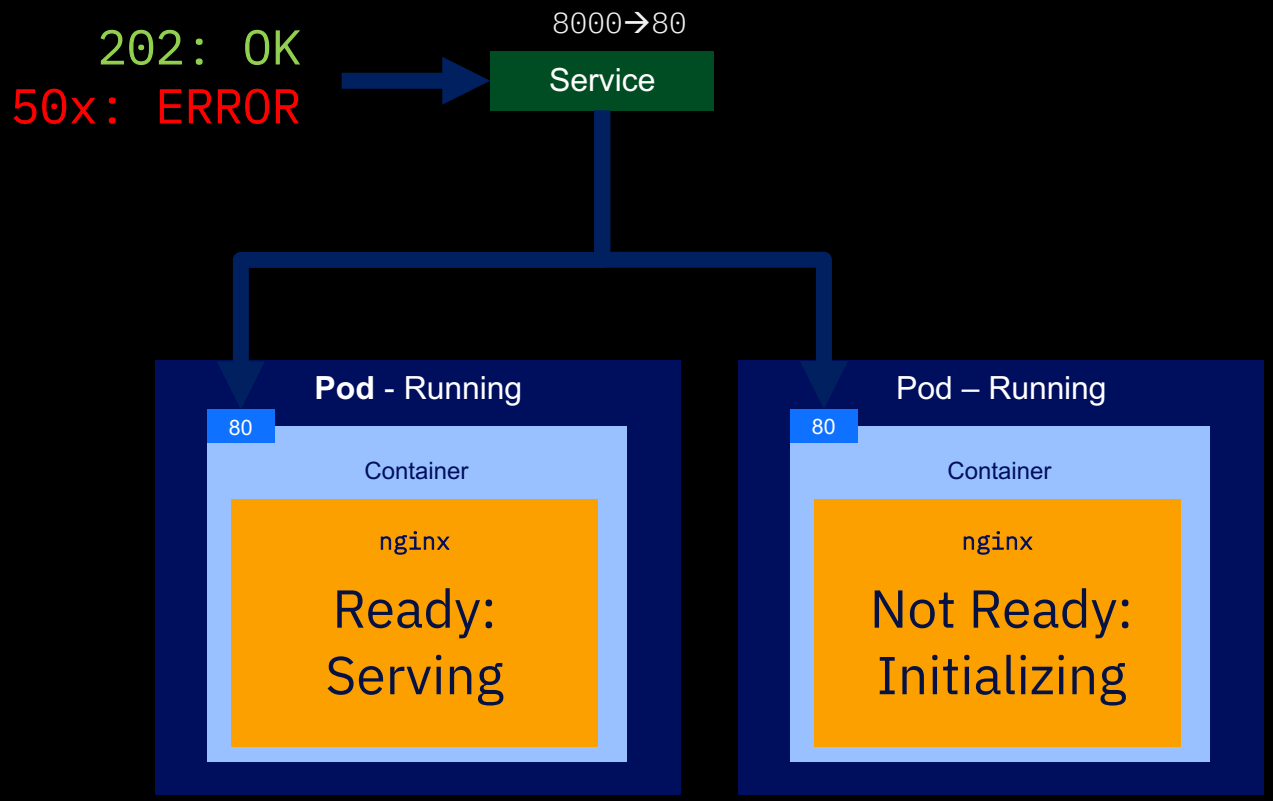
- NGINX Application running
- Scale-up




```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  …
```

Readiness and Liveness Probes

- NGINX Application running
- Scale-up
- Traffic routed to the new Pod as soon as it's Running, even if the process is not ready




```
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  …
```

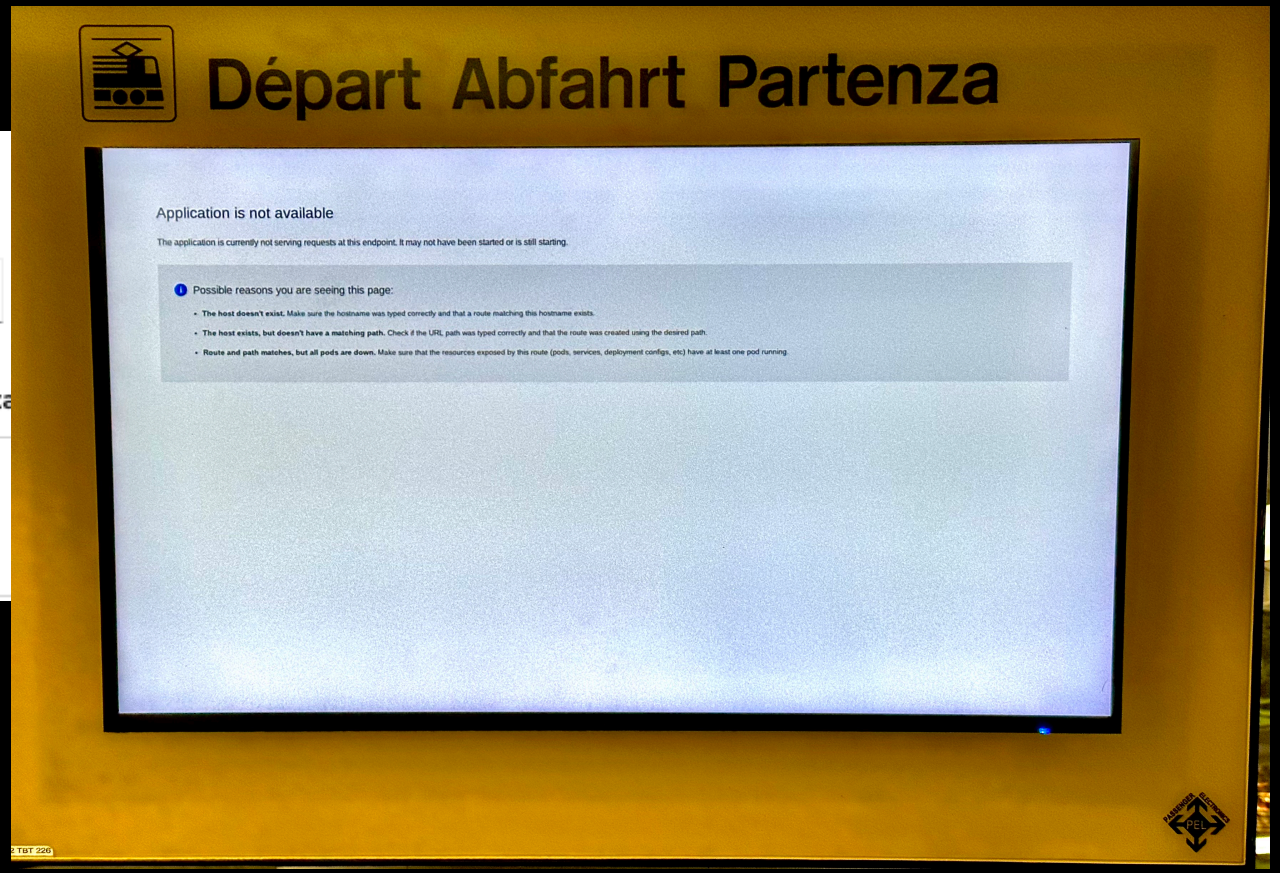
Readiness and Liveness Probes

RUNNING ≠ READY

Pods

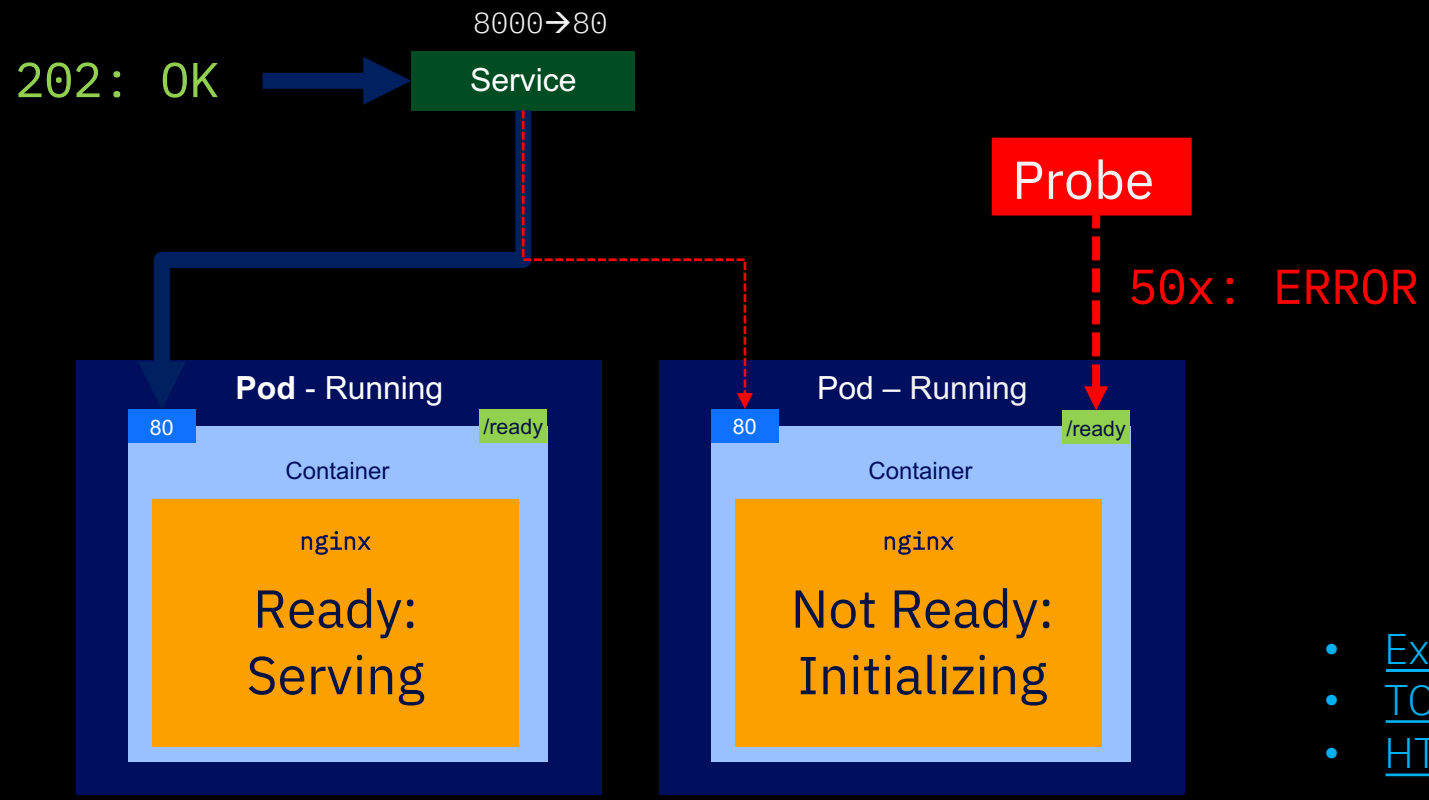
Filter | Name | Search by name... /

Name	Status	Ready	Resta
cp4waiops-demo-ui-7fbbf8cc5d-7tsmn	Running	0/1	0



Readiness Probes

- NGINX Application running
- Readiness Probe checks /ready Endpoint
- If no response, keep status Initializing

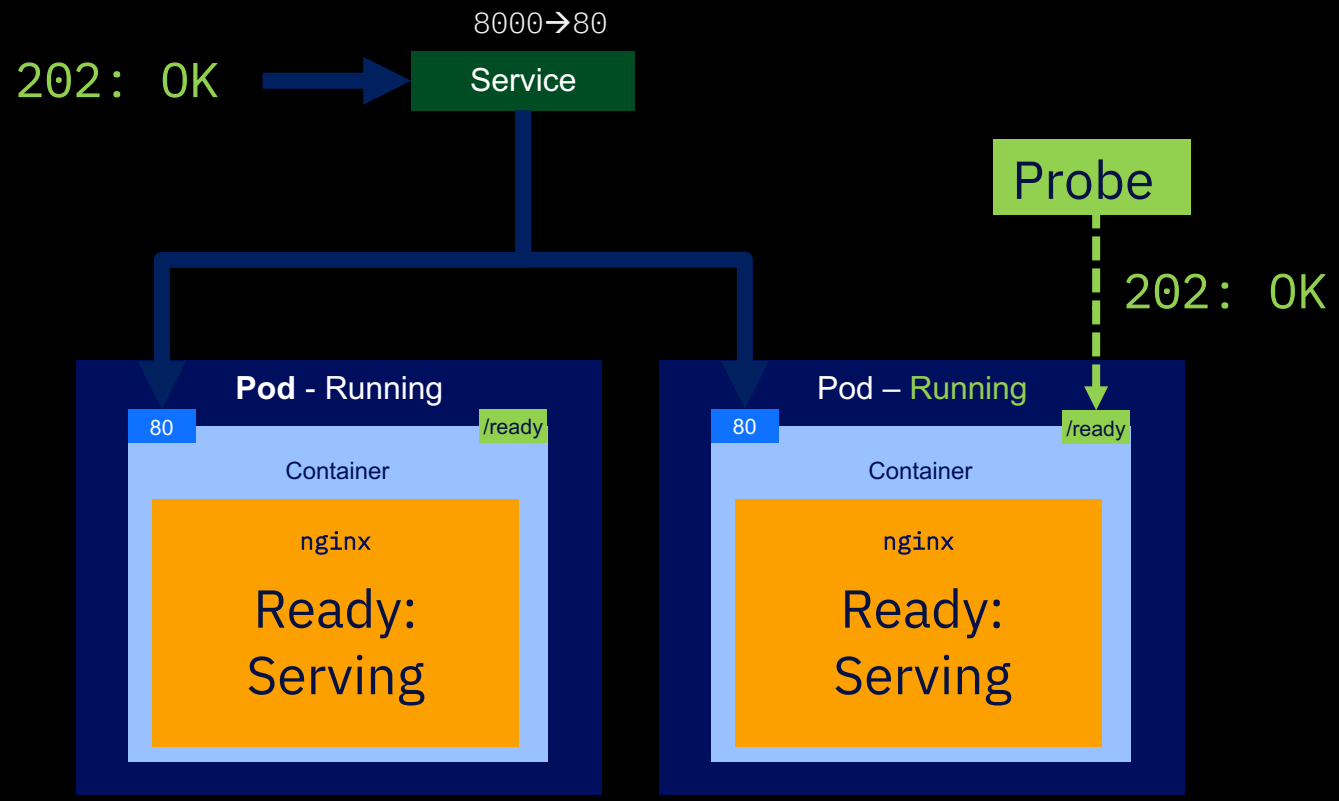



```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  …
  readinessProbe:
    # an http probe
    httpGet:
      path: /ready
      port: 8080
    initialDelaySeconds: 20
    periodSeconds: 5
```

- ExecAction: Executes a specified command
- TCPSocketAction: Performs a TCP check
- HTTPGetAction: Performs an HTTP GET request

Readiness Probes

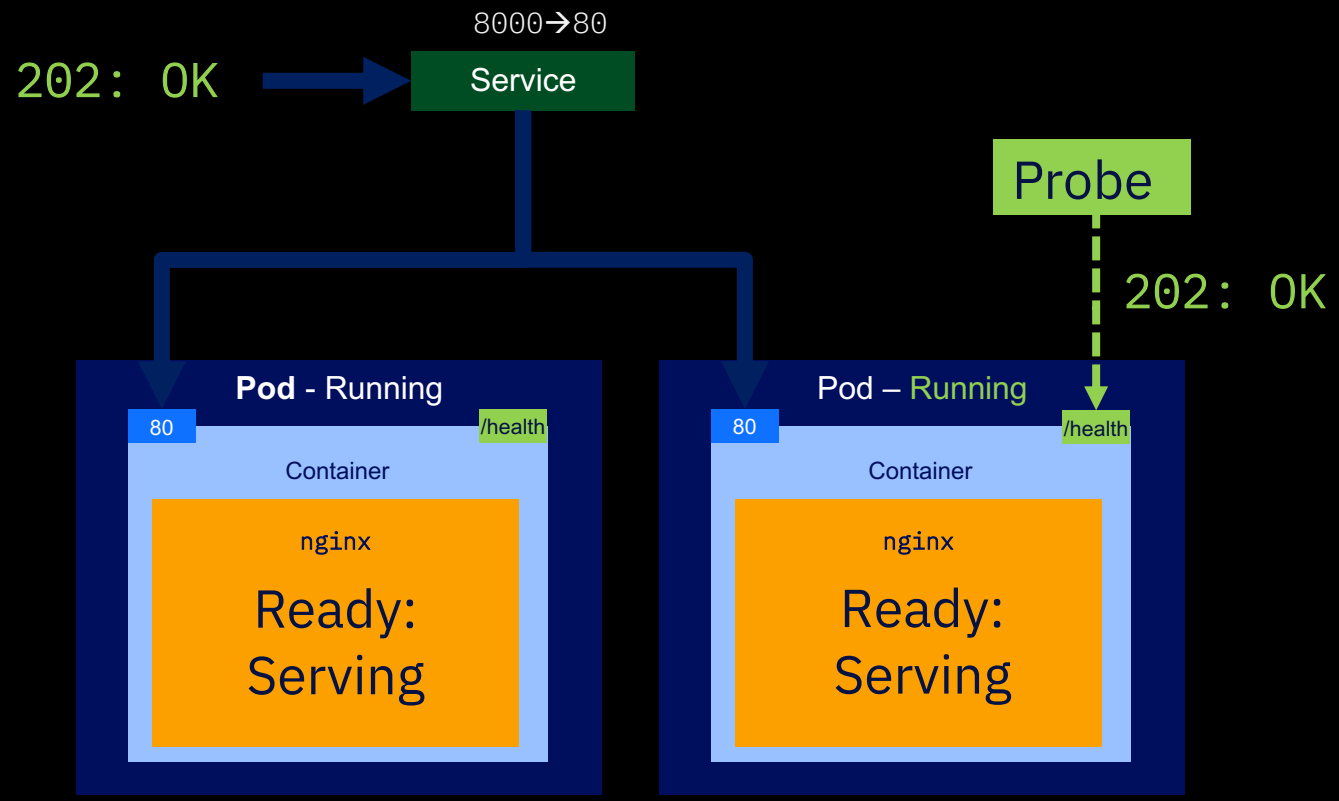
- NGINX Application running
- Readiness Probe checks /ready Endpoint
- If response, set status Ready




```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  …
  readinessProbe:
    # an http probe
    httpGet:
      path: /ready
      port: 8080
    initialDelaySeconds: 20
    periodSeconds: 5
```

Liveness Probes

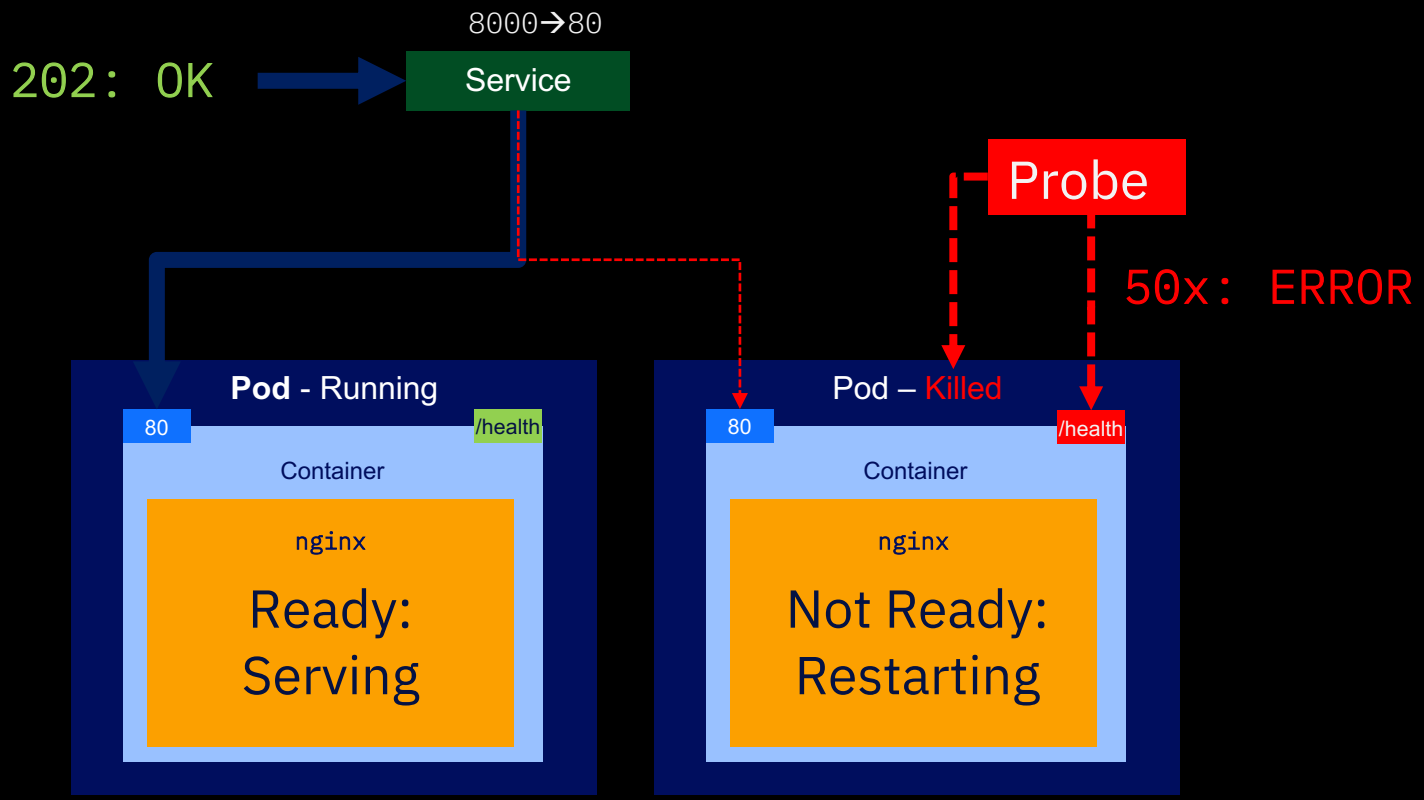
- NGINX Application running
- Liveness Probe checks /health Endpoint
- If no response, restarts Pod




```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  …
  livenessProbe:
    # an http probe
    httpGet:
      path: /health
      port: 8080
    initialDelaySeconds: 20
    periodSeconds: 5
```

Liveness Probes

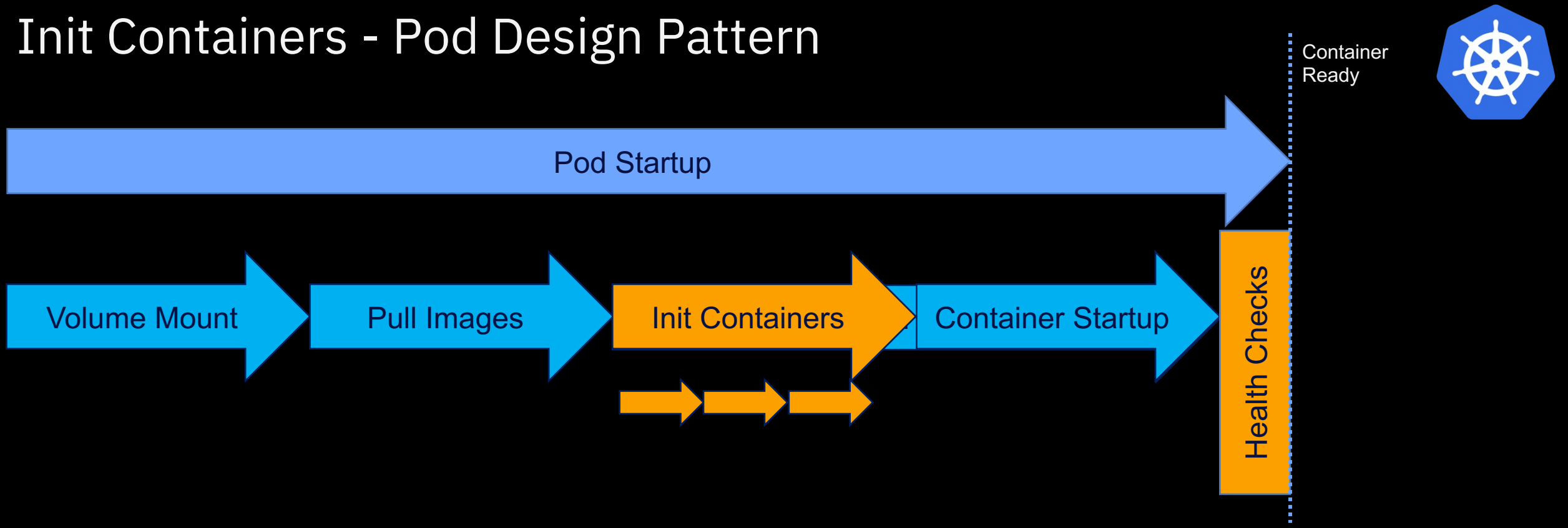
- NGINX Application running
- Liveness Probe checks /health Endpoint
- If no response, restarts Pod




```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  …
  livenessProbe:
    # an http probe
    httpGet:
      path: /health
      port: 8080
    initialDelaySeconds: 20
    periodSeconds: 5
```

Init Containers

Init Containers - Pod Design Pattern



- Init Containers can contain utilities or custom code for setup that are not present in an app image.
- Init Containers can run with a different view of the filesystem than app containers in the same Pod.
- Init Containers offer a mechanism to block or delay app container startup until a set of preconditions are met.
- Init Containers can securely run utilities or custom code that would otherwise make an app container image less secure.

Source: https://kubernetes.io/docs/concepts/workloads/pods/init-containers/

Init Containers - Pod Design Pattern

Examples:

- Wait for a Service to be created, using a shell one-line command
- Register the Pod with a remote server from the downward API with a command
- Wait for some time before starting the app container
- Clone a Git repository into a Volume
- Place values into a configuration file and run a template tool to dynamically generate a configuration file for the main app container.

Source: https://kubernetes.io/docs/concepts/workloads/pods/init-containers/

Init Containers

```
apiVersion: app/v1
kind: Deployment
metadata:
  name: nginx-deployment
…
```

Source: https://kubernetes.io/docs/concepts/workloads/pods/init-containers/

Init Containers

Example

- Sleep before starting NGINX

example.yaml

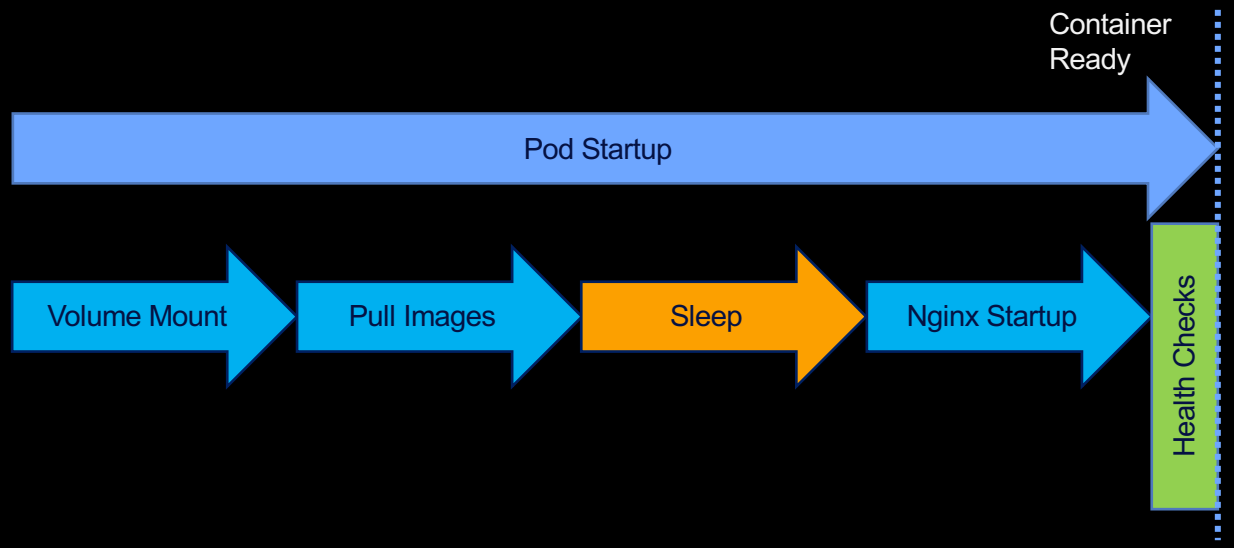
```yaml
apiVersion: app/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  template:
     metadata:
       labels:
         app: nginx
     spec:
       containers:
       - name: nginx
          image: nginx:1.7.9
          ports:
          - containerPort: 80
```

Source: https://kubernetes.io/docs/concepts/workloads/pods/init-containers/

Init Containers

Example

- Sleep before starting NGINX



example.yaml

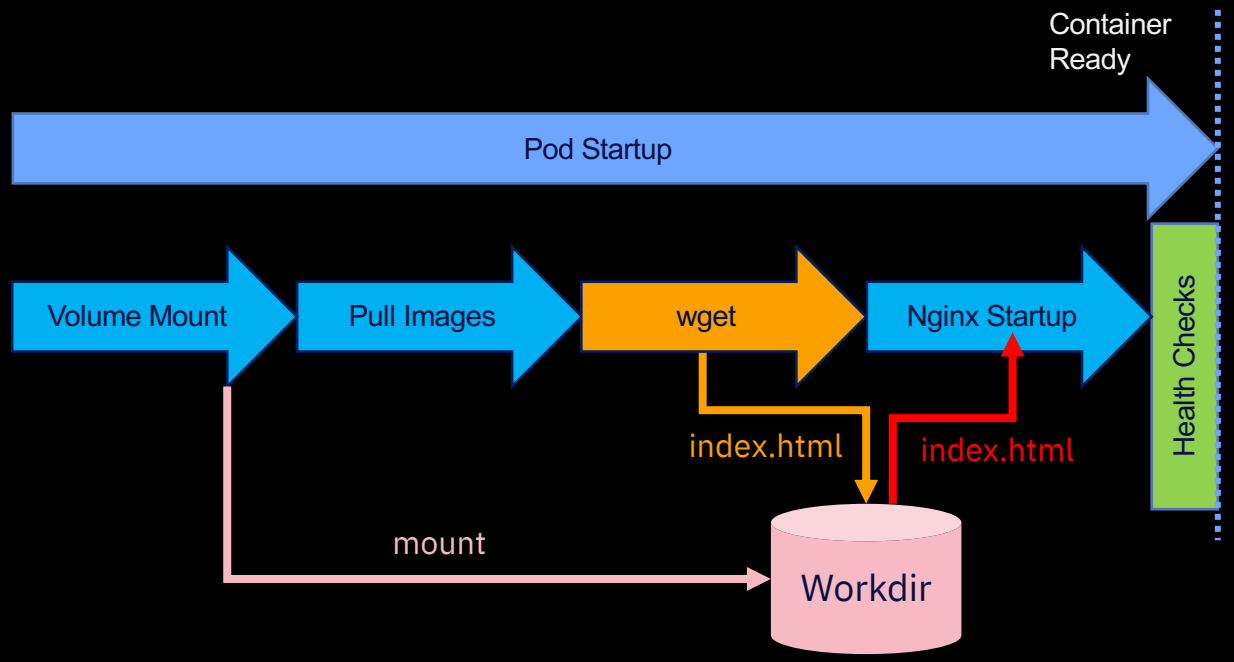
```yaml
apiVersion: app/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  template:
     metadata:
       labels:
         app: nginx
     spec:
       containers:
       - name: nginx
          image: nginx:1.7.9
          ports:
          - containerPort: 80

     #This container is run during pod initialization
     initContainers:

     #Sleep for 2 Seconds
     - name: sleep-1
       image: busybox
       imagePullPolicy: IfNotPresent
       command: ['sh', '-c', 'sleep 2;echo sleep complete;']
```

Source: https://kubernetes.io/docs/concepts/workloads/pods/init-containers/

Init Containers

Example

- NGINX Data initialization



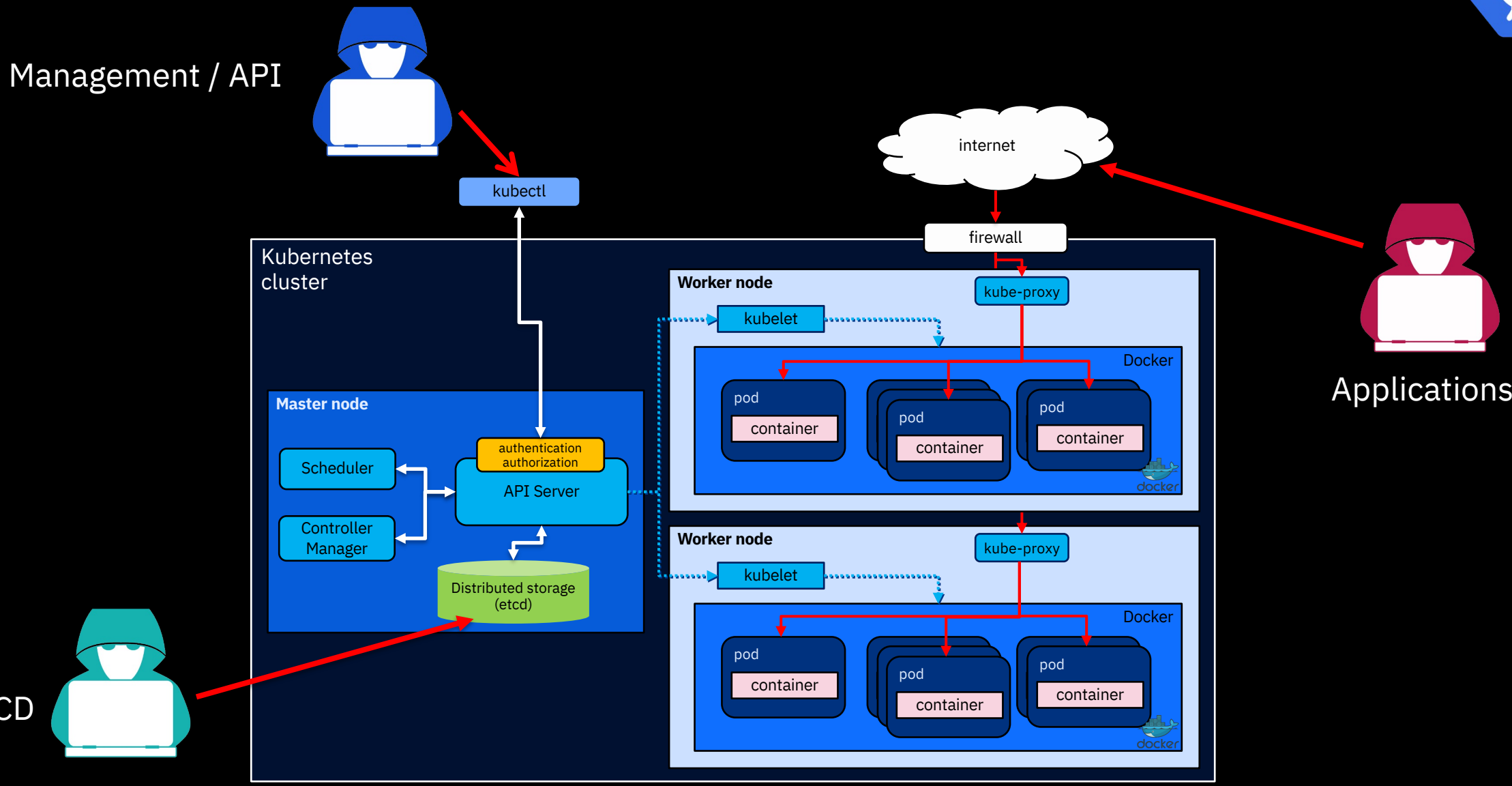
example.yaml

```yaml
apiVersion: app/v1
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
       #Volume mapped into the nginx pod
       volumeMounts:
       - name: workdir
         mountPath: /usr/share/nginx/html

     #This container is run during pod initialization
     initContainers:

     - name: install
       image: busybox

       #Download index.html and store into work-dir
       command:
       - wget
       - "-O"
       - "/work-dir/index.html"
       - http://kubernetes.io
       volumeMounts:
       - name: workdir
         mountPath: "/work-dir"

     volumes:
     - name: workdir
     emptyDir: {}
```

Source: https://kubernetes.io/docs/concepts/workloads/pods/init-containers/

Basic Security

Kubernetes – **Security – Attack surface**



Kubernetes – **Security Basic Topics**

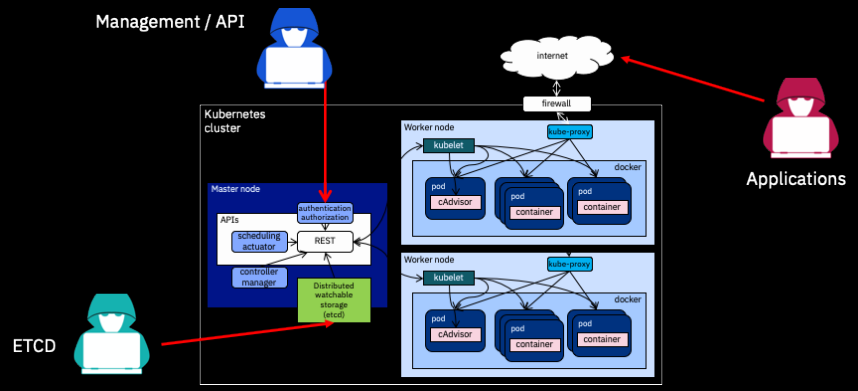
Reduce Kubernetes Attack Surfaces

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/

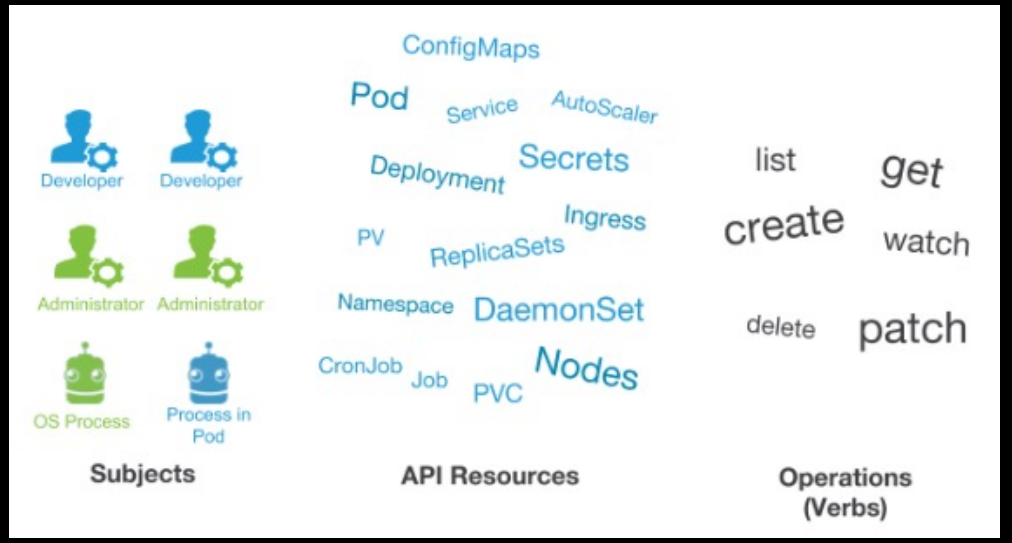
RBAC Basic Elements

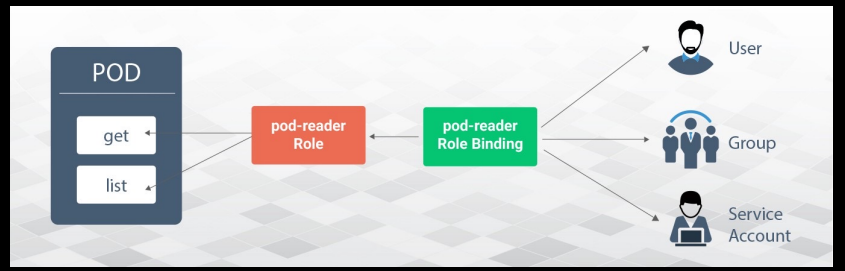
Rules
Operations (verbs) that can be carried out on a group of resources which belong to different API Groups.

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)

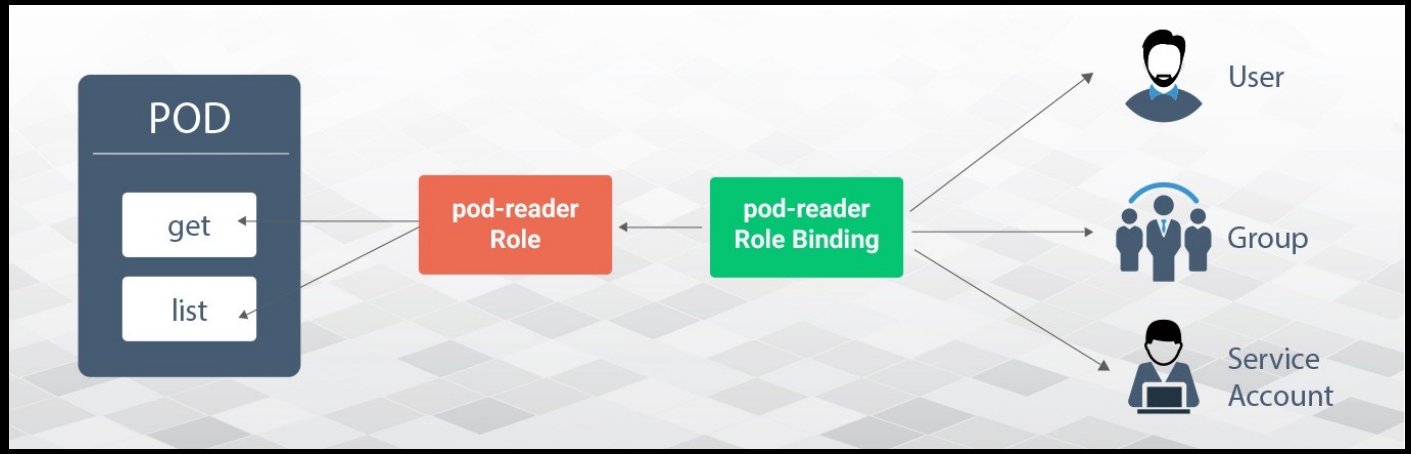
RoleBindings and **ClusterRoleBindings**
Bind subjects to roles





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

RBAC Example




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

```yaml
kind: RoleBinding
metadata:
  name: pod-reader
subjects:
- kind: Serviceaccount
  name: my-account
roleRef:
  kind: Role
  name: pod-reader
```

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

Troubleshooting
Debugging

Most common errors

```
❯ kubectl get pods
NAME                READY   STATUS             RESTARTS   AGE
nginx-pull-error    0/1     ImagePullBackOff   0          31s
nginx-crashloop     0/1     CrashLoopBackOff   2          27s
```

Most common errors

```
❯ kubectl get pods
NAME                READY   STATUS             RESTARTS   AGE
nginx-pull-error    0/1     ImagePullBackOff   0          31s
nginx-crashloop     0/1     CrashLoopBackOff   2          27s
```

ImagePullBackOff

1. The specified image is not present in the registry.
2. A typo in the image name or tag.
3. Image pull access denied from the given registry due to credential issue.

```
apiVersion: v1
kind: Deployment
metadata:
  name: nginx-xyz
spec:
…
 containers:
  - name: nginx-xyz
    image: 'nginx-fail:1.7.9'
 serviceAccountName: default
 imagePullSecrets:
 - name: default-dockercfg-8q42l
…
```

Most common errors

```
❯ kubectl get pods
NAME                READY   STATUS             RESTARTS   AGE
nginx-pull-error    0/1     ImagePullBackOff   0          31s
nginx-crashloop     0/1     CrashLoopBackOff   2          27s
```

CrashLoopBackOff

1.Check Logs
2.Check Probes
3.kubectl debug
4....

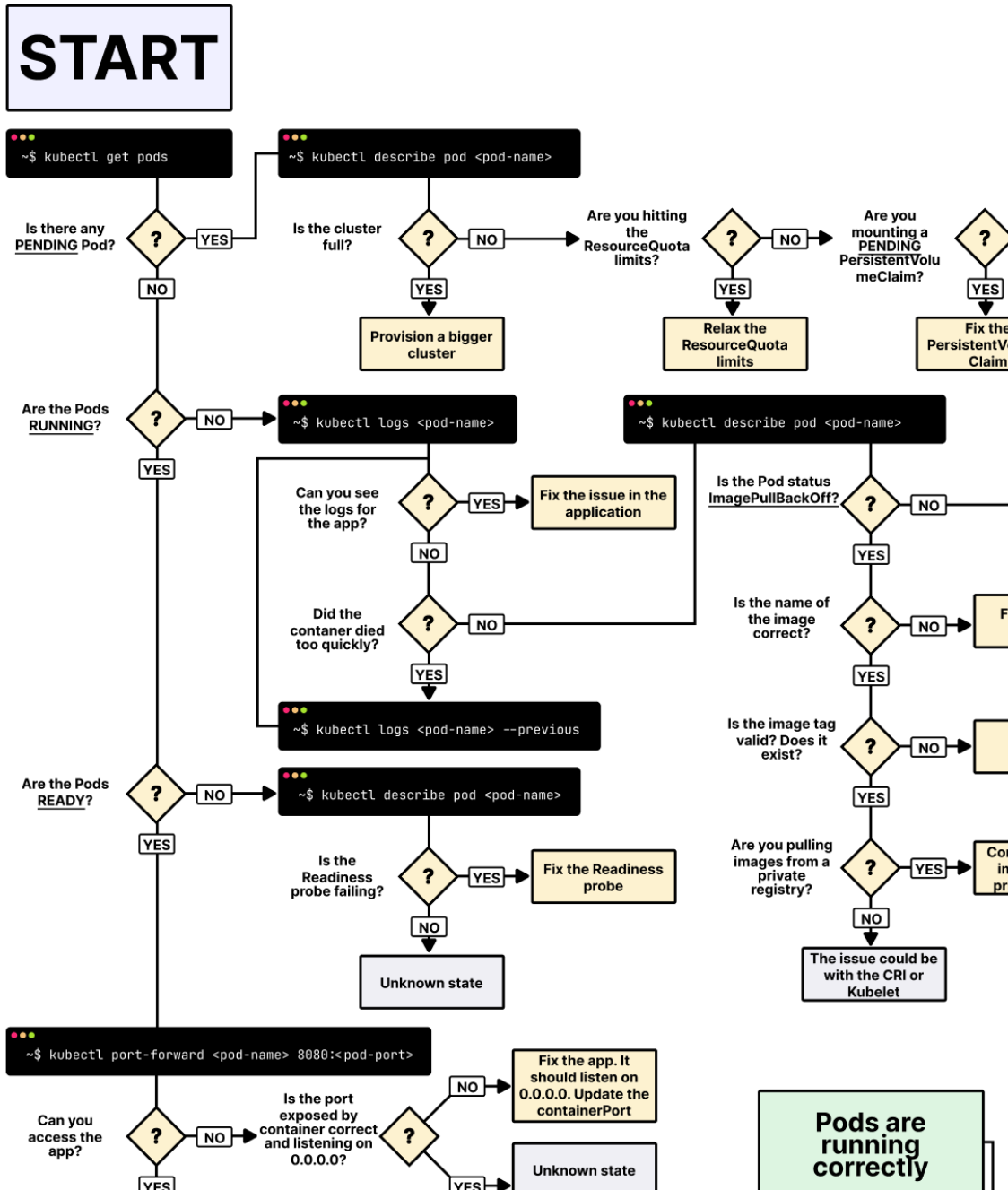
exec /usr/sbin/nginx: exec format error

```
java.lang.NullPointerException: null
at
org.springframework.boot.actuate.metrics.web.servlet.LongTaskTimingHandlerInterceptor.stop
TaskTimers(LongTaskTimingHandlerInterceptor.java:123) ~[spring-boot-actuator-
2.3.3.RELEASE.jar!/:2.3.3.RELEASE]
at
org.springframework.boot.actuate.metrics.web.servlet.LongTaskTimingHandlerInterceptor.afte
pletion(LongTaskTimingHandlerInterceptor.java:79) ~[spring-boot-actuator-
2.3.3.RELEASE.jar!/:2.3.3.RELEASE]
...
at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49) [tomcat
embed-core-9.0.37.jar!/:9.0.37]
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
[na:1.8.0_302]
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
[na:1.8.0_302]
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61) [tom
embed-core-9.0.37.jar!/:9.0.37]
at java.lang.Thread.run(Thread.java:748) [na:1.8.0_302]
2023-04-03 14:20:09.012 INFO 1 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryB
: Closing JPA EntityManagerFactory for persistence unit 'default'
```

Troubleshooting

A visual guide on troubleshooting Kubernetes deployments
https://learnk8s.io/troubleshooting-deployments

How To Troubleshoot Kubernetes Pods: Beginners
https://devopscube.com/troubleshoot-kubernetes-pods/



Dev/GitOps

DevOps – GitOps and DevSecOps

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

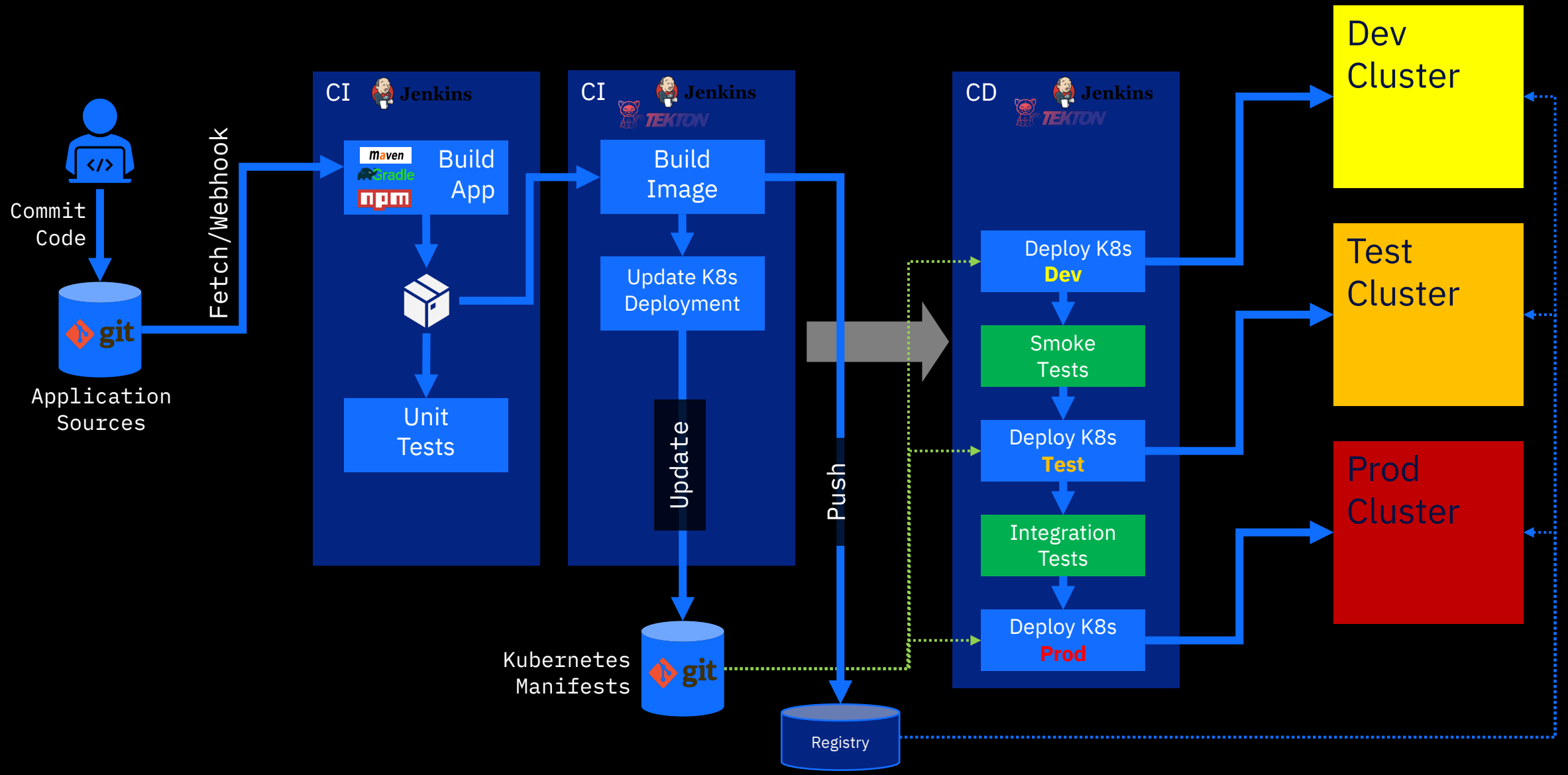
1. Having a completely automated delivery pipeline that can roll out changes to your infrastructure when changes are made to Git.
2. Everything has to be version controlled and stored in a single source of truth from which you can recover.

Dev**Sec**Ops means thinking about application and infrastructure security from the start.

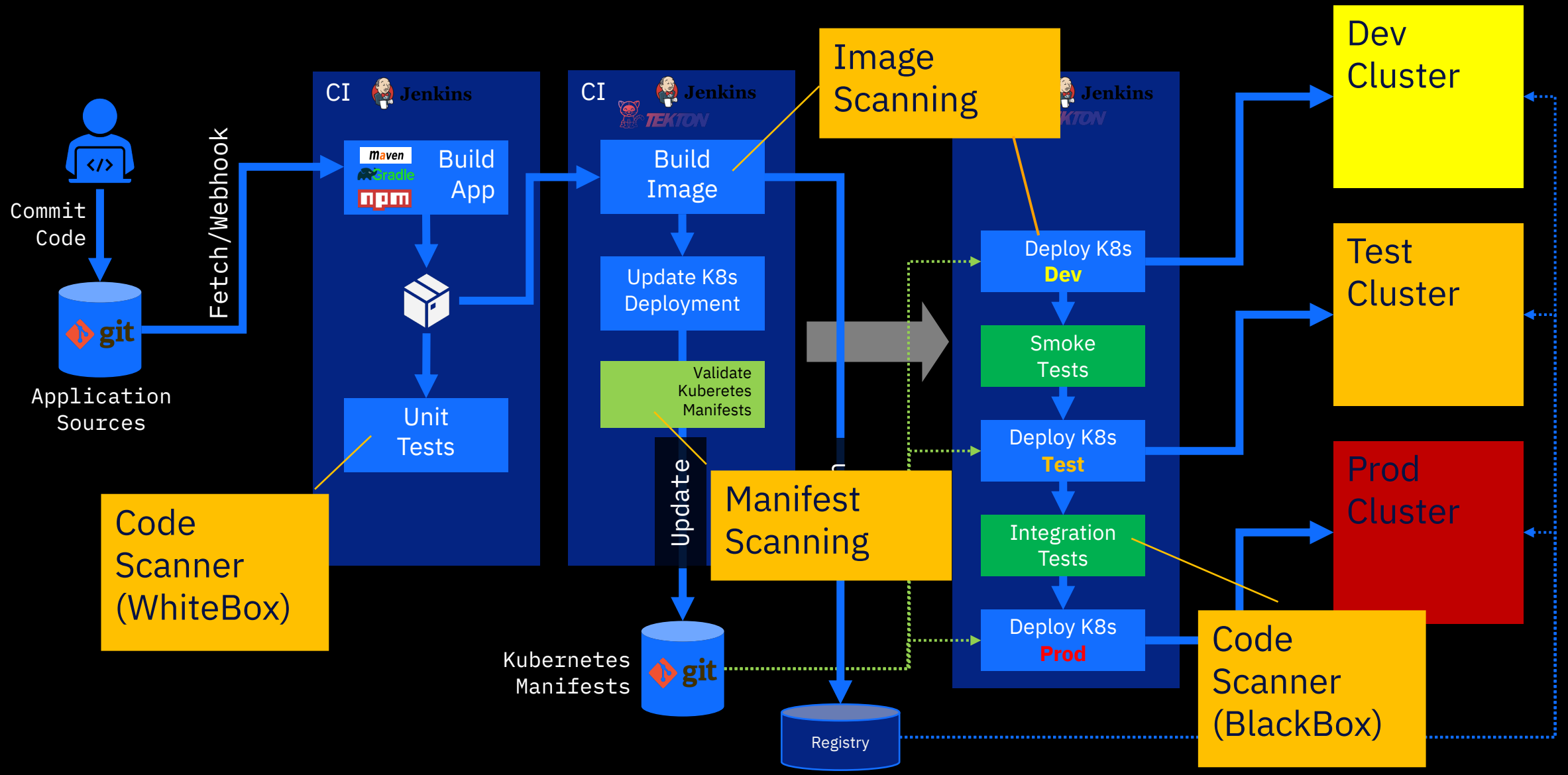
1. The idea is to get security back in to the lifecycle → shifting security left
2. The goal is to make security as silent and seamless as possible
3. «Everyone is responsible for security» mindset

- Developers write secure code
- SREs secure the environments

Kubernetes – CI/CD Pipeline - GitOps



Kubernetes – CI/CD Pipeline - GitOps



QUESTIONS?

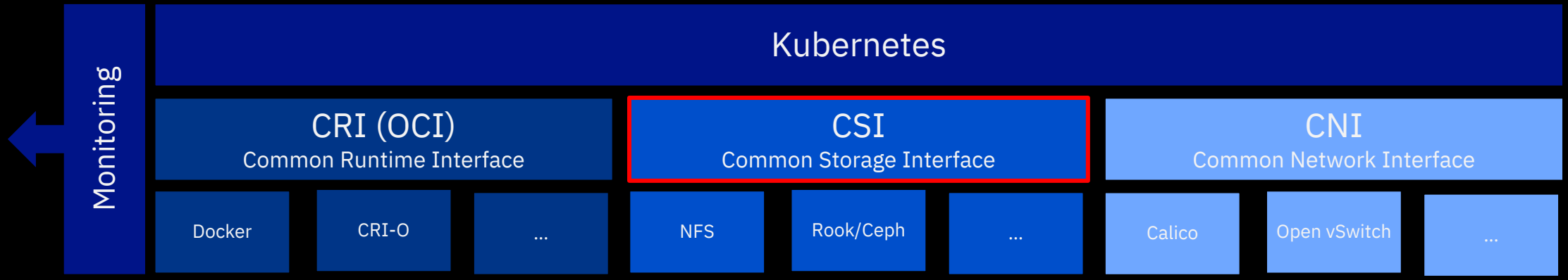
Kubernetes Workshop Series

Container State

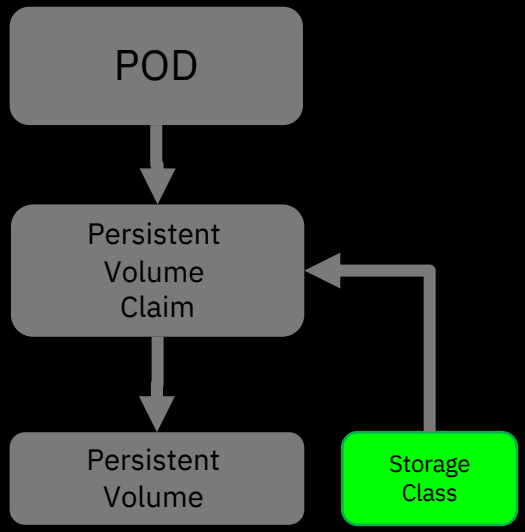
07

Kubernetes – Storage

The Common Storage Interface (CSI) provides an API for users and administrators that abstracts details of how storage is provided from how it is consumed.



Kubernetes – PersistentVolume subsystem



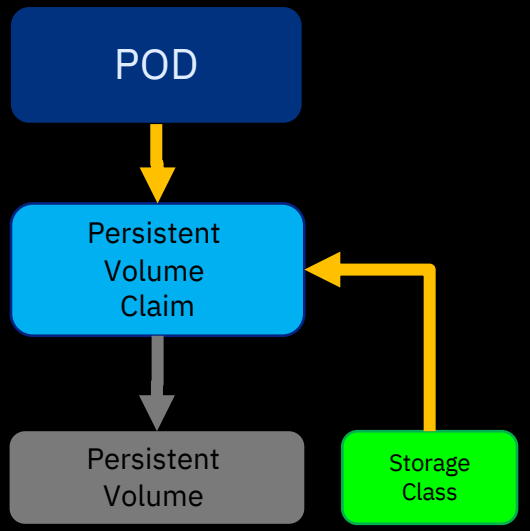
Storage Class

Provides a way for administrators to describe the “classes” of storage they offer.

Different classes might have different quality-of-service levels, backup policies, …

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: my-storage-class
  annotations:
    storageclass.beta.kubernetes.io/is-default-class: "true"
  labels:
    addonmanager.kubernetes.io/mode: Reconcile
provisioner: k8s.io/minikube-hostpath
```

Kubernetes – PersistentVolume subsystem



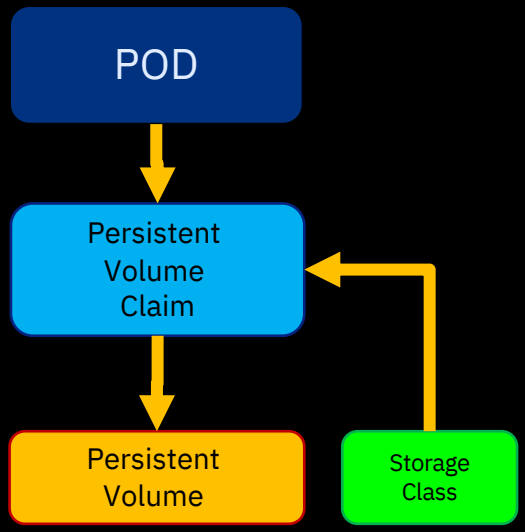
PersistentVolumeClaim (PVC)

Is a request for storage by a user.

- Pods attach to PVCs to persist data.
- PVCs can request specific size and access modes (e.g., they can be mounted once read/write or many times read-only).

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pv-claim
spec:
  storageClassName: my-storage-class
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Kubernetes – PersistentVolume subsystem



PersistentVolume (PV)

Is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes

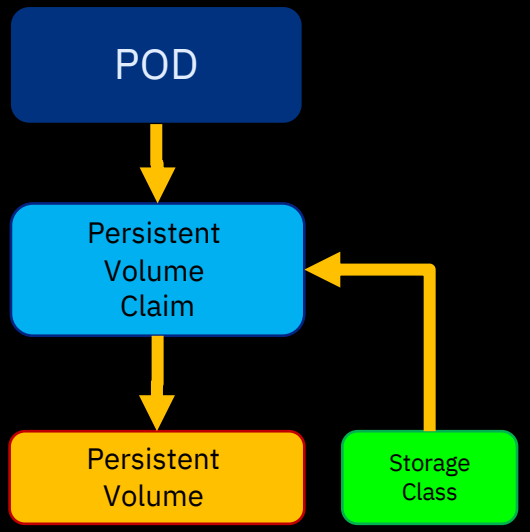
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv-volume
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

Ususally created dynamically except for NFS, hostPath and iSCSI.

There is an dynamic provisioner for NFS:

https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner

Kubernetes – PersistentVolume subsystem



Provisioning Persistent Volumes

There are two ways PVs can be provisioned:

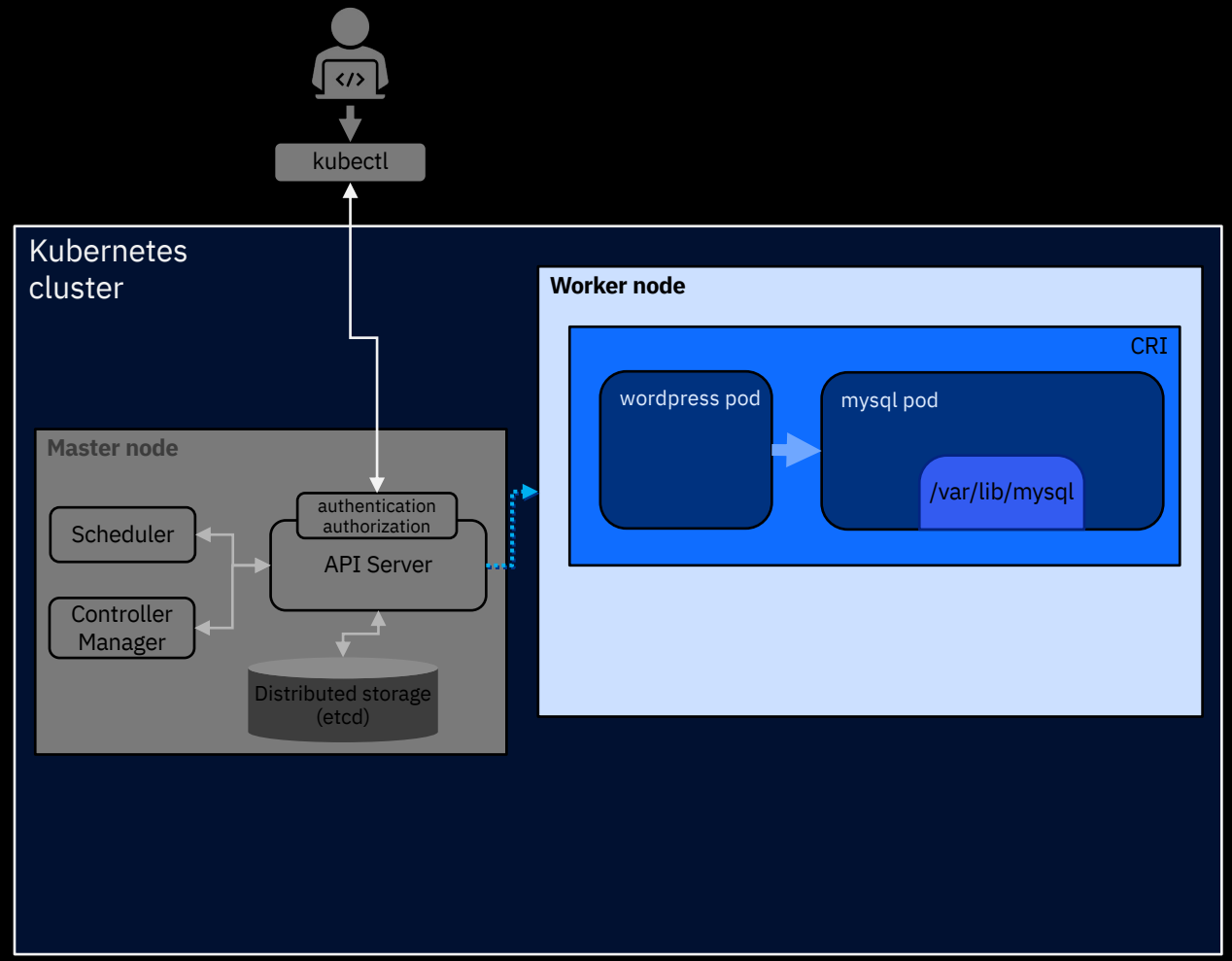
- **Static**
  A cluster administrator manually creates a number of PVs. They carry the details of the real storage, which is available for use by cluster users.

- **Dynamic**
  When none of the static PVs the administrator created match a user's PersistentVolumeClaim, the cluster may try to dynamically provision a volume specially for the PVC if there is a storage solution that supports it.

Persistence - Example

I have a Wordpress server with a mysql backend.

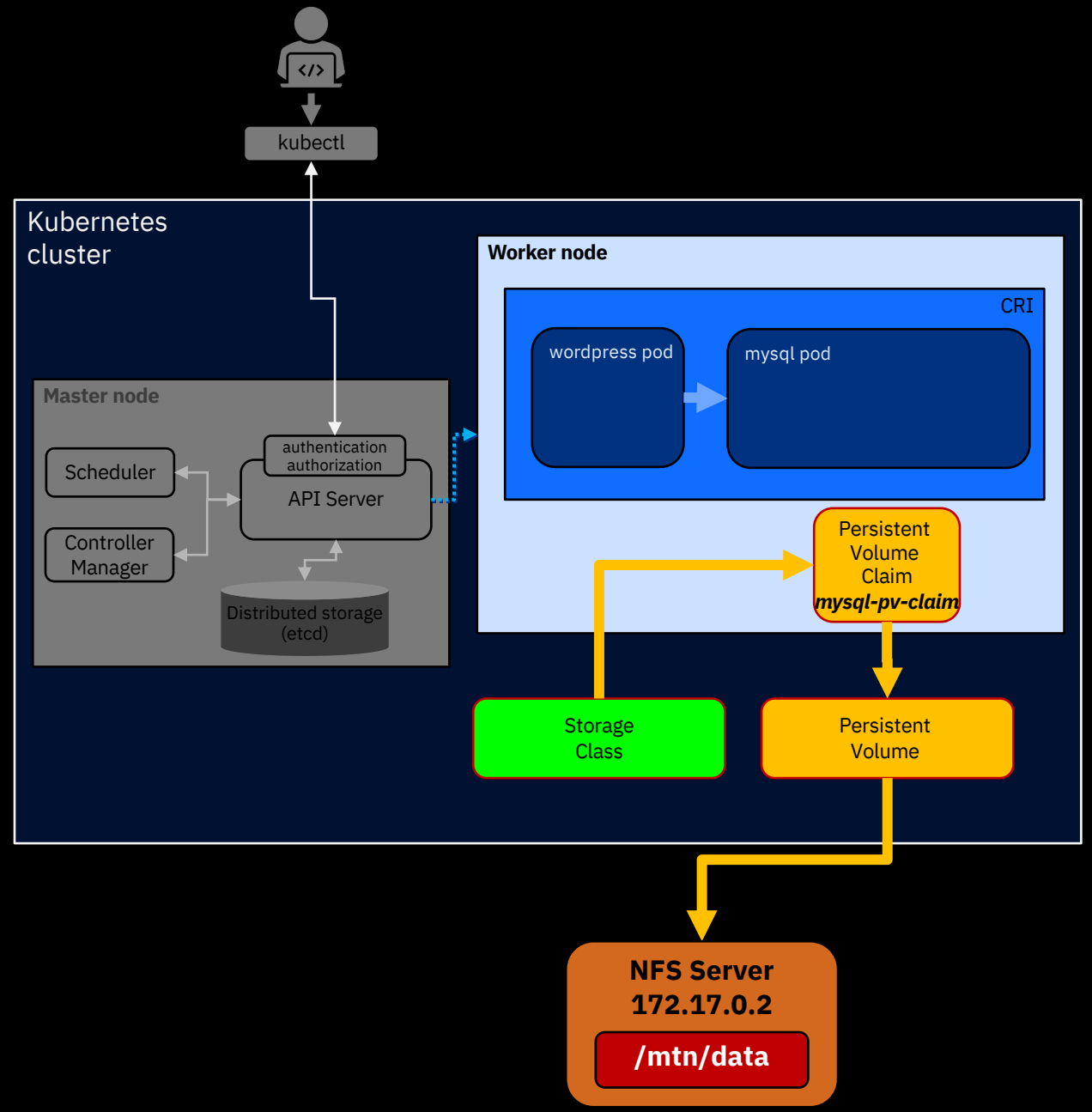
Each time the mysql pod restarts I lose all my data.



Persistence - Example

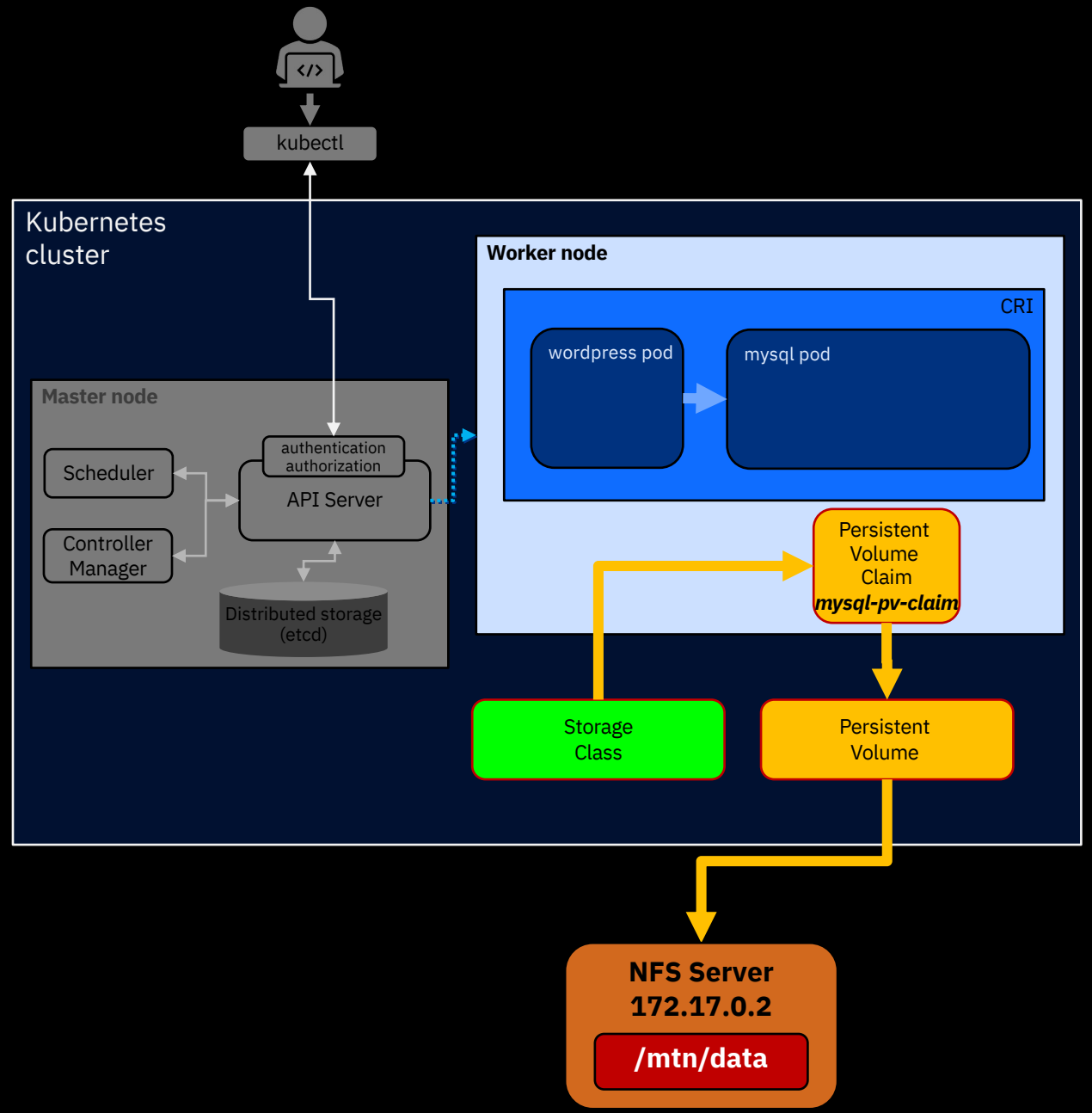
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv-volume
spec:
  capacity:
    storage: 20Gi
  accessModes:
  - ReadWriteOnce
  nfs:
    path: "/mnt/data"
    server: 172.17.0.2
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: wordpress
spec:
  storageClassName: nfs-client
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```



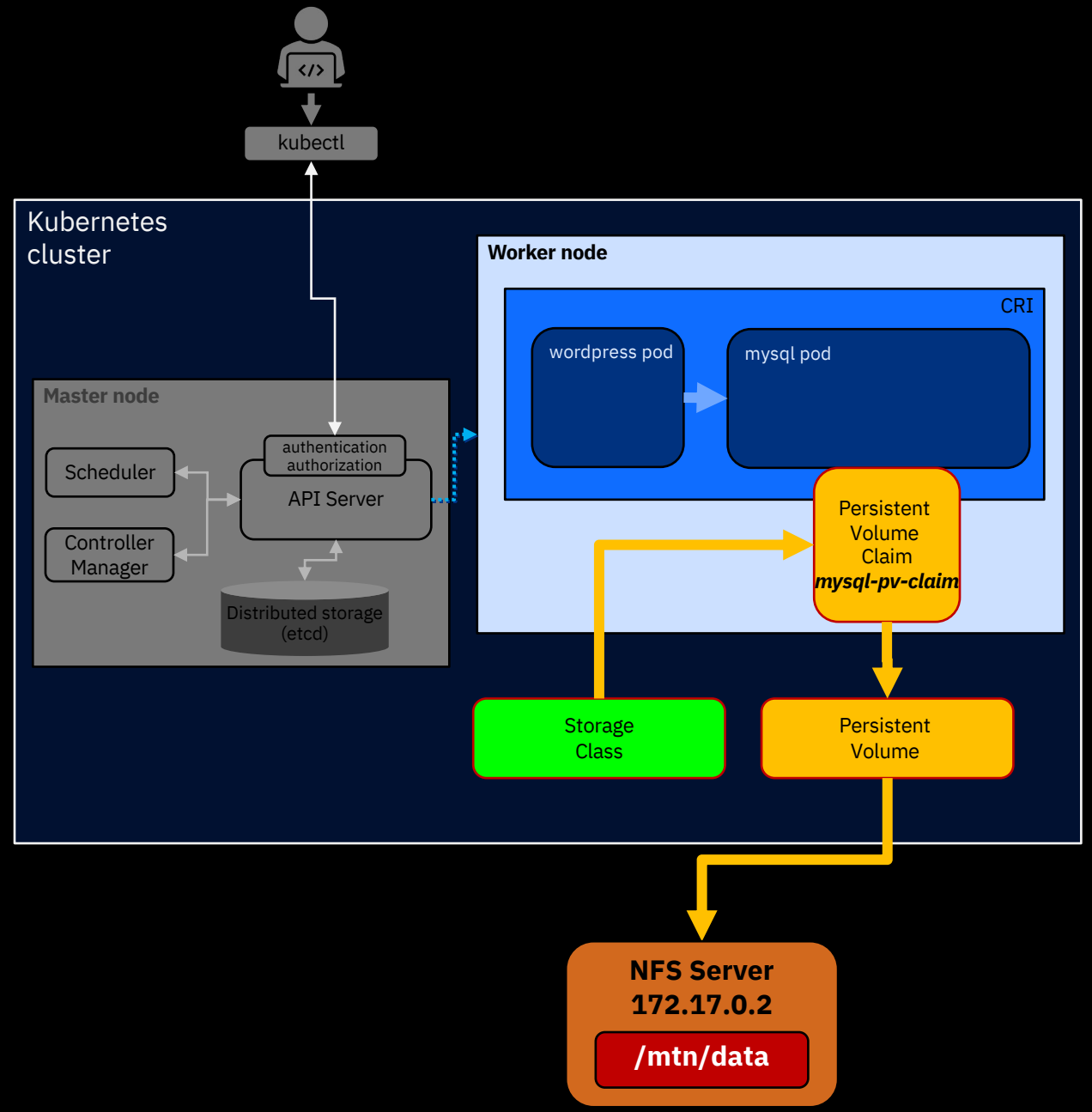
Persistence - Example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
spec:
  template:
    spec:
      containers:
      - image: mysql:5.6
        name: mysql

        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql

      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```



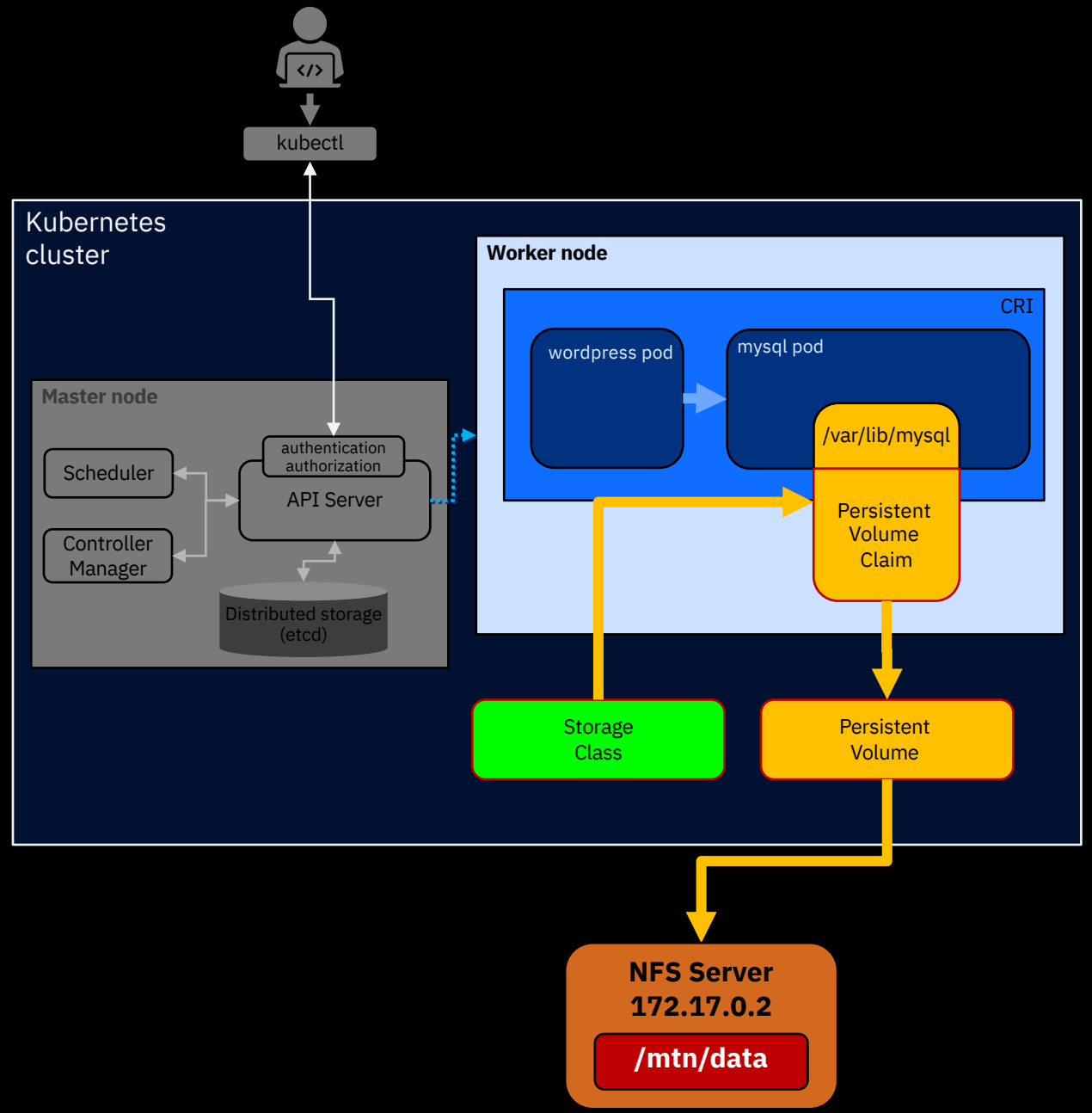
Persistence - Example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
spec:
  template:
    spec:
      containers:
      - image: mysql:5.6
        name: mysql

        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql


      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```



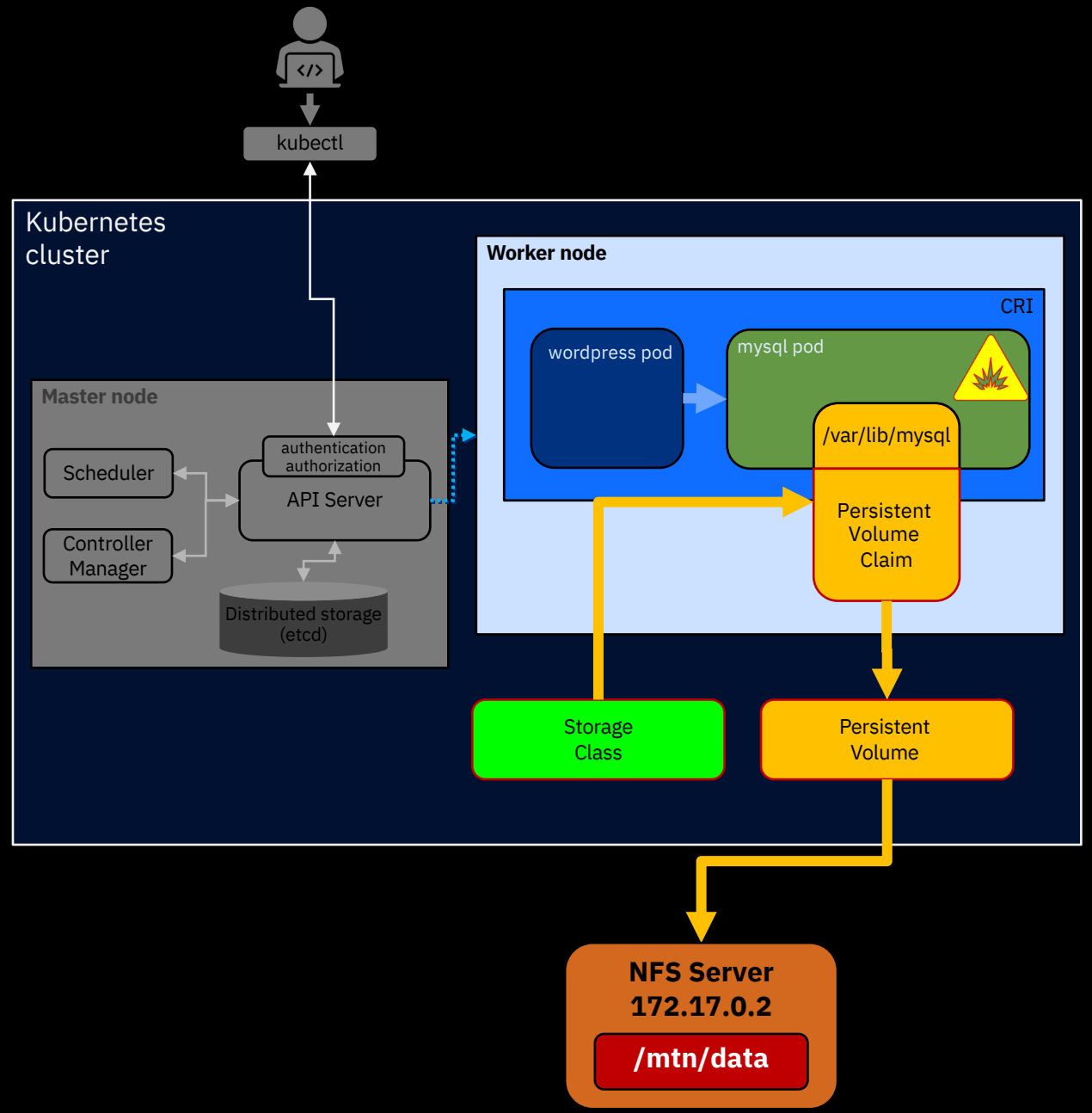
Persistence - Example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
spec:
  template:
    spec:
      containers:
      - image: mysql:5.6
        name: mysql

        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql

      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```



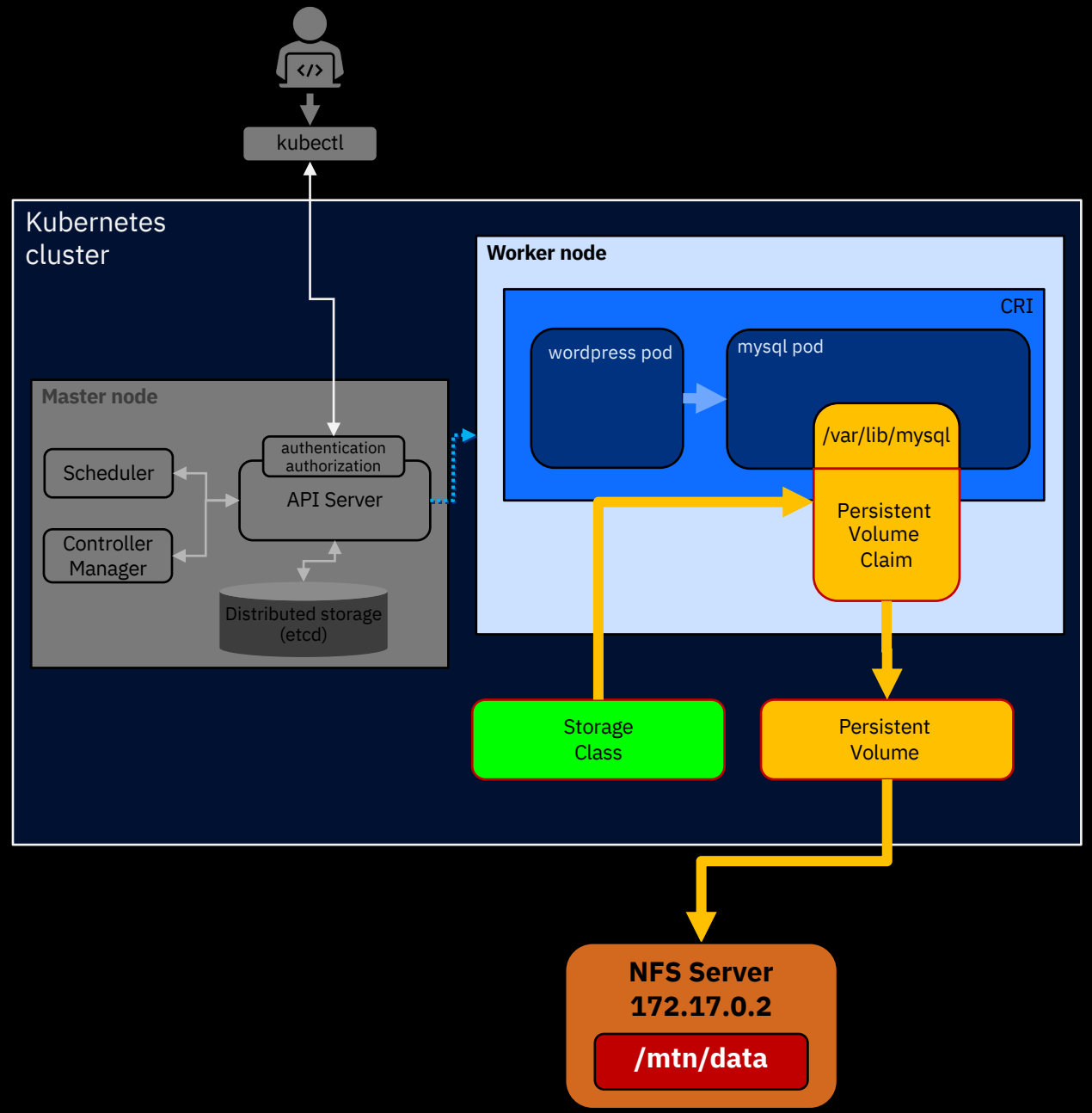
Persistence - Example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
spec:
  template:
    spec:
      containers:
      - image: mysql:5.6
        name: mysql

        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql

      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```



Persistence - Example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
spec:
  template:
    spec:
      containers:
      - image: mysql:5.6
        name: mysql

        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql

      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```



Kubernetes – PersistentVolume subsystem

Access Modes

ReadWriteOnce
The volume can be mounted as read-write by a single node

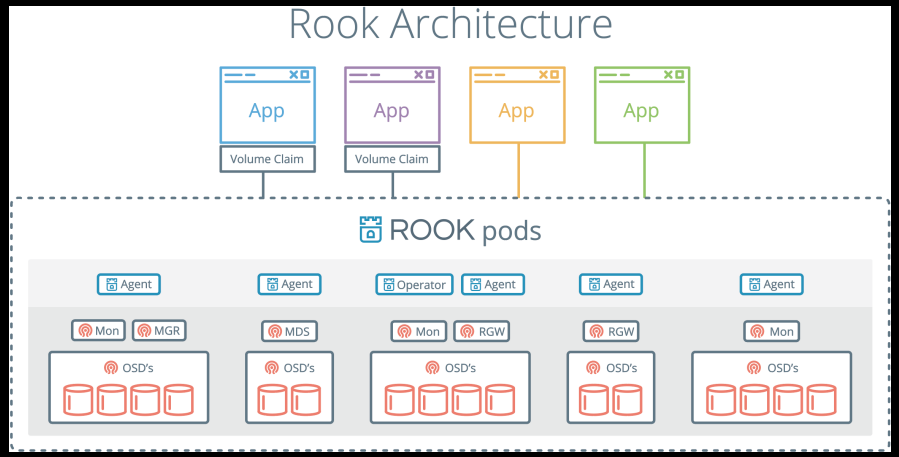
ReadOnlyMany
The volume can be mounted read-only by many nodes

ReadWriteMany
The volume can be mounted as read-write by many nodes

Rook/Ceph / ODF

Rook turns distributed storage systems into self-managing, self-scaling, self-healing storage services.

Ceph is a highly scalable distributed storage solution for **block storage**, **object storage**, and **shared file systems** with years of production deployments.

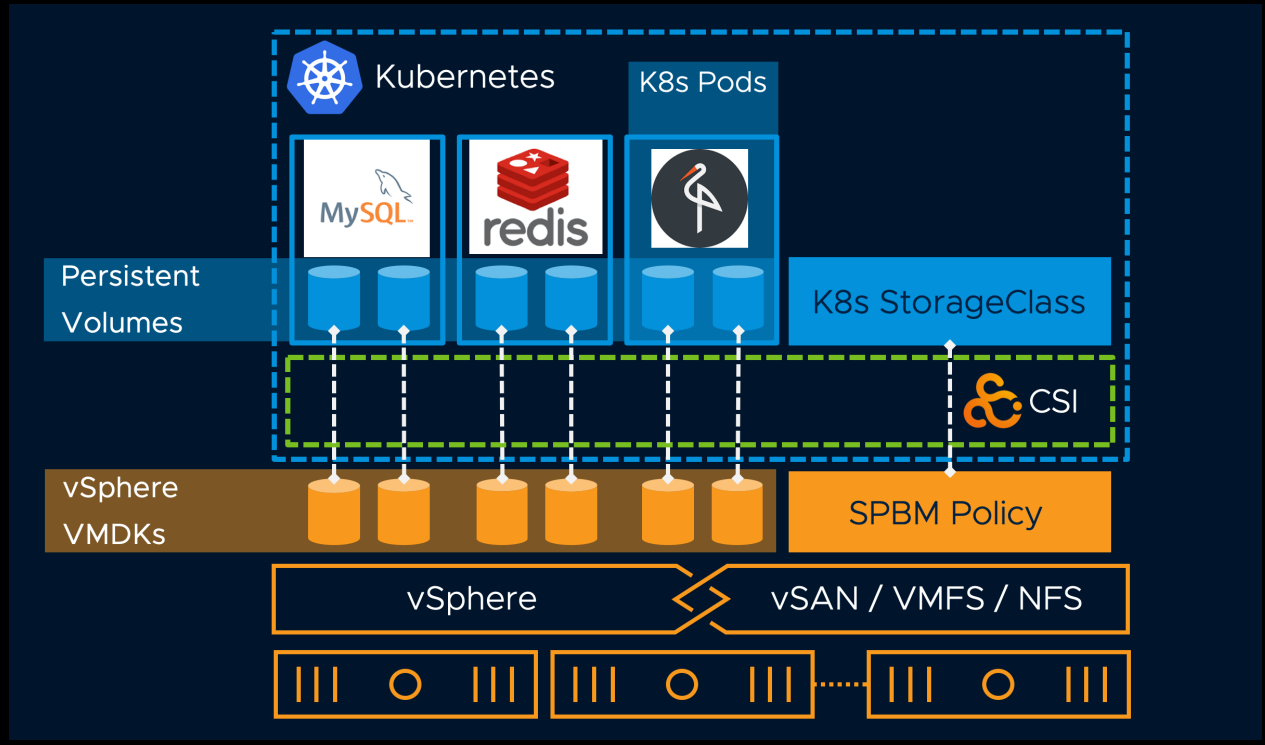


cassandra	crds: Set annotations pods, deployments and so on
ceph	We were defaulting to `ext4` at first and then moved to
cockroachdb	crds: Set annotations pods, deployments and so on
edgefs	edgefs image version update in examples and docs
minio	minio: add necessary update verb in minio RBAC
nfs	NFS: Fix operator.yaml line endings
noobaa	Rook-NooBaa Design Doc
yugabytedb	yugabytedb: Documentation, user guides & examples

https://rook.io/docs/rook/v0.9/ceph-quickstart.html

vSphere Container Storage

Cloud Native Storage (CNS) is a vSphere and Kubernetes (K8s) feature that makes K8s aware of how to provision storage on vSphere on-demand

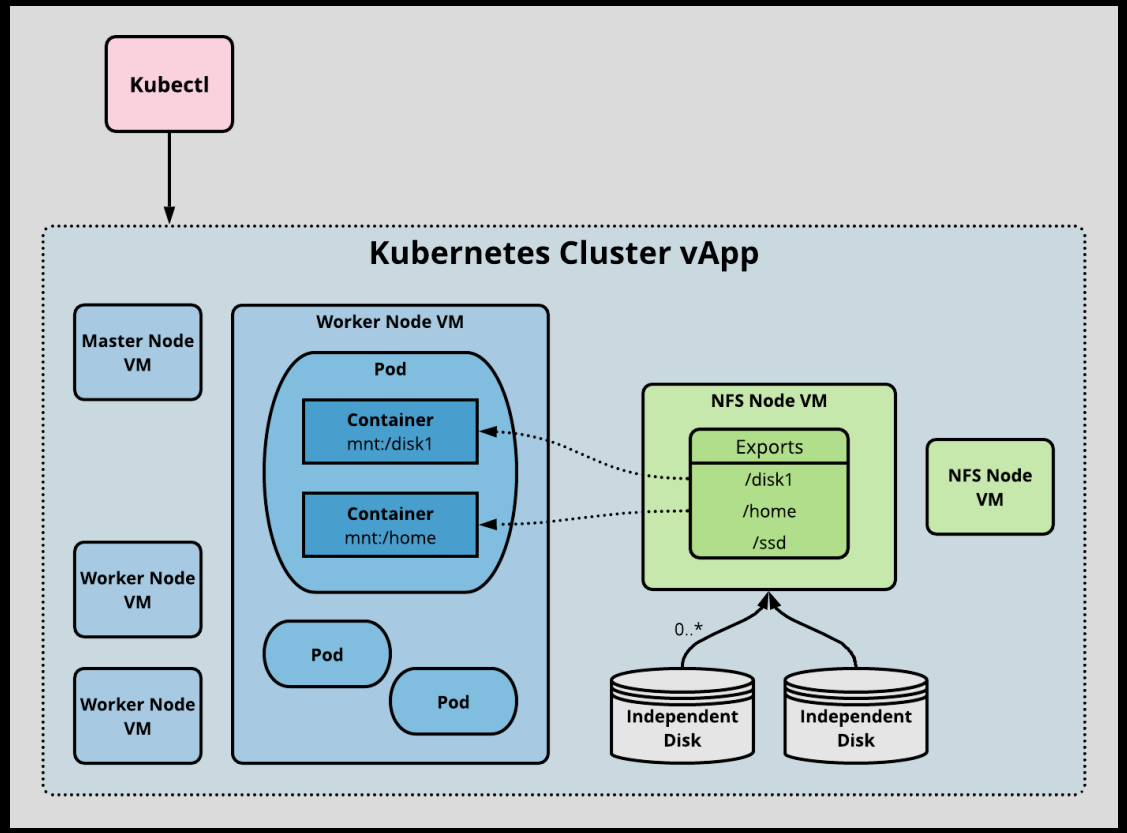


https://blogs.vmware.com/virtualblocks/2019/08/14/introducing-cloud-native-storage-for-vsphere/

NFS

NFS stands for Network File System – it's a shared filesystem that can be accessed over the network.

- The NFS server must already exist – Kubernetes doesn't run the NFS, pods in just access it.
- NFS provisioning is not dynamic by default:
  You have to create the PV manually and then the PVC.



QUESTIONS?

Kubernetes Workshop Series

Mesh Networking

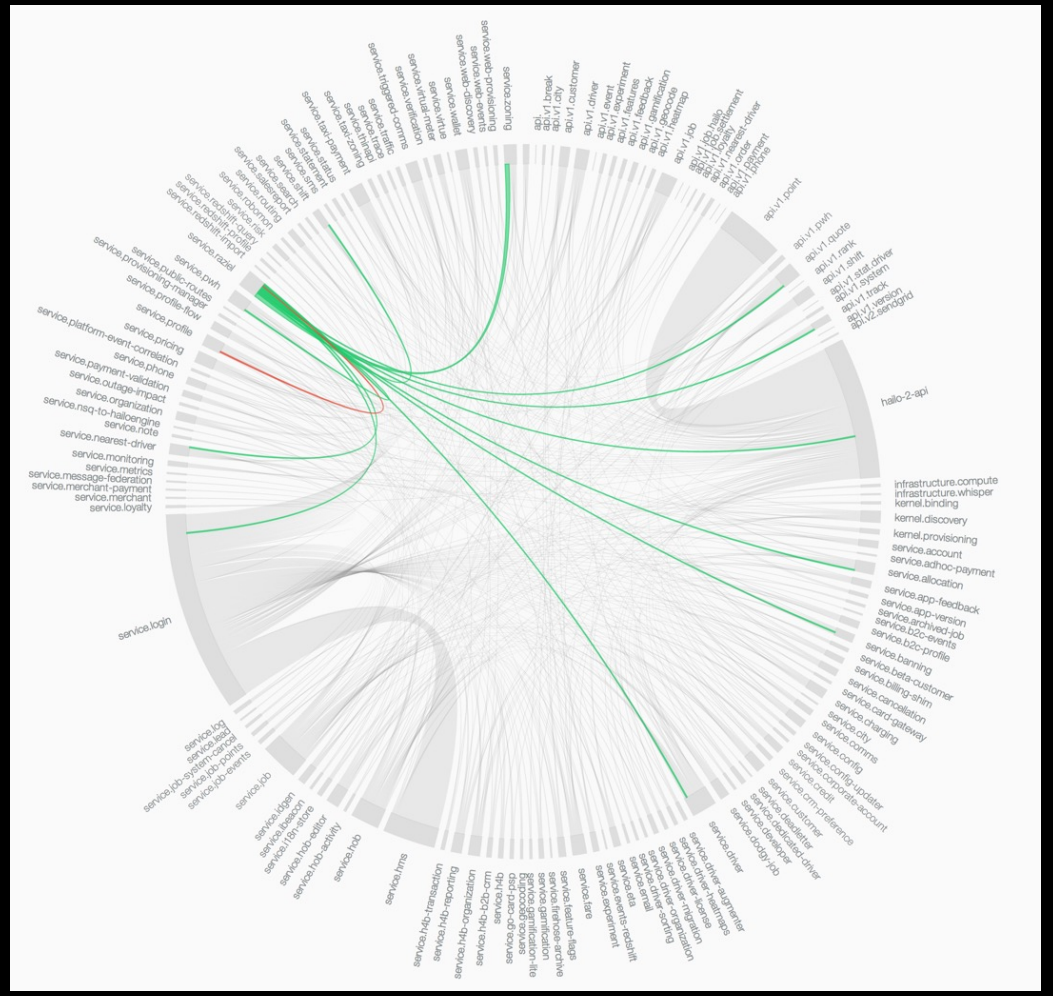
08

The trade off

Improved delivery velocity

in exchange for

increased operational complexity



Addressing DevOps Challenges

Istio

#	CHALLENGE	
CHALLENGE 1	ROLL OUT NEW VERSION WITHOUT DOWNTIME OR CHANGING CODE	
CHALLENGE 2	HOW TO DO CANARY TESTING	
CHALLENGE 3	HOW TO DO A/B TESTING	
CHALLENGE 4	THINGS DON'T ALWAYS GO CORRECTLY IN PRODUCTION...	
CHALLENGE5	HOW CAN I LIMIT RATE FOR SOME OF MY SERVICES?	
CHALLENGE 6	I NEED TO VIEW AND MONITOR WHAT IS GOING ON WHEN CRISIS ARISES	
CHALLENGE 7	HOW CAN I SECURE MY SERVICES?	

Service Mesh

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

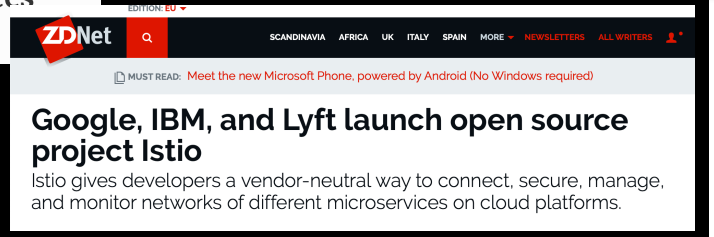
Istio

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

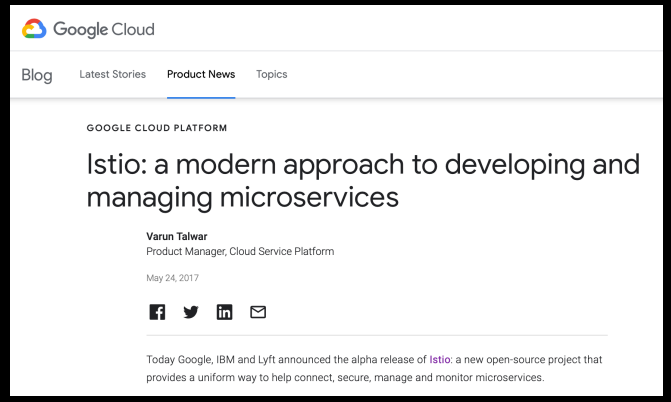

ZDNet

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

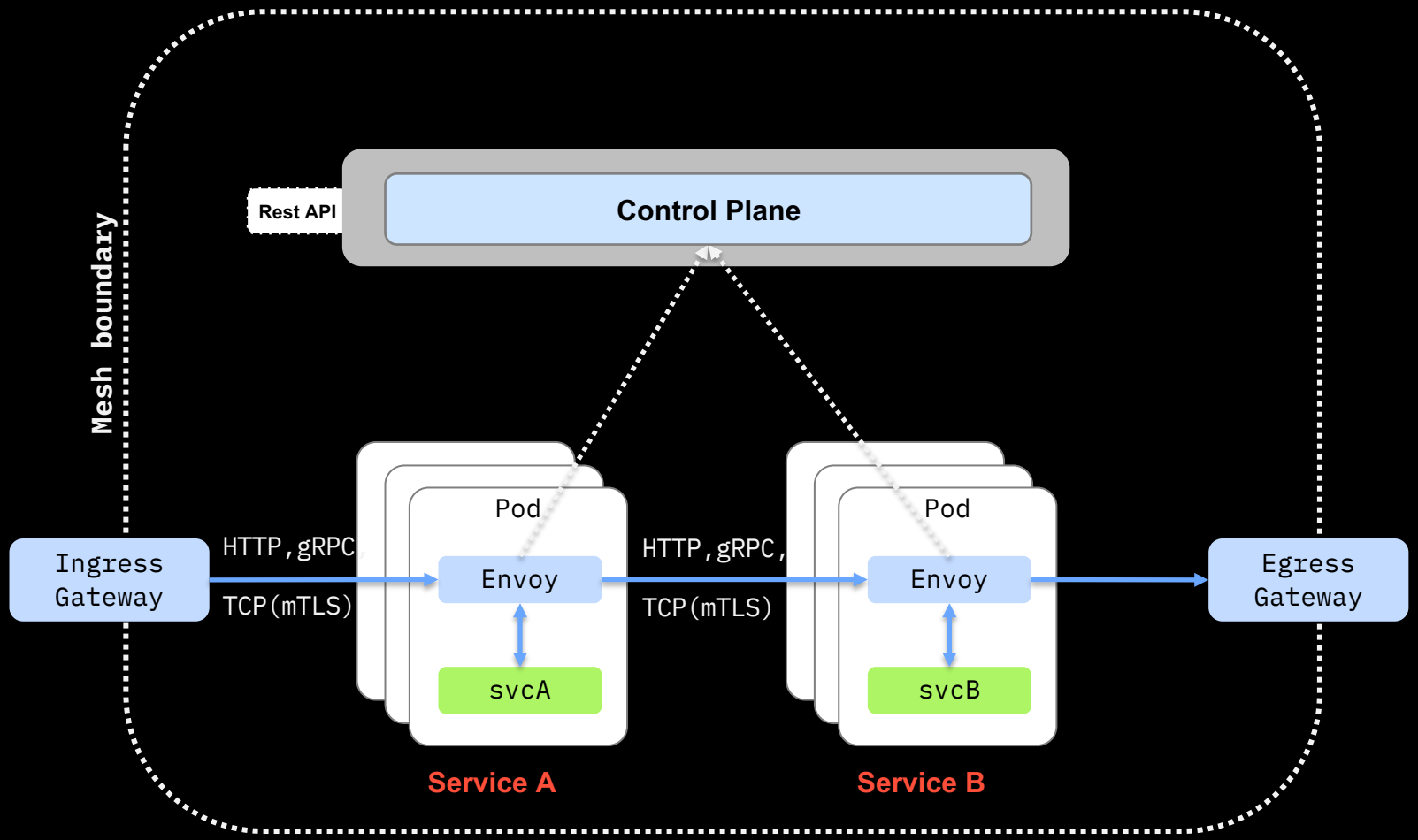
Today Google, IBM and Lyft announced the alpha release of Istio: a new open-source project that provides a uniform way to help connect, secure, manage and monitor microservices.

Launched in May 2017 by Google, Lyft and IBM

Open Source

Zero Code Changes

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
Gateway

Load balancer operating at the edge of the mesh receiving incoming HTTP/TCP connections

- Configures ports to expose externally
- Maps each exposed port to a request destination
- Each gateway can have one or more Virtual Services that defines these request destinations

Gateway is attached to Istio Ingress Controller (which can be the Kubernetes Ingress Controller)

Request destinations

- Ports for the gateway to expose and hosts for the corresponding services
- Attributes: **servers**

```yaml
kind: Gateway
metadata:
  name: helloworld-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - hosts:
    - myapp.demo.com
    port:
      name: http
      number: 80
      protocol: HTTP
```

ISTIO

Custom resource definitions

Istio

```
kind: Gateway
metadata:
  name: helloworld-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - hosts:
    - myapp.demo.com
    port:
      name: http
      number: 80
      protocol: HTTP
```

http://myapp.demo.com

POD
helloworld

version = **v1**

SERVICE
helloworld

selector
  app: helloworld

ISTIO
Virtual Service

Request sources
- Hosts that sources can invoke
- Attributes: **hosts** and **gateways**

Route destinations
- Subset of the destination
- Attributes: **route** and **destination**

Protocol selection
- How to connect to the destination subset
- Attributes: **http**, **tcp**, **tls**

Routing rules
- Additional routing attributes, applied for the route destinations
- Attributes: **weight** and **match**

HTTP traffic policy
- Protocol-specific connection quality of service
- Attributes: **timeout**, **retries**, **fault**, **rewrite**, and **redirect**

Istio

```
kind: VirtualService
metadata:
  name: helloworld
spec:
  hosts:
  - myapp.demo.com
  gateways:
  - helloworld-gateway
  http:
  - match:
    - uri:
        exact: /demo
    route:
    - destination:
        host: helloworld
```

ISTIO

Custom resource definitions

Ingress Configuration

```
kind: Gateway
metadata:
  name: helloworld-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - hosts:
    - myapp.demo.com
    port:
      name: http
      number: 80
      protocol: HTTP
```

URL Routing

```
kind: VirtualService
metadata:
  name: helloworld
spec:
  hosts:
  - myapp.demo.com
  gateways:
    helloworld-gateway
  http:
  - match:
    - uri:
        exact: /demo
    route:
    - destination:
        host: helloworld
```

POD
helloworld

version = **v1**

SERVICE
helloworld

selector
app: helloworld

`helloworld.namespace.svc.cluster.local`

`http://myapp.demo.com/demo`

ISTIO
Destination Rule

Istio

Destination host
- Host that route destinations can select
- Attribute: **host**

Host subset
- Identify a subset of service endpoints
- Attribute: **labels**

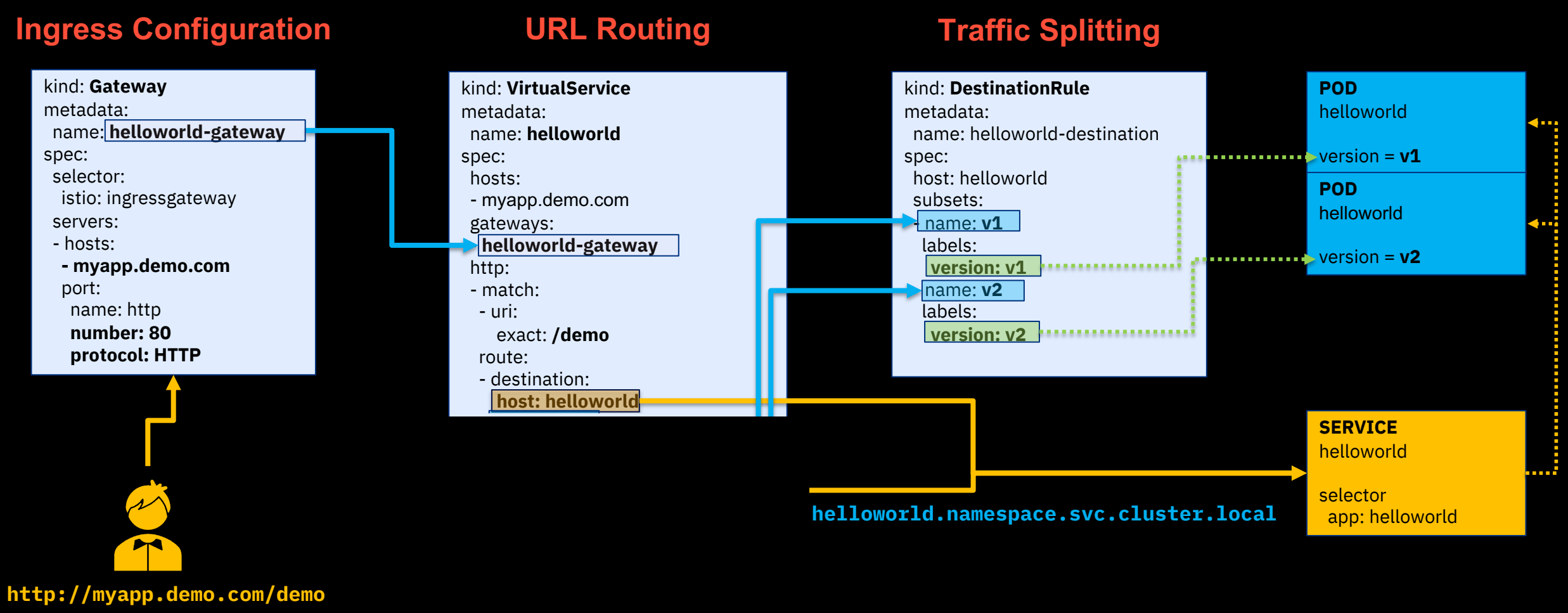
Traffic policy
- Influence expected quality of service for destinations
- Attributes: **trafficPolicy**
  loadBalancer,connectionPool, outlierDetection, and tls

Traffic policy can be applied to a port
- Attribute: **portLevelSettings**

```
kind: DestinationRule
metadata:
  name: helloworld-destination
spec:
  host: helloworld
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  trafficPolicy:
    tls:
      mode: SIMPLE
    connectionPool:
      tcp:
        maxConnections: 100
```

ISTIO

Custom resource definitions



ISTIO
Virtual Service

Request sources
- Hosts that sources can invoke
- Attributes: **hosts** and **gateways**

Route destinations
- Subset of the destination
- Attributes: **route** and **destination**

Protocol selection
- How to connect to the destination subset
- Attributes: **http**, **tcp**, **tls**

Routing rules
- Additional routing attributes, applied for the route destinations
- Attributes: **weight** and **match**

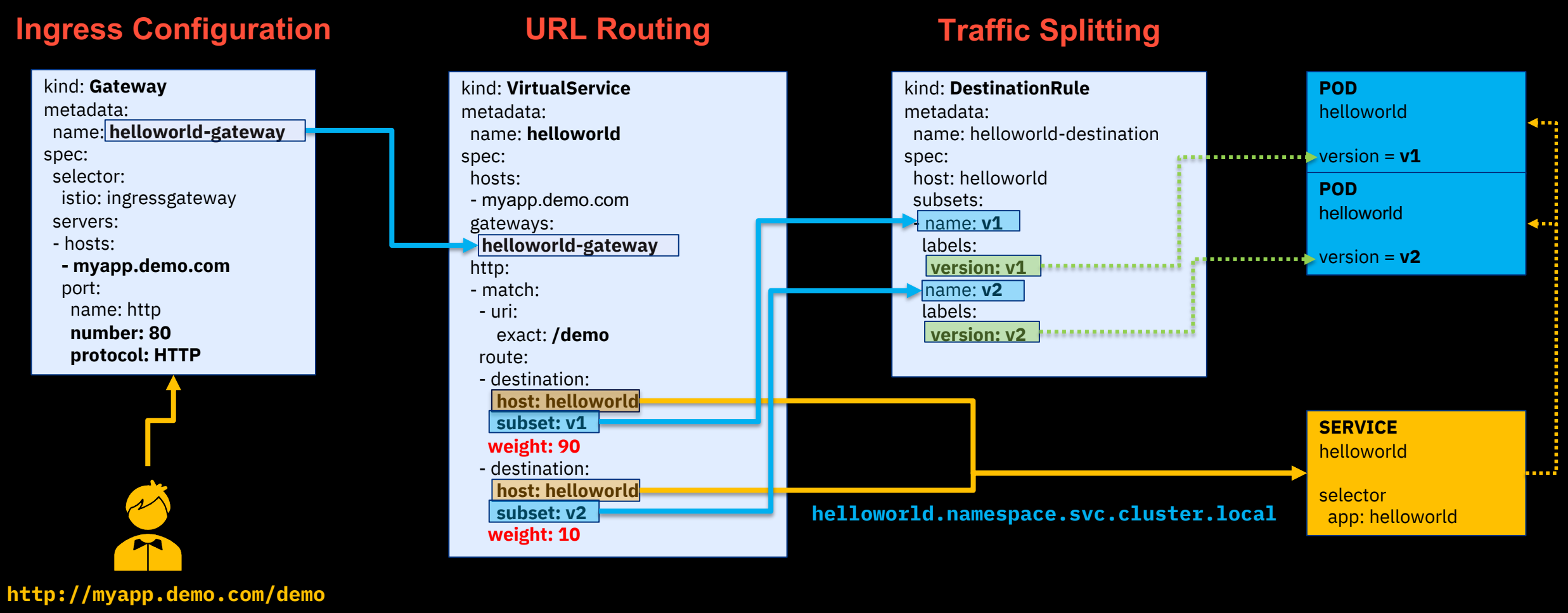
HTTP traffic policy
- Protocol-specific connection quality of service
- Attributes: **timeout**, **retries**, **fault**, **rewrite**, and **redirect**

Istio

```
kind: VirtualService
metadata:
  name: helloworld
spec:
  hosts:
  - myapp.demo.com
  gateways:
  - helloworld-gateway
  http:
  - match:
    - uri:
        exact: /demo
    route:
    - destination:
        host: helloworld
        subset: v1
      weight: 90
    - destination:
        host: helloworld
        subset: v2
      weight: 10
```

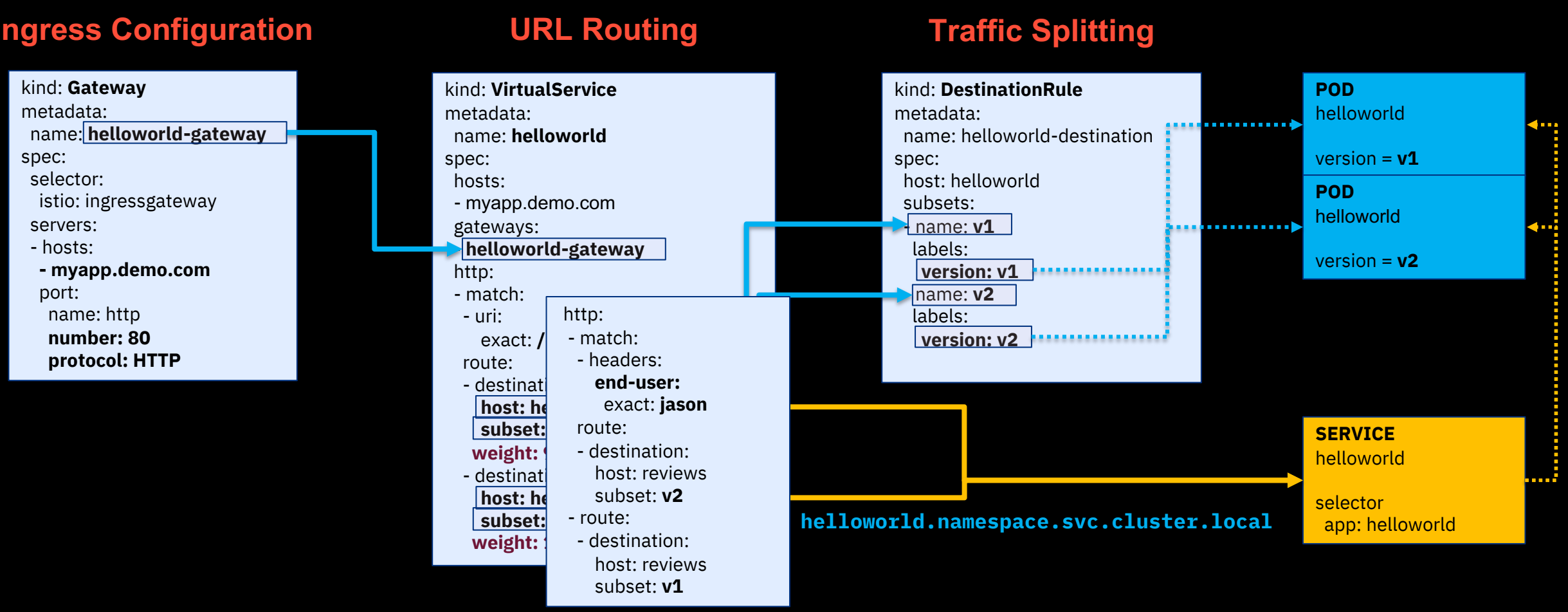
ISTIO

Custom resource definitions



ISTIO

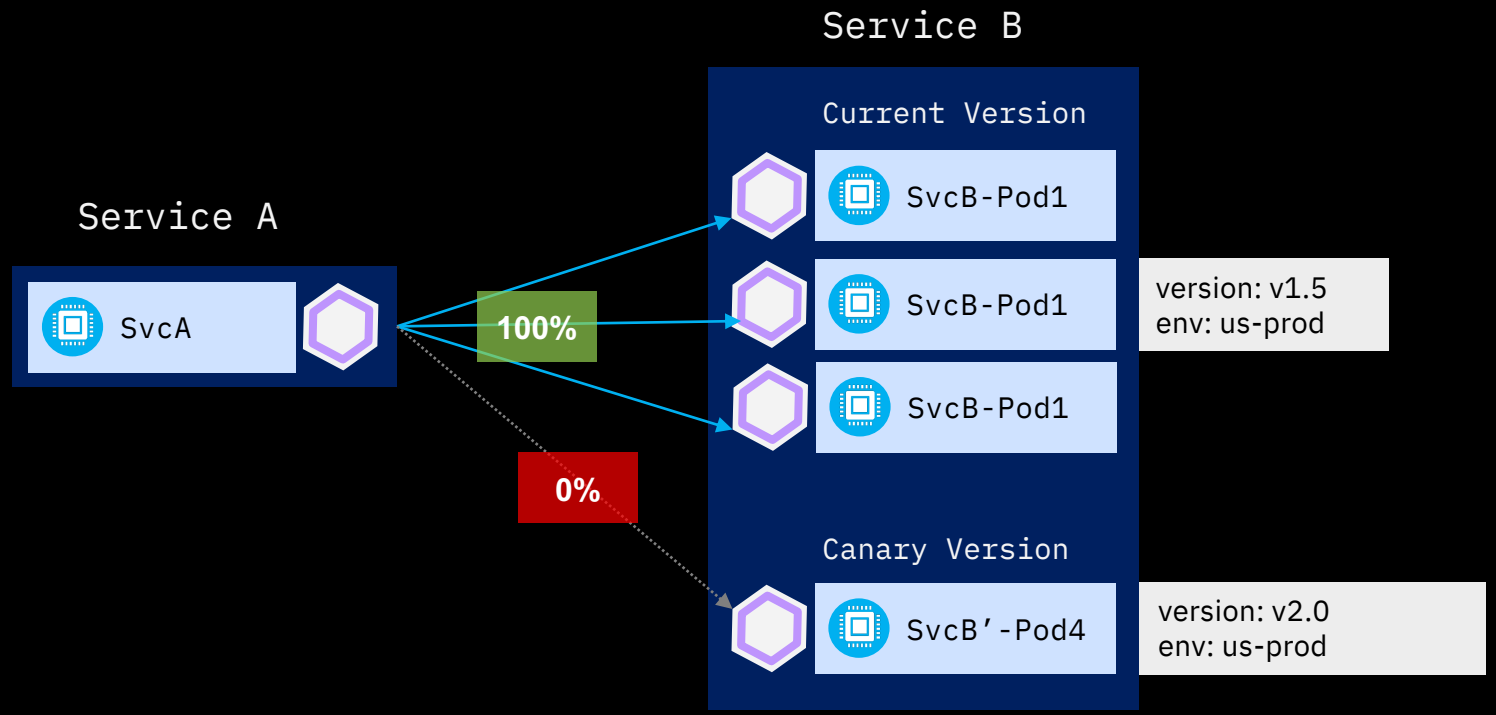
Custom resource definitions



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

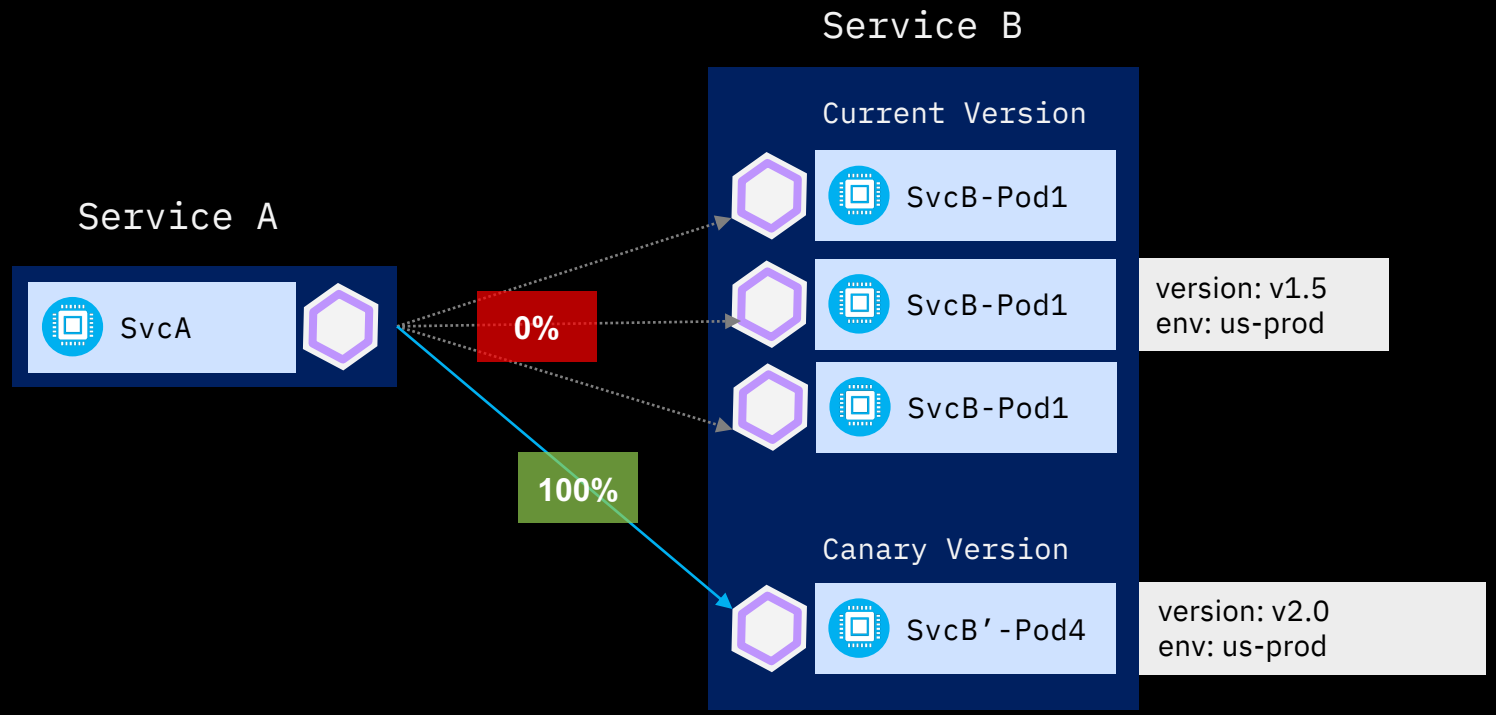
Traffic Control




```
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

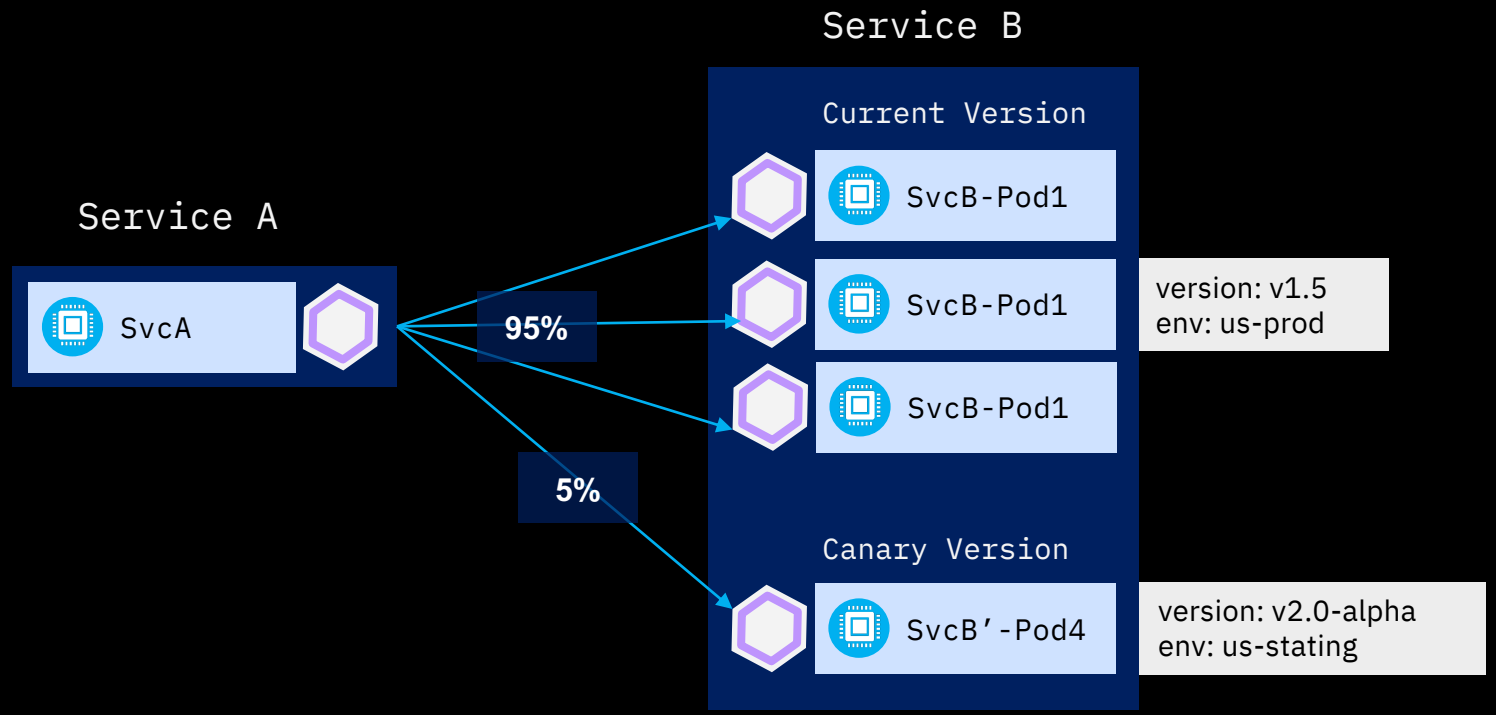



```
kind: VirtualService
destination:
  serviceB.cluster.local
match:
  source:
    serviceA.cluster.local
route:
  - labels:
    version: v2.0
    env: us-prod
  weight: 100
```

Update

CHALLENGE 1
ROLL OUT NEW VERSION WITHOUT DOWNTIME OR CHANGING CODE

Traffic Splitting

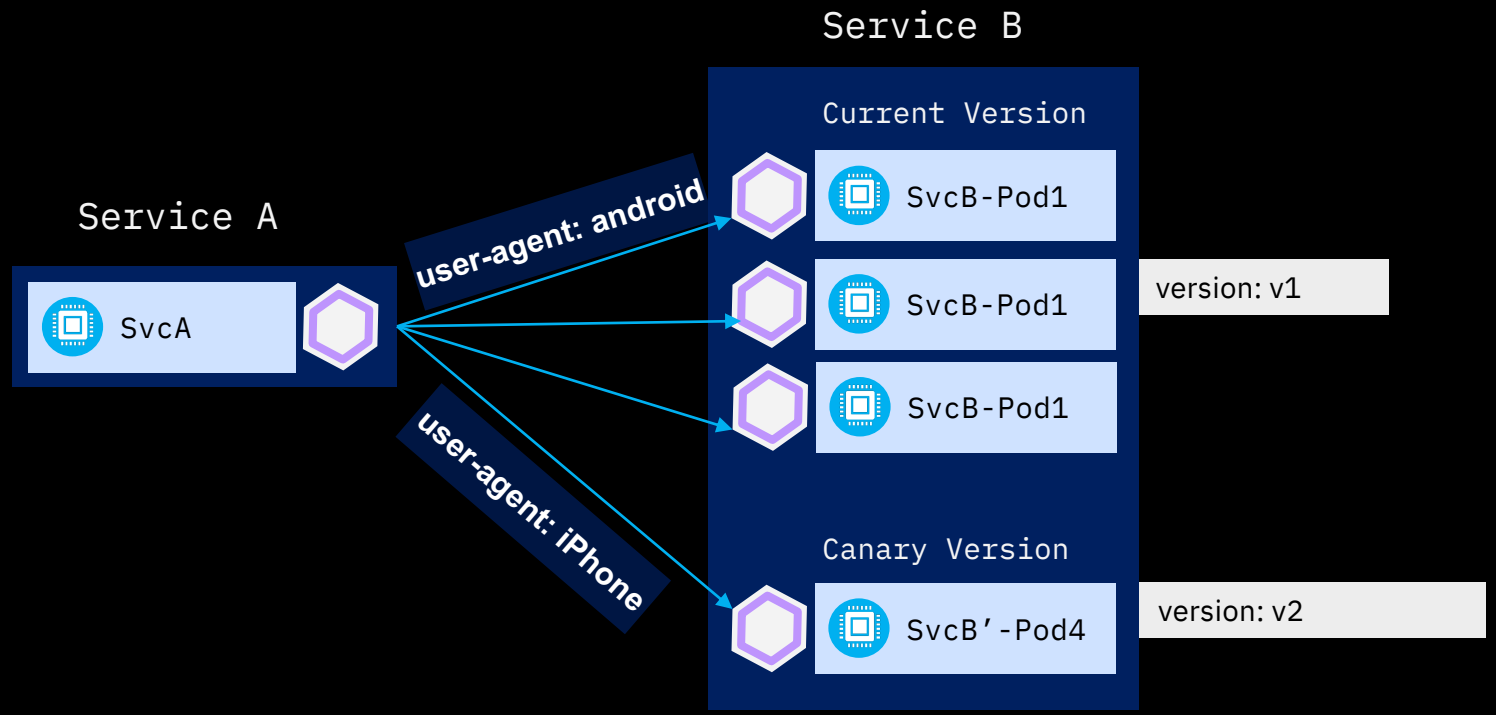



```
kind: VirtualService
destination:
  serviceB.cluster.local
match:
  source:
    serviceA.cluster.local
route:
  - labels:
    version: v1.5
    env: us-prod
  weight: 95
  - labels:
    version: v2.0-alpha
    env: us-staging
  weight: 5
```

CHALLENGE 2
HOW TO DO CANARY TESTING

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

Traffic Steering

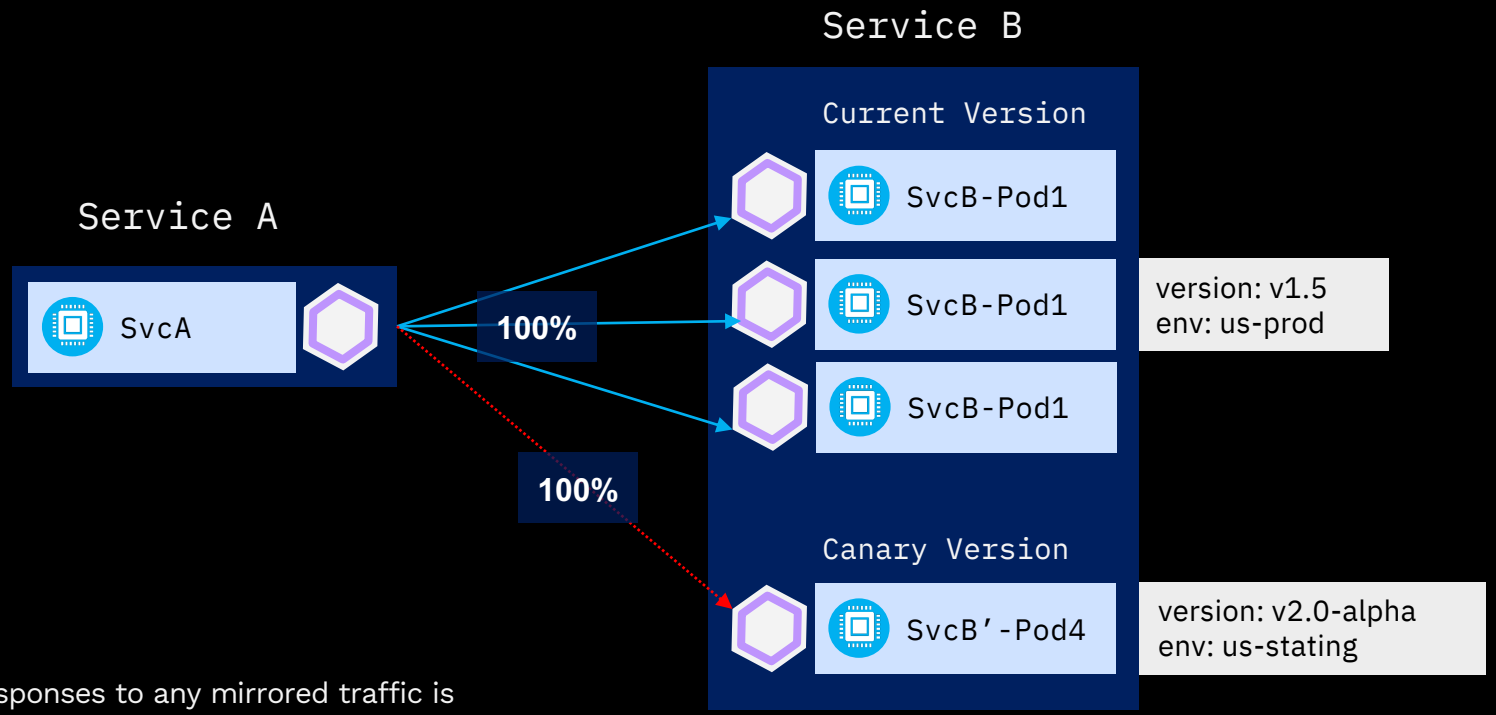



```
kind: VirtualService
destination:
 serviceB.cluster.local
match:
  httpHeaders:
    user-agent:
      regex:
        ^(.*?;)?(iPhone)(;.*)?$
 route:
- labels:
  version: v2
```

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...

Resiliency
Circuit Breakers

Istio

Connection pool

- Limits connections for reviews to invoke ratings
- **Limited to 1 concurrent connection, 1 request per connection (One concurrent request total)**
  - While requests are using all of the connections in a pool, any new requests are rejected

Outlier detection

- **If there are 3 requests in 2 seconds, reviews will be ejected for 3 minutes**
  - Request 1 will take more than 2 seconds, blocking the connection during that time
  - Request 2 won't get a connection, which will generate the first error
  - Request 3 won't get a connection, which will generate the second error, causing ejection

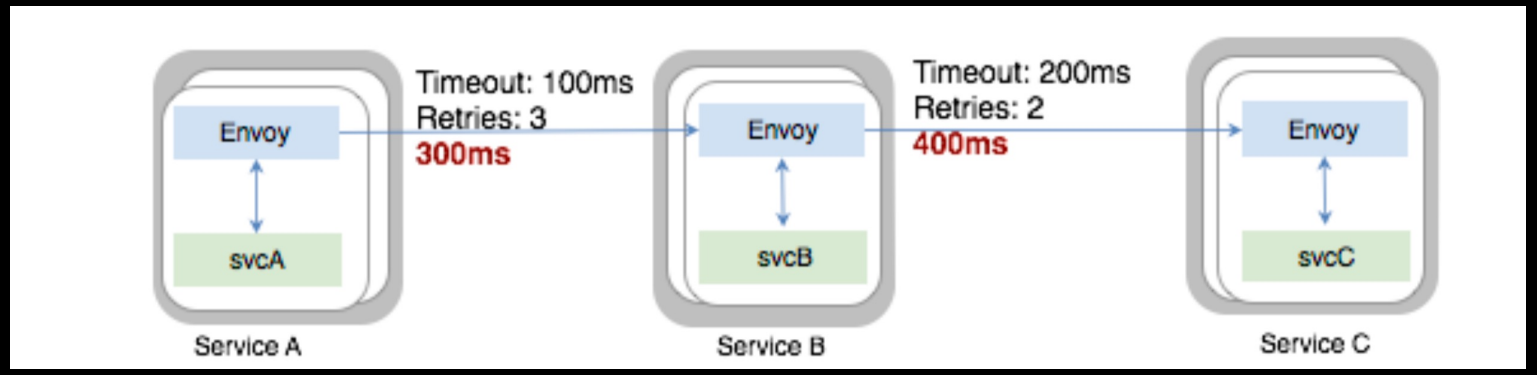
```
kind: DestinationRule
host: reviews
trafficPolicy:
  connectionPool:
    tcp:
      maxConnections: 1
    http:
      http1MaxPendingRequests: 1
      maxRequestsPerConnection: 1
  outlierDetection:
    consecutiveErrors: 2
    interval: 2s
    baseEjectionTime: 3m
    maxEjectionPercent: 100
```

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

Resiliency
Fault injection

Fault injection can be used for testing
- Faults are caused on a percentage of requests
- Faults can cause a request delay or failure

In this example, ratings is being invoked
- All of the requests from bar have a 2 second timeout
- 40% of the requests from bar also have a 5 second delay



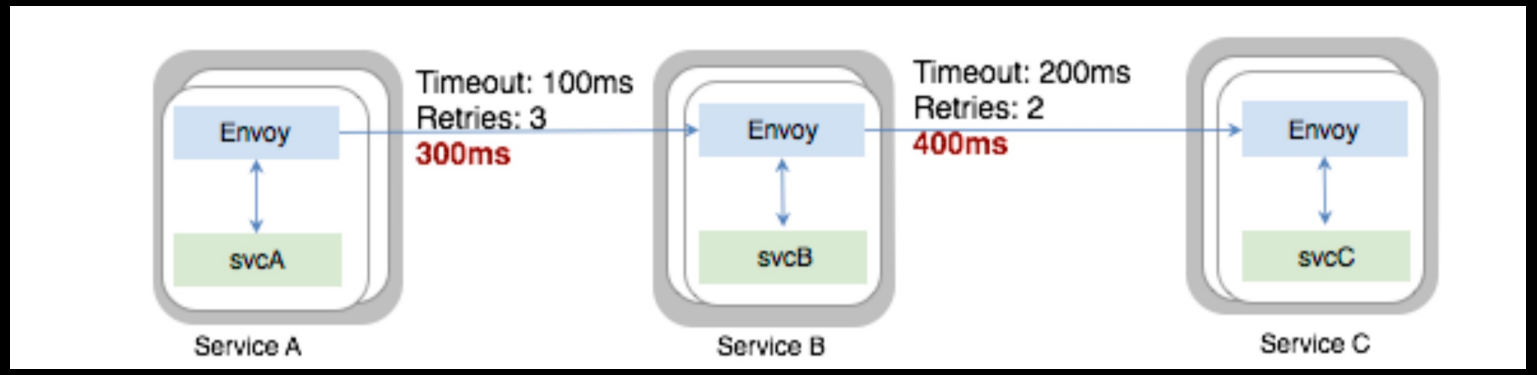

```
kind: VirtualService
hosts:
  - ratings
http:
- match:
  - headers:
      end-user:
        exact: bar
  fault:
    delay:
      percent: 40
      fixedDelay: 5s
  timeout: 2s
  route:
  - destination:
      host: ratings
```

CHALLENGE 4
HOW DO I INJECT FAULT TO MY MICROSERVICES TO PREPARE MYSELF?

Resiliency
Fault injection

Fault injection can be used for testing
- Faults are caused on a percentage of requests
- Faults can cause a request delay or failure

In this example, ratings is being invoked
- 20% of the requests from foo get an HTTP 500 error
- All other requests (not foo or bar) have a 4 second timeout




```
kind: VirtualService
hosts:
  - ratings
http:
- match:
  - headers:
      end-user:
        exact: foo
  fault:
    abort:
      percent: 20
      httpStatus: 500
  route:
  - destination:
      host: ratings
- route:
  - destination:
      host: ratings
  timeout: 4s
```

Timeout is measured after the host is invoked, it is calculated after delay period has passed.

The 40% of requests from bar will time out after 7 seconds (5 sec delay + 2 sec timeout)

CHALLENGE 4
HOW DO I INJECT FAULT TO MY MICROSERVICES TO PREPARE MYSELF?

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?

Rate limiting

Every distinct rate limit configuration represents a counter.
If the number of requests in the last validDuration duration exceed maxAmount, Mixer returns a RESOURCE_EXHAUSTED message to the proxy.

Global rate limit of 500 calls per second.

If “reviews” is called, it’s limited to one call every 5 seconds.

If “reviews” is called from 10.28.11.20, it’s limited to 99 calls per seconds.

```
kind: handler
quotas:
- name: requestcountquota.instance.istio-system
  maxAmount: 500
  validDuration: 1s

  overrides:

  - dimensions:
      destination: reviews
    maxAmount: 1
    validDuration: 5s

  - dimensions:
      destination: productpage
      source: "10.28.11.20"
    maxAmount: 99
    validDuration: 1s
```

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

Telemetry

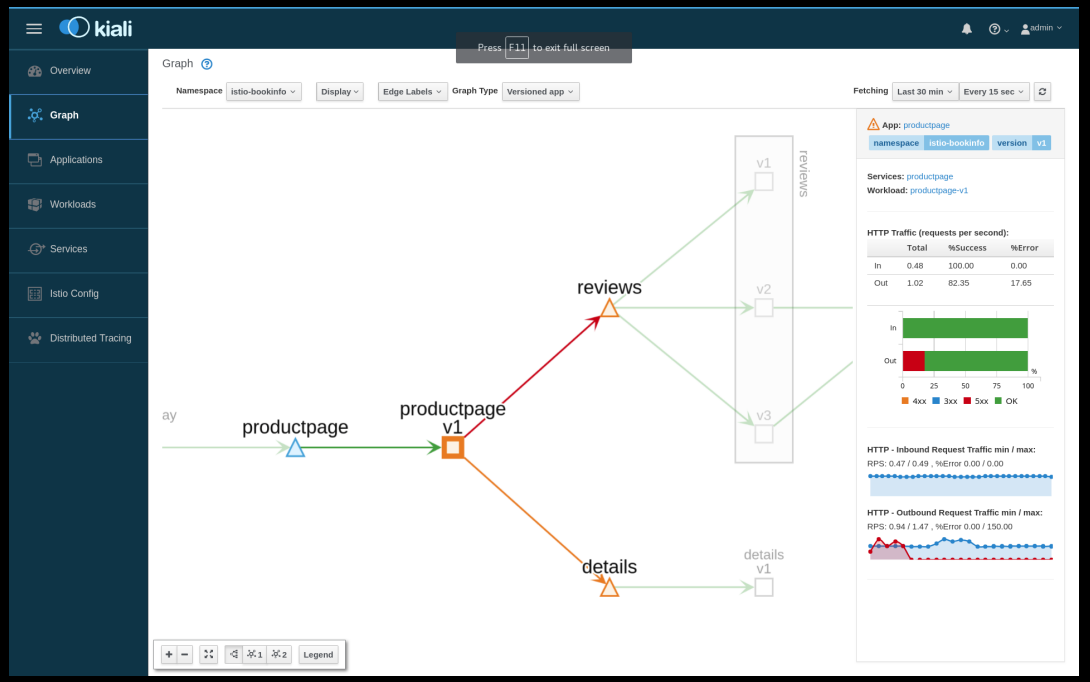
- Monitoring & tracing should not be an afterthought in the infrastructure

- 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

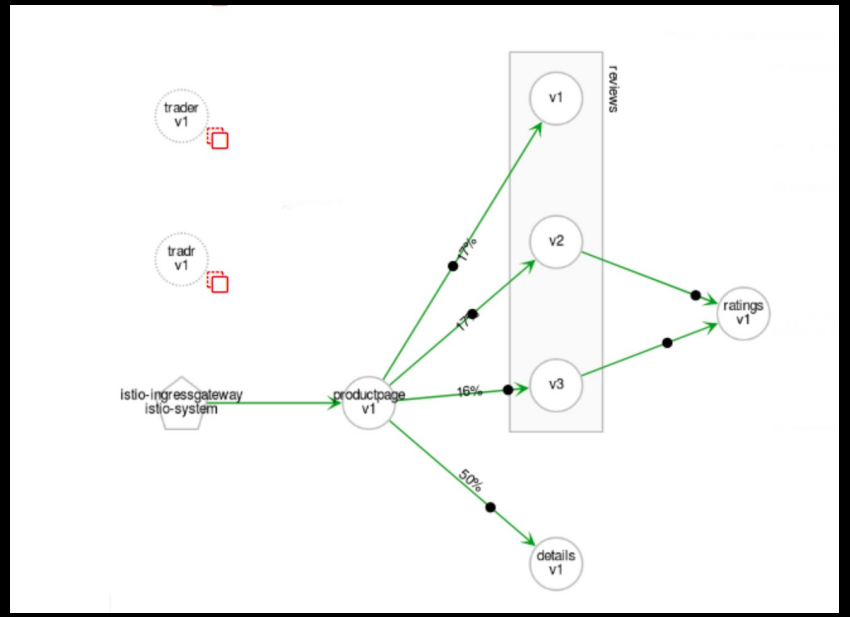
Kiali

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



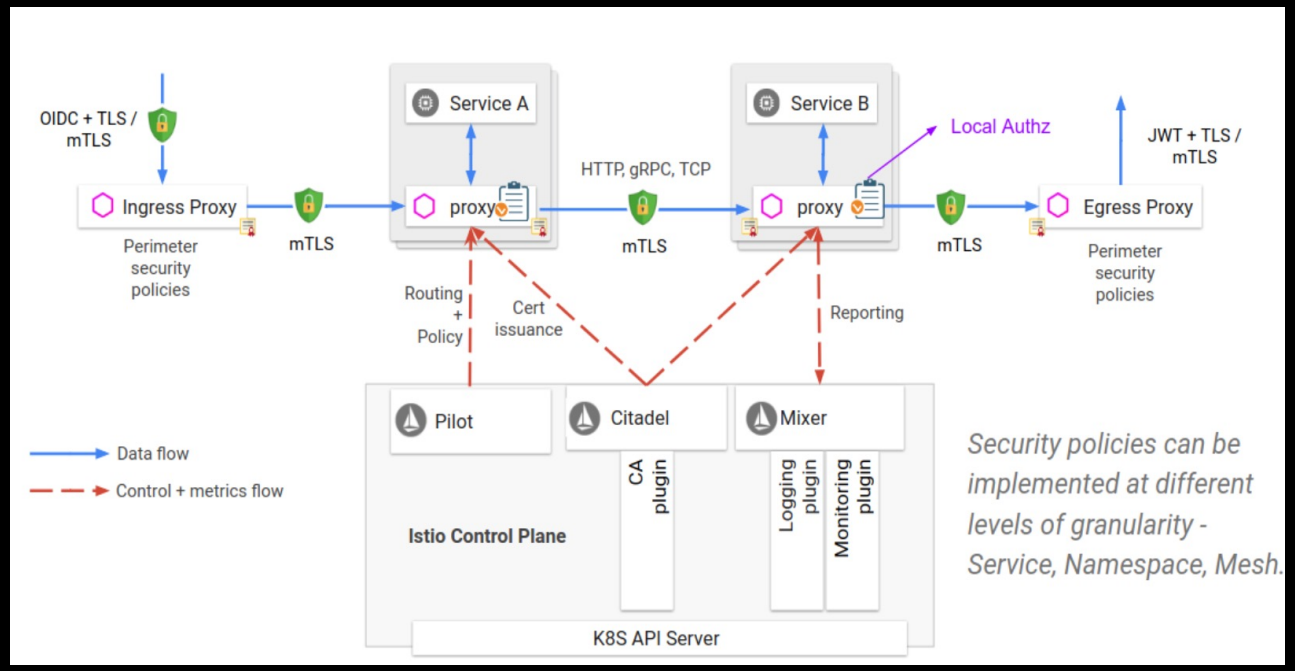
Features

- **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

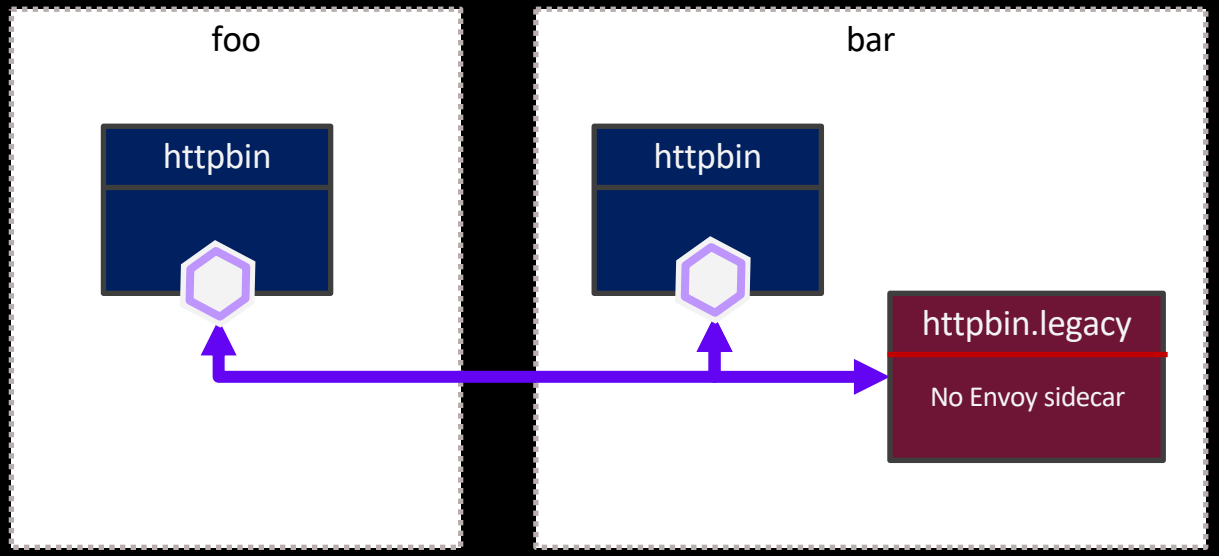
Security



- **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?

Security - Authentication

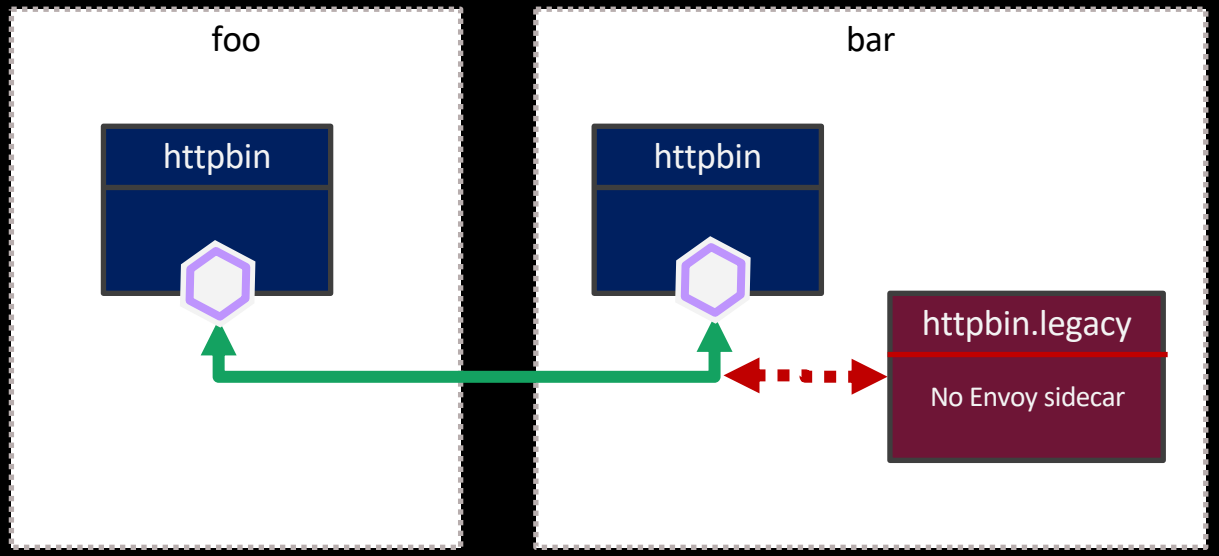


EXAMPLE

CHALLENGE 7
HOW CAN I SECURE MY SERVICES?

Not encrypted
Encrypted (TLS)
No communication

Security - Authentication

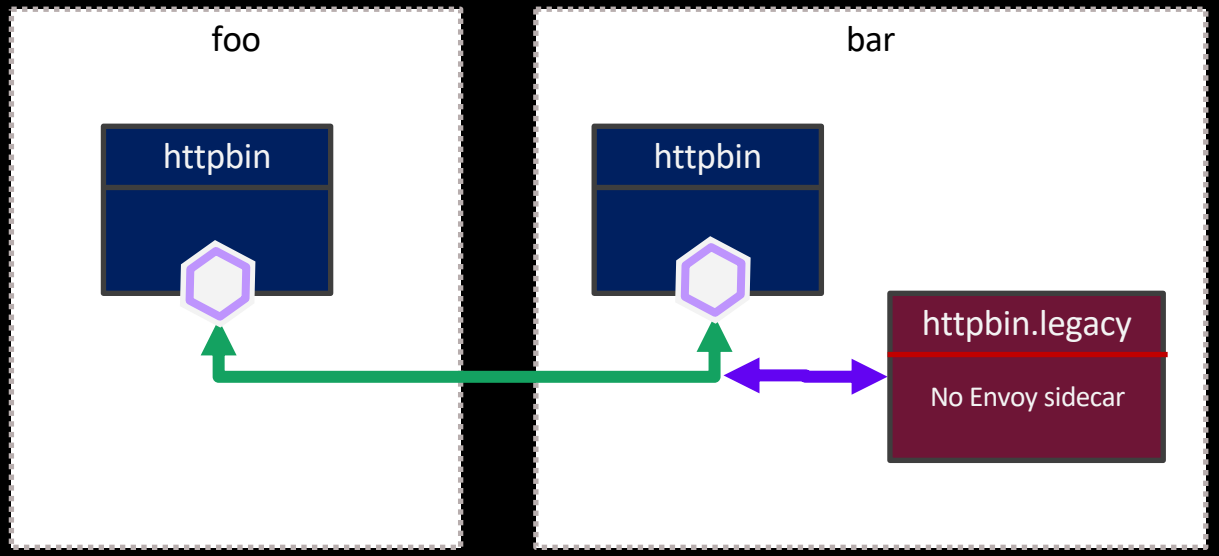



```
kind: DestinationRule
metadata:
  name: "default"
spec:
  host: "*.local"
  trafficPolicy:
    tls:
      mode: ISTIO_MUTUAL
```

CHALLENGE 7
HOW CAN I SECURE MY SERVICES?

Not encrypted
Encrypted (TLS)
No communication

Security - Authentication




```
kind: DestinationRule
metadata:
 name: "httpbin-legacy"
spec:
 host: "httpbin.legacy.svc.cluster.local"
 trafficPolicy:
   tls:
     mode: DISABLE
```

CHALLENGE 7
HOW CAN I SECURE MY SERVICES?

Not encrypted
Encrypted (TLS)
No communication

Security - Authorization

Istio Role Based Access

- **OFF**: Istio authorization is disabled.
- **ON**: enabled for all services in the mesh.
- **ON_WITH_INCLUSION**: enabled for all services specified in the inclusion field.
- **ON_WITH_EXCLUSION**: enabled for all services except the ones in the exclusion field.

```
kind: RbacConfig
metadata:
  name: my-user
spec:
  mode: 'ON_WITH_INCLUSION'
  inclusion:
    services:
    - "webapp.default.svc.cluster.local"
    - "frontend.default.svc.cluster.local"
    - "feedback.default.svc.cluster.local"
```

CHALLENGE 7
HOW CAN I SECURE MY SERVICES?

Security - Authorization

Istio Role Based Access

- **services**: A list of service names.
- **methods**: A list of HTTP method names (GET, POST,…)
- **paths**: HTTP paths (in the form of /packageName.serviceName/methodName)
- **constraints**: additional conditions for your rules.

```
kind: ServiceRole
metadata:
  name: service-viewer
spec:
  rules:
  - services:
    - "webapp.default.svc.cluster.local"
    - "frontend.default.svc.cluster.local"
    methods: ["GET", "HEAD"]
    paths: ["*"]
    constraints:
    - key: request.headers[version]
      values: ["v1", "v2"]
```

CHALLENGE 7
HOW CAN I SECURE MY SERVICES?

Security - Authorization

Access for:

- Service in **Namespace “default” only accessible by authenticated users** and services
- User: "*" assigns the ServiceRole to all (both authenticated and unauthenticated)

```
kind: ServiceRoleBinding
metadata:
  name: binding-products-all-authenticated-users
spec:
  subjects:
  - properties:
      source.principal: "*"
  - properties:
      source.namespace: "default”
  roleRef:
    kind: ServiceRole
    name: ”service-viewer"
```

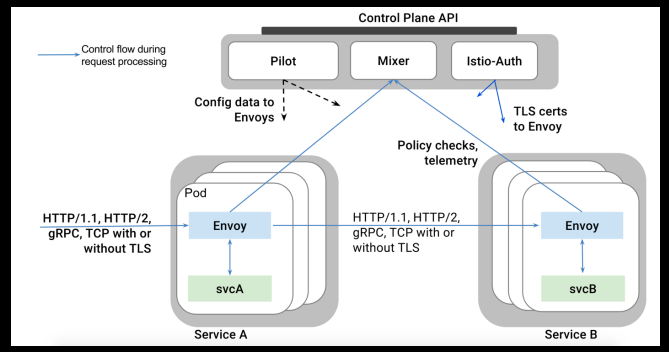
CHALLENGE 7
HOW CAN I SECURE MY SERVICES?

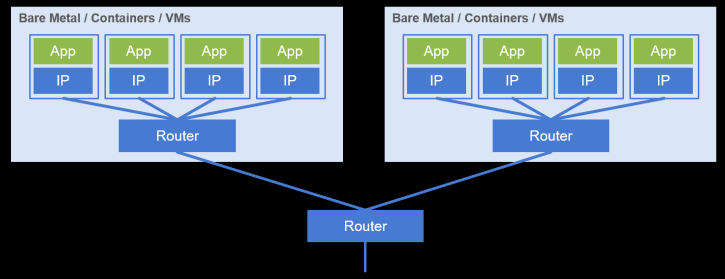
Security
Using Istio in concert with Calico

Istio

PROJECT CALICO

“RPC” — L7	Layer	“Network” — L3-4
Userspace	Implementation	Kernel
Pod	Enforcement Point	Node

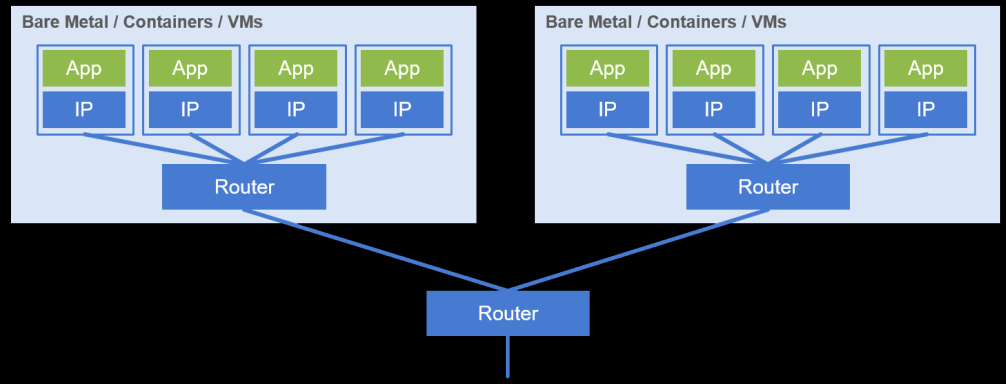




Security
Using Istio in concert with Calico

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.



Security
Using Istio in concert with Calico

“RPC” — L7	Layer	“Network” — L3-4
Userspace	Implementation	Kernel
Pod	Enforcement Point	Node
Ideal for applying policy in support of operational goals, like service routing, retries, circuit-breaking, etc	Strengths	Universal, highly efficient, and isolated from the pods, making it ideal for applying policy in support of security goals

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

QUESTIONS?

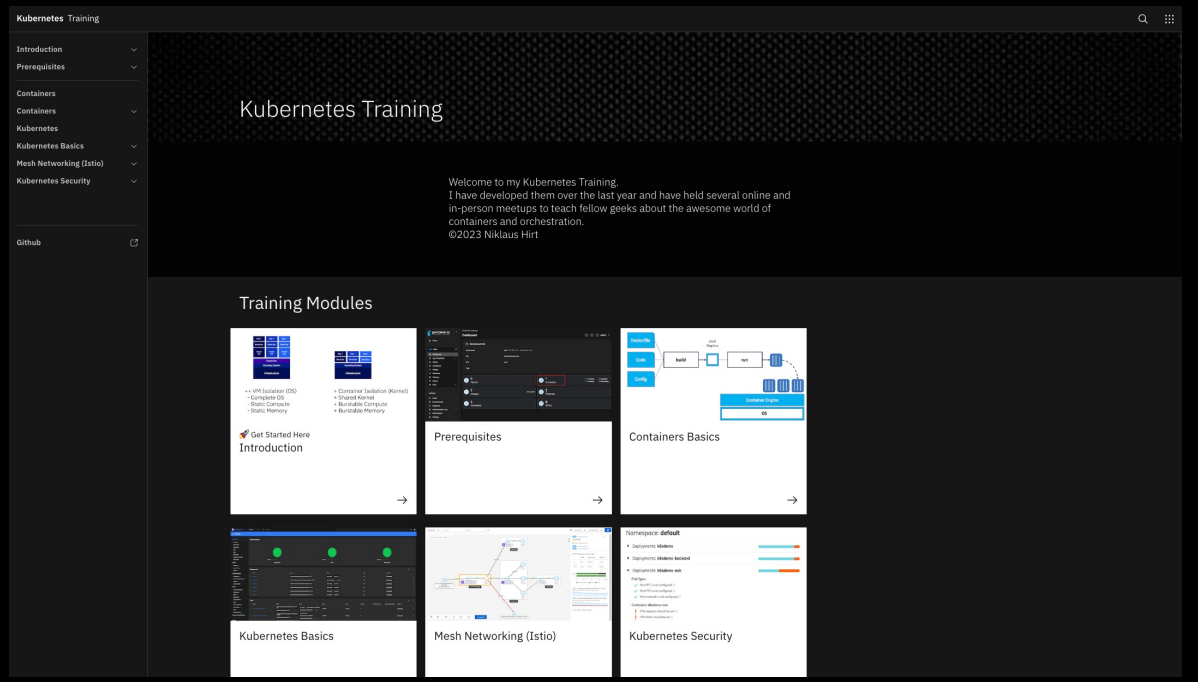
Kubernetes Workshop Series

Labs – Self Study

Labs

Self Study

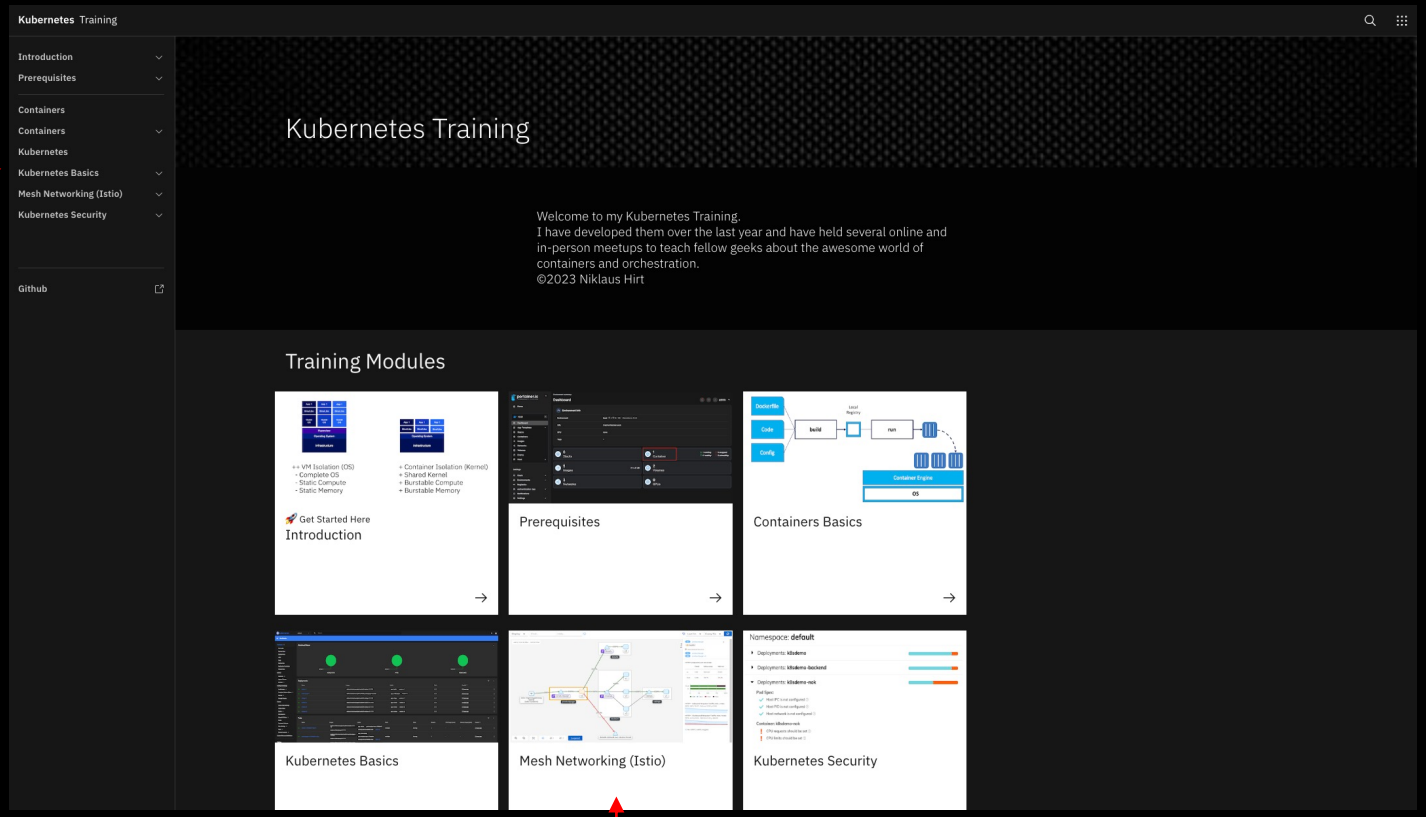
Following the lectures there are ***hands-on*** labs that each participant can complete on their own time.



https://niklaushirt.github.io/k8s_training_web/

Using the online course

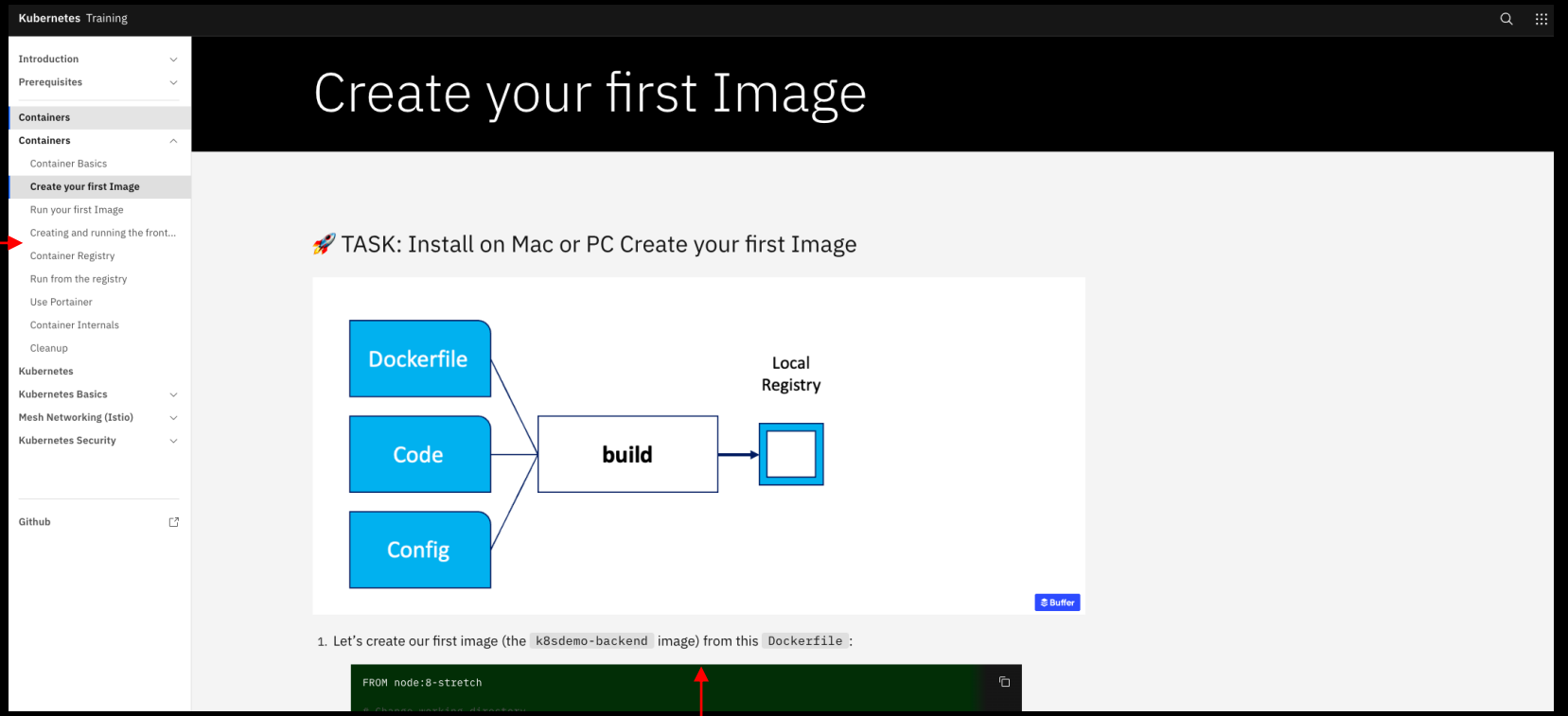
Select the course



Current course catalog

Using the online course

Labs for the Course



Course content - Tasks

Using the online course

Commands to be executed

```
kubectl delete -f ./training/istio/createTraffic.yaml
kubectl delete -f ./training/istio/samples/bookinfo/platform/k
kubectl delete -f  ./training/istio/samples/bookinfo/networkin
kubectl delete -f ./training/istio/samples/bookinfo/networking
kubectl delete -f ./training/istio/samples/bookinfo/networking

istioctl manifest generate --set profile=demo | kubectl delete

kubectl delete ns istio-system
```

Sample output

```
> STEP 1/11: FROM node:8-stretch
> Resolved "node" as an alias (/etc/containers/registries.conf.d/000-shortnames.conf)
> Trying to pull docker.io/library/node:8-stretch...
> Getting image source signatures
```

Sample Code (json, yaml, …)

```
FROM node:8-stretch

# Change working directory
WORKDIR "/app"

# Update packages and install dependency packages for services
RUN apt-get update \
    && apt-get dist-upgrade -y \
    && apt-get clean \
```

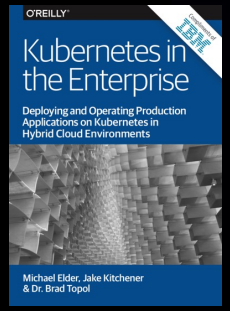
Show more

QUESTIONS?

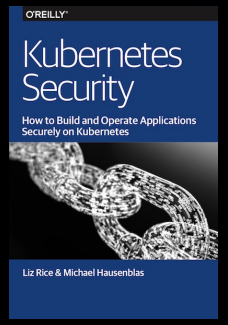
Kubernetes Workshop Series Wrap-up

Kubernetes – Some Reading Tips

The de facto guide to improving your enterprise with the cloud, created by distinguished members of our Solution Engineering team
http://ibm.biz/playbook



Deploying and Operating Production Applications on Kubernetes in Hybrid Cloud Environments
https://ibm.co/2LQketN (excerpt)



https://kubernetes-security.info/

Kubernetes – Some Reading Tips

Deployment of Kubernetes Clusters

- Kubespray - https://github.com/kubernetes-sigs/kubespray
- Minikube - https://github.com/kubernetes/minikube
- Kubeadm - https://github.com/kubernetes/kubeadm
- Kops - https://github.com/kubernetes/kops
- Red Hat OpenShift Local - https://developers.redhat.com/products/openshift-local/overview

Management Tools

- k9s - https://github.com/derailed/k9s
- Lens - https://k8slens.dev/

Security

- Kubesec - https://kubesec.io/
- Falco - https://falco.org/
- kubehunter - https://github.com/aquasecurity/kube-hunter
- conftest - https://github.com/open-policy-agent/conftest

Deployment

- helm - https://helm.sh/
- argocd - https://argo-cd.readthedocs.io/en/stable/

Sources and documentation are available here:

Course Documents
https://github.com/niklaushirt/k8s_training_public

Do you want to go further?

Online Course
https://niklaushirt.github.io/k8s_training_web

Course Videos – Kubernetes: from zero to hero
https://www.youtube.com/channel/UCIS0jmGOQrG2AKKPkTJYj9w/videos

Course Files
https://github.com/niklaushirt/training

THANK YOU!!!!

Niklaus Hirt

 nikh@ch.ibm.com

 @nhirt