

CAS Cloud and Platform Manager

# **Microservices, Containers und Mesh Networking**

**Niklaus Hirt**

Cloud and DevOps Architect

[nikh@ch.ibm.com](mailto:nikh@ch.ibm.com)

Rotkreuz, 18.12.2020

# Who am I?

## Niklaus Hirt

Passionate about tech for over 35 years

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



✉ nikh@ch.ibm.com

🐦 @nhirt

# Agenda – Microservices and Docker - Basic Concepts

Module 0: Prepare the Labs

Module 1: Microservices

Module 2: Containers with Docker

Module 3: Let's get real

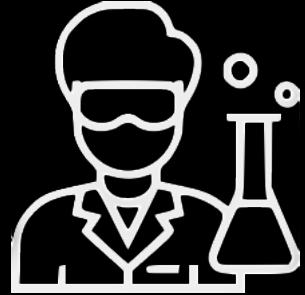
Module 4: Docker Hands-On

Lunch

Module 5: Kubernetes

Module 6: Mesh Networking

Module 7: Kubernetes Hands-On



Sources and documentation will be available here:

Online Course

[https://niklaushirt.github.io/k8s\\_training\\_web/](https://niklaushirt.github.io/k8s_training_web/)

Course Documents

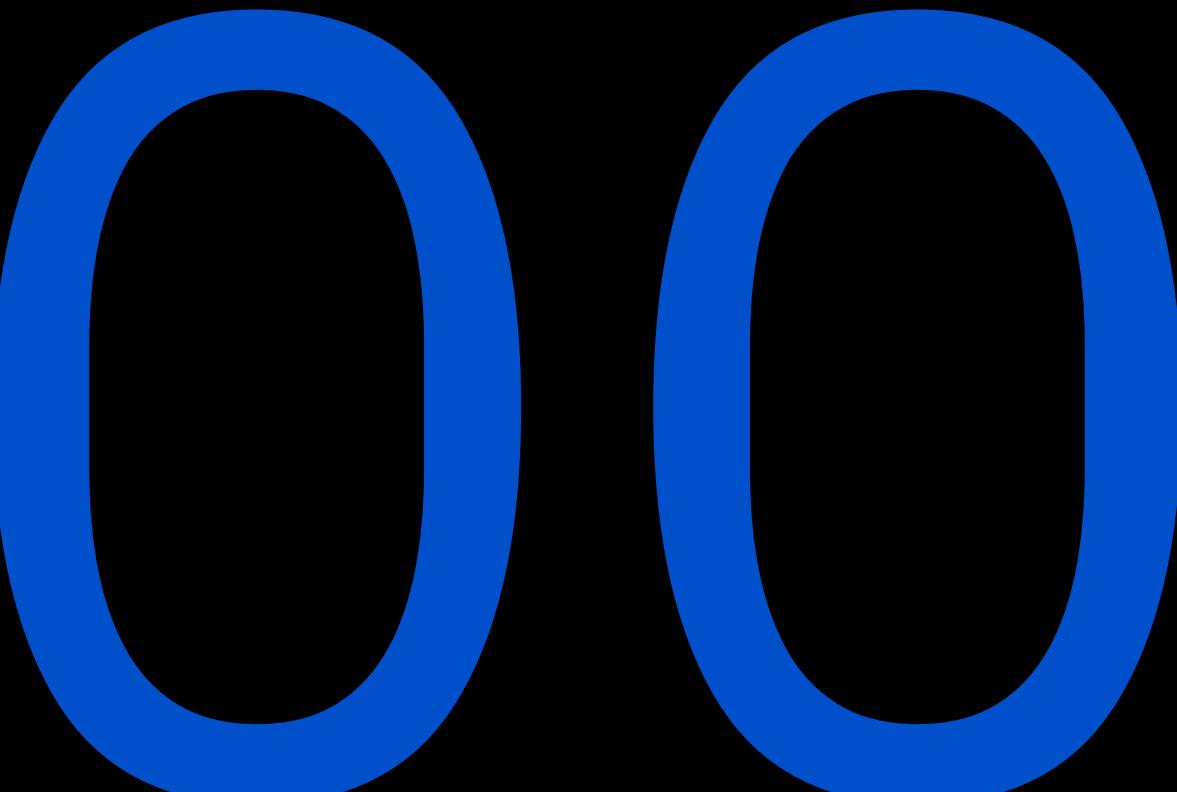
[https://github.com/niklaushirt/k8s\\_training\\_public](https://github.com/niklaushirt/k8s_training_public)

Course Files

<https://github.com/niklaushirt/training>



## ◦◦ Kubernetes Workshop Series **Prepare the Labs**



# Session Objectives

Attendees will run their own ***Personal Training Environment (PTE)*** in the VM.

Following the lectures there will be ***hands-on*** labs that each participant can complete in the PTE.



Kubernetes Training

Search

Introduction  
JTC01 Docker  
JTC02 Kubernetes Labs  
Lab 1: Get to know Kubernetes  
Lab 2: Deploy your first Pod  
**Lab 3: Deploy your first application**  
Lab 4: Scale and Update Deployments  
Lab 5: Stateful Deployments  
Lab 6: Cleanup  
JTC10 Istio  
JTC14 Kubernetes Ansible Operators  
JTC16 Kubernetes Security Labs  
JTC17 Kubernetes Advanced Security

## Lab 3: Deploy your first application

Learn how to deploy an application to a Kubernetes cluster.  
Once your client is configured, you are ready to deploy your first application, **k8sdemo**.

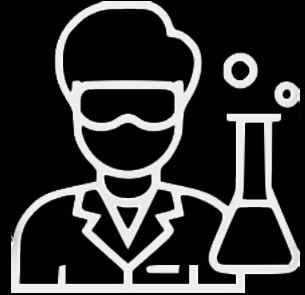
### The frontend example application

In this part of the lab we will deploy an application called **k8sdemo** that has already been built and uploaded to DockerHub under the name **niklaushirt/k8sdemo**.  
We will use the following yaml:

```
kind: Deployment
metadata:
  name: k8sdemo
  namespace: default
spec:
  replicas: 1
  template:
    metadata:
```

Copy

The front end example application  
Deploy the frontend application  
Deploy the application backend



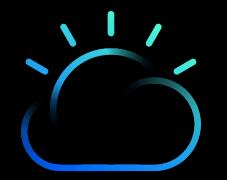
JTC90 Lab Setup

Task 1: Download Training VM

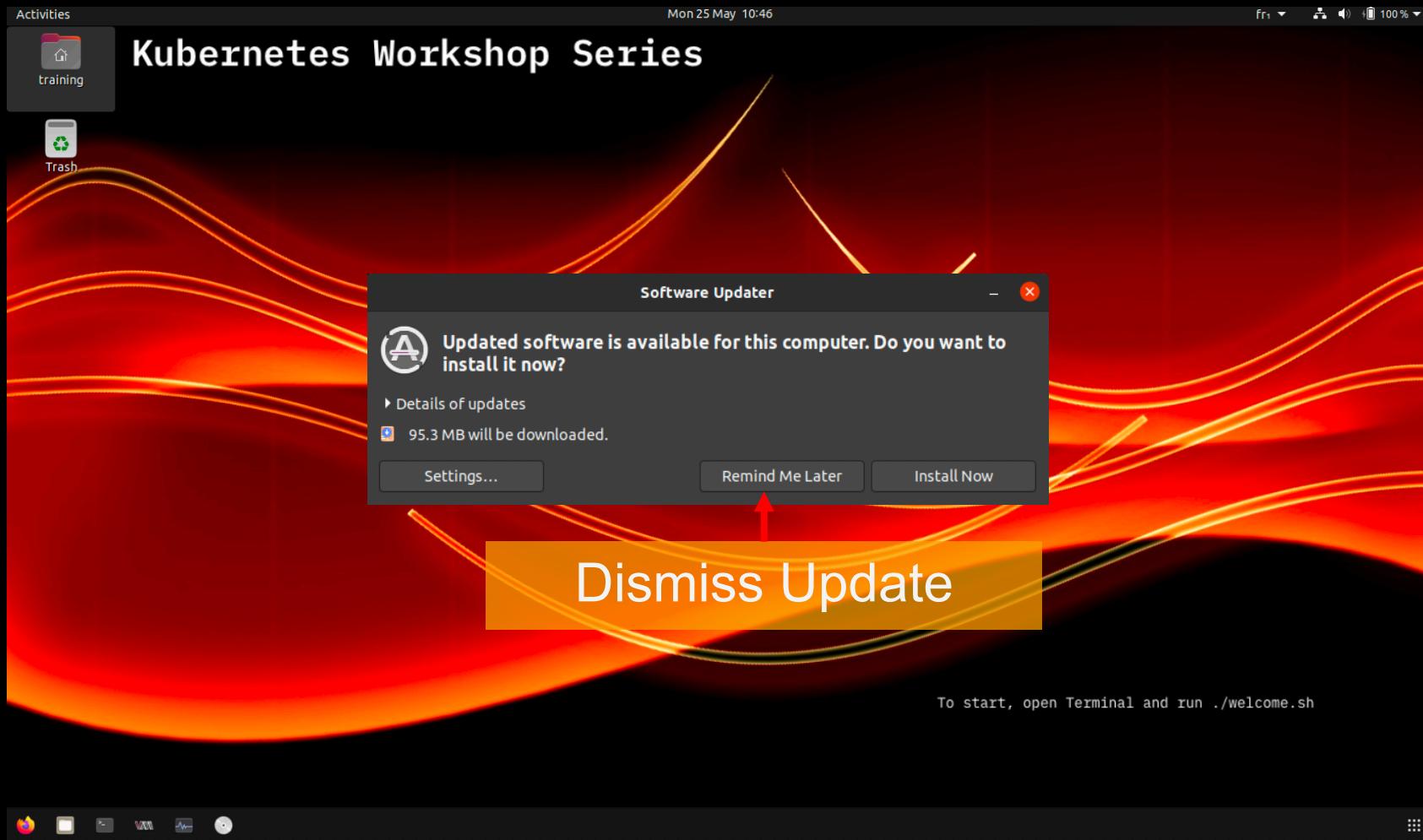
Task 2: Setup VMWare / VirtualBox

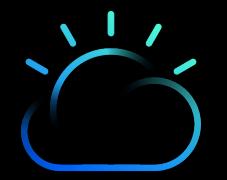
Task 3: Start Training VM

Task 4: Login / Check



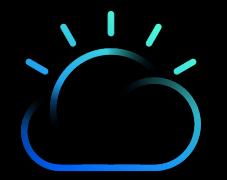
# Using your Personal Training Environment





# Using your Personal Training Environment





# Using your Personal Training Environment



Start Terminal

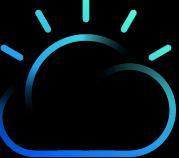


# Using your Personal Training Environment

A screenshot of a terminal window titled "training@ubuntu: ~". The command "training@ubuntu:~\$ ./welcome.sh" is visible, with a red box highlighting the command line. A red arrow points from the text "Run ./welcome.sh" below the terminal to the highlighted command line.

```
training@ubuntu:~$ ./welcome.sh
```

- Start Docker
- Start minikube
- Prepares networking
- StartPTE
- Start Kubernetes Dashboard

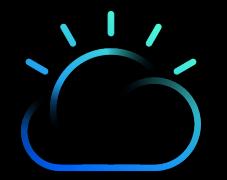


# Using your Personal Training Environment



The screenshot shows a Firefox browser window with the title "Welcome to the Kubernetes Course | Kubernetes Training - Mozilla Firefox". The address bar displays the URL [https://niklaushirt.github.io/k8s\\_training\\_web/](https://niklaushirt.github.io/k8s_training_web/). The main content area features a dark blue header with the text "Kubernetes Training". On the left, there's a sidebar with a "Search" input field and a list of course links: "Introduction", "JTC01 Docker", "JTC02 Kubernetes Labs", "JTC10 Istio", "JTC14 Kubernetes Ansible Operators", "JTC16 Kubernetes Security Labs", and "JTC17 Kubernetes Advanced Security". The central content area has a large blue header "Welcome to the Kubernetes Course" and a sub-section "Welcome to my courses around Kubernetes." Below these, there's a paragraph about the developer's history with Kubernetes training and links to various resources like YouTube videos and GitHub repositories. A blue button at the bottom of the central area says "Introduction >".

First your Online Course will open automatically



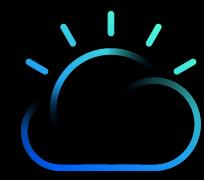
# Using your Personal Training Environment

The screenshot shows the Kubernetes Dashboard running locally at `127.0.0.1:39833/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/overview?namespace=default`. The left sidebar has a 'Cluster' section with 'Cluster Roles', 'Namespaces', 'Nodes', 'Persistent Volumes', and 'Storage Classes'. Under 'Namespaces', 'default' is selected. The 'Overview' tab is active. The main content area has two sections: 'Discovery and Load Balancing' and 'Config and Storage'. In 'Discovery and Load Balancing', the 'Services' table shows one entry: 'kubernetes' (Name), 'default' (Namespace), 'component: apiserver, provider: kubernetes' (Labels), '10.96.0.1' (Cluster IP), 'kubernetes:443 TCP, kubernetes:0 TCP' (Internal and External Endpoints), and '5 months ago' (Created). In 'Config and Storage', the 'Secrets' table shows one entry: 'default-token-8bvs4' (Name), 'default' (Namespace), '-' (Labels), 'kubernetes.io/service-account-token' (Type), and '5 months ago' (Created).

Name	Namespace	Labels	Cluster IP	Internal Endpoints	External Endpoints	Created
kubernetes	default	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	5 months ago

Name	Namespace	Labels	Type	Created
default-token-8bvs4	default	-	kubernetes.io/service-account-token	5 months ago

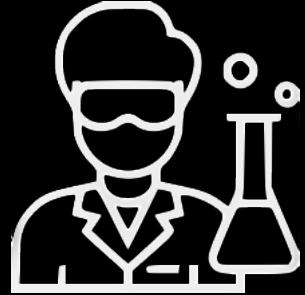
When completed, the Kubernetes Dashboard will open automatically



# Using your Personal Training Environment

## Troubleshooting

- If the startup script doesn't work you can run `./resetEnvironment.sh`  
(this can take up to 30 minutes as it has to redownload all Docker images)
- If you lose your Training Webpage go to [https://niklaushirt.github.io/k8s\\_training\\_web/](https://niklaushirt.github.io/k8s_training_web/)



JTC90 Lab Setup

**EVERYBODY**

Task 1: Download Training VM

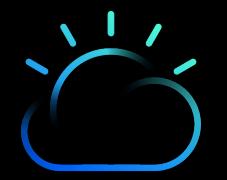
**OK**

Task 2: Setup VMWare / VirtualBox

**?**

Task 3: Start Training VM

Task 4: Login / Check



# Using your Personal Training Environment

Select course to begin

Welcome to the Kubernetes Course | Kubernetes Training - Mozilla Firefox

Wed 11 Nov 08:32

Welcome to the Kubernetes Course

Kubernetes Training

Search

Introduction  
JTC01 Docker  
JTC02 Kubernetes Labs  
JTC10 Istio  
JTC14 Kubernetes Ansible Operators  
JTC16 Kubernetes Security Labs  
JTC17 Kubernetes Advanced Security

Welcome to my courses around Kubernetes.

I have developed them over the last year and have held several online and in-person meetups to teach fellow geeks about the awesome world of containers and orchestration.

You can find the sources here

Videos <https://www.youtube.com/channel/UCIS0jmGOQrG2AKPKTJYj9w/>

Training Documents [https://github.com/niklaushirt/k8s\\_training\\_public](https://github.com/niklaushirt/k8s_training_public)

Training files <https://github.com/niklaushirt/training.git>

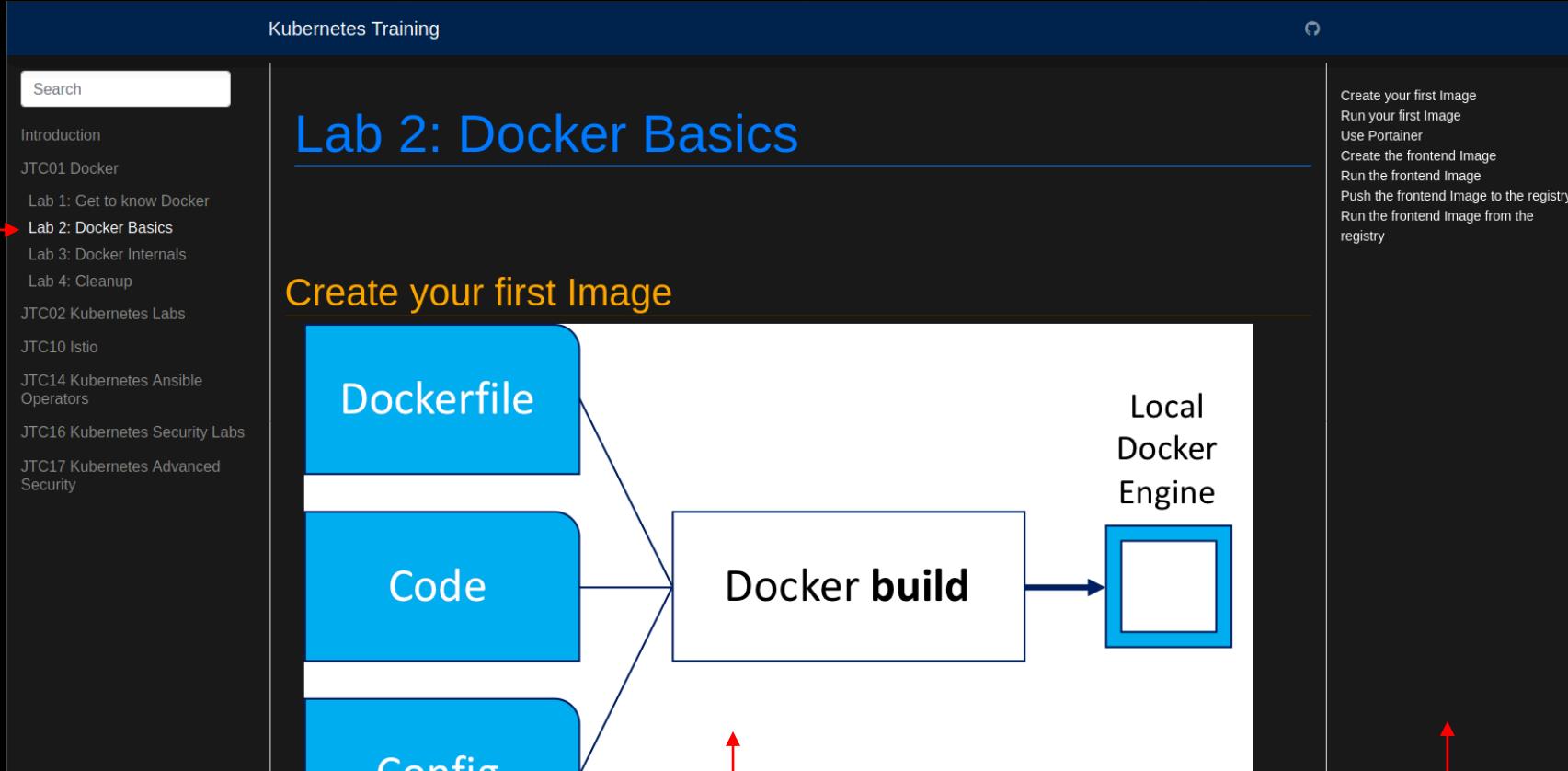
©2020 Niklaus Hirt

Introduction >

Current course catalog

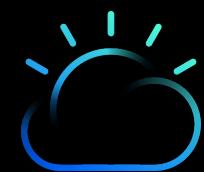
# Using your Personal Training Environment

Labs for the Course



Course content - Tasks

Lab Steps



# Using your Personal Training Environment

Kubernetes Training

Search

Introduction  
JTC01 Docker  
Lab 1: Get to know Docker  
**Lab 2: Docker Basics**  
Lab 3: Docker Internals  
Lab 4: Cleanup  
JTC02 Kubernetes Labs  
JTC10 Istio  
JTC14 Kubernetes Ansible Operators  
JTC16 Kubernetes Security Labs  
JTC17 Kubernetes Advanced Security

Run the frontend Image from the registry

1. Now let's start the Web Frontend container with the image from the registry

```
docker run -d --rm --name k8sdemo -p 3000:3000 --env BACKEND_URL=http://172.17.0.1:3001/api localhost:5000/k8sdemo:lab
```

This command runs the frontend server:

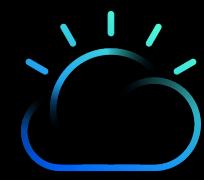
- `-rm` makes sure that the container is deleted once it's stopped
- `--name` gives the container a fixed name
- `--env` defines the environment variable that points to the `k8sdemo-backend` server API
- `-p` exposes the container port 3000 to the outside port 3000
- `-d` runs the container in the background (as a daemon)
- `localhost:5000/k8sdemo:lab` is the image we have pushed to the registry before

3. Go back to your browser and refresh the `k8sdemo` web application to make sure that the container has been started.

Congratulations!!! This concludes the Lab

Lab 3: Docker Internals >

Click to go to the next Lab



# Using your Personal Training Environment

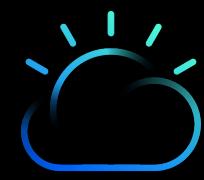
Commands  
to be  
executed

Sample  
output

```
cd ~/training/demo-app/k8sdemo_backend
docker build -t k8sdemo-backend:lab .

> Sending build context to Docker daemon 6.975MB
> Step 1/11 : FROM node:8-stretch
>  --> 7a9afc16a57f
> Step 2/11 : WORKDIR "/app"
>  --> Using cache
>  --> a2515f8a3ec5
...
> Step 11/11 : CMD ["npm", "start"]
>  --> Using cache
>  --> b9b0f3fea9f7
> Successfully built b9b0f3fea9f7
> Successfully tagged k8sdemo-backend:lab
```

Copy



# Using your Personal Training Environment

## 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 \
&& echo 'Finished installing dependencies'

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

COPY . /app

ENV NODE_ENV production
ENV BACKEND_MESSAGE HelloWorld

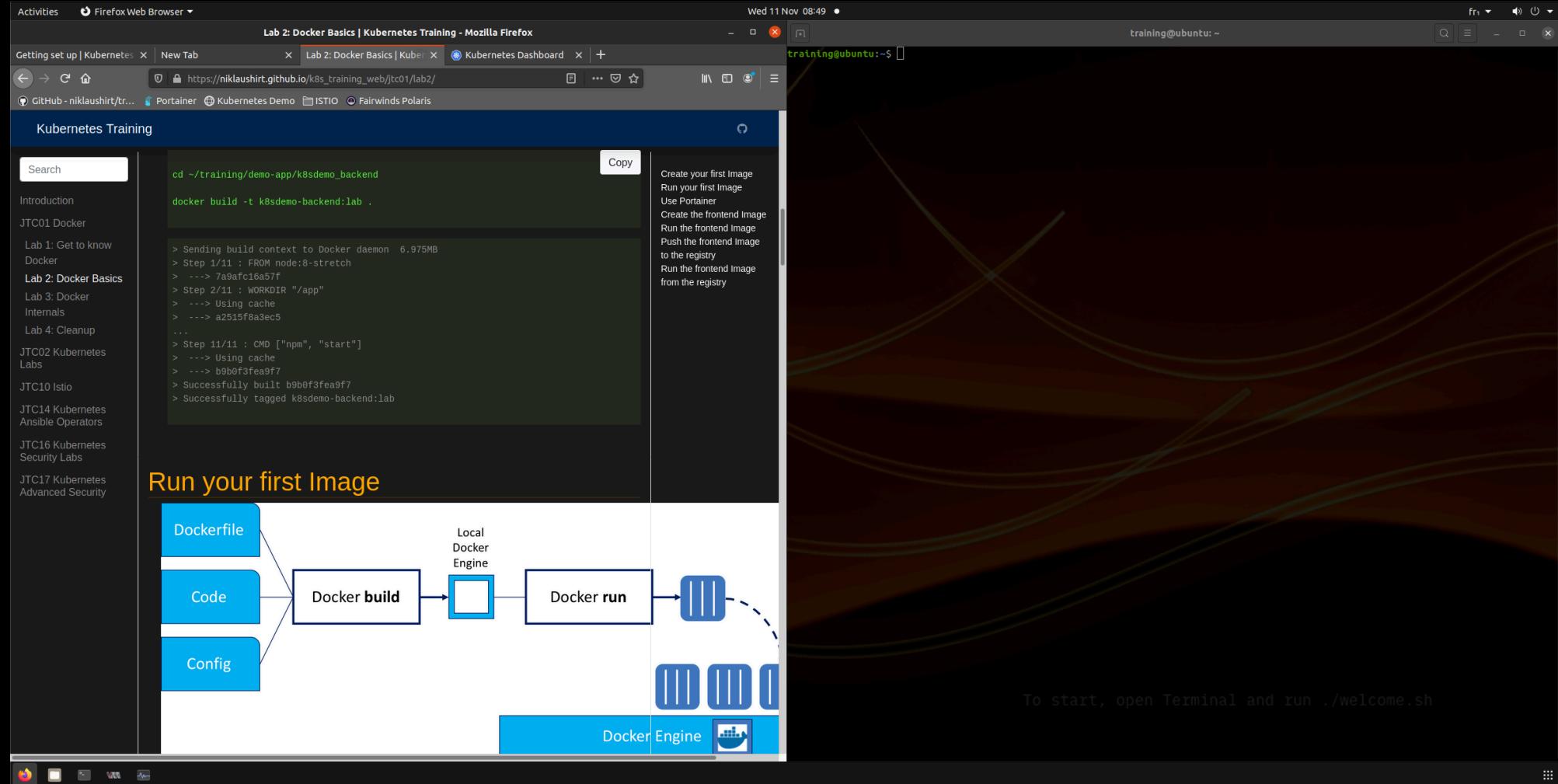
ENV PORT 3000

EXPOSE 3000

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

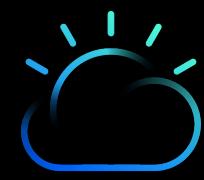
Copy

# Using your Personal Training Environment



Tip 1: Work side by side

→ Just drag Firefox to the left screen edge and the Terminal to the right



# Using your Personal Training Environment

Tip 2: Copy / Paste in the VM

CTRL-C / CTRL-P (Mac users be aware)

Tip 3: Copy / Paste in the Terminal (in the VM)

SHIFT-CTRL-C / SHIFT-CTRL-P

CTRL-C interrupts the currently running command

# QUESTIONS?



# Kubernetes Workshop Series

## **Microservices**

01





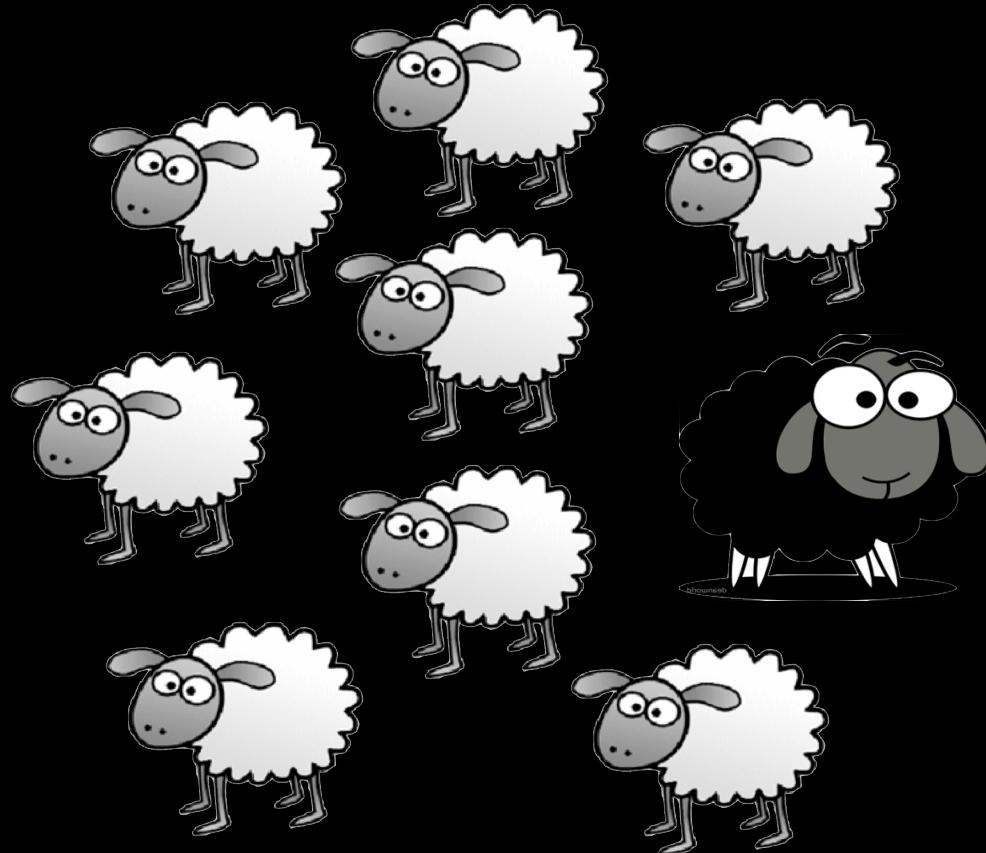
# disruption

dɪs'ruptʃn/

*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/

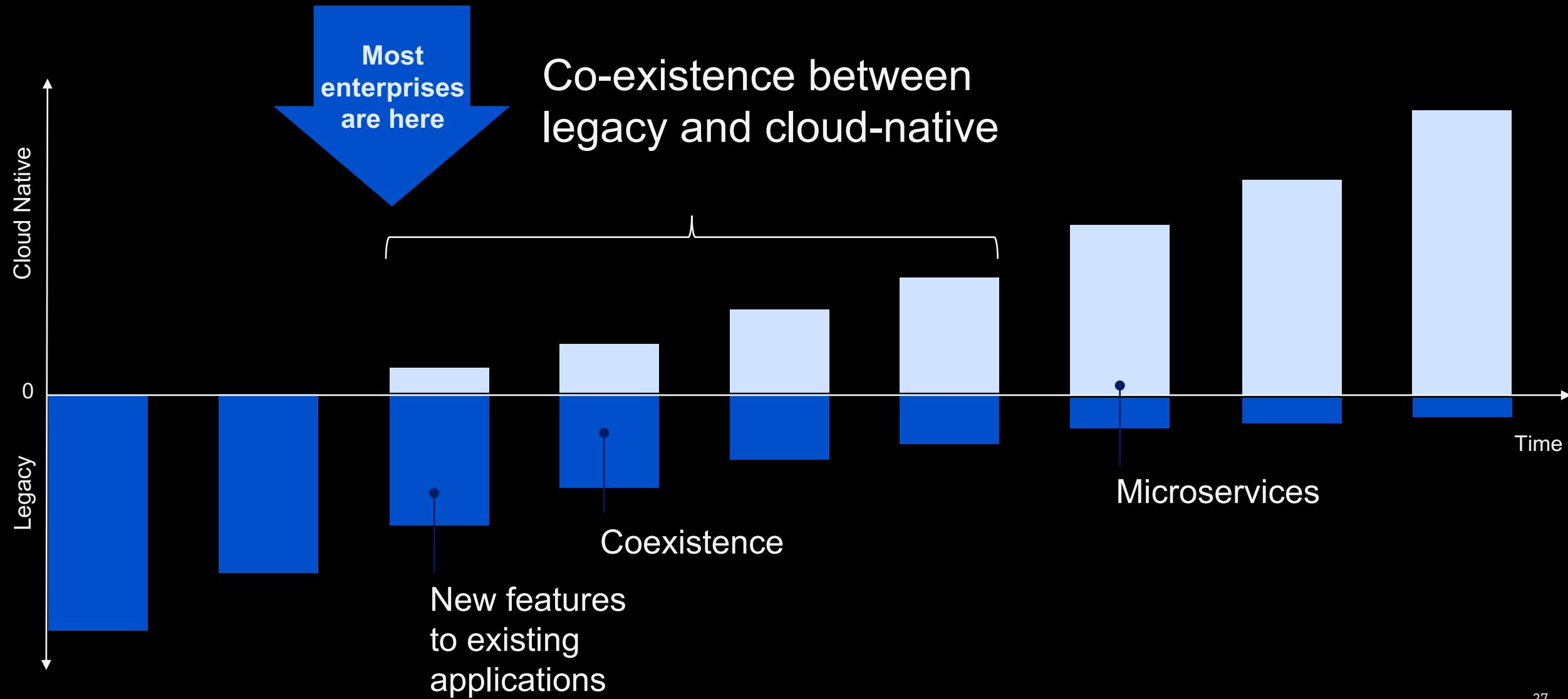
*noun*

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

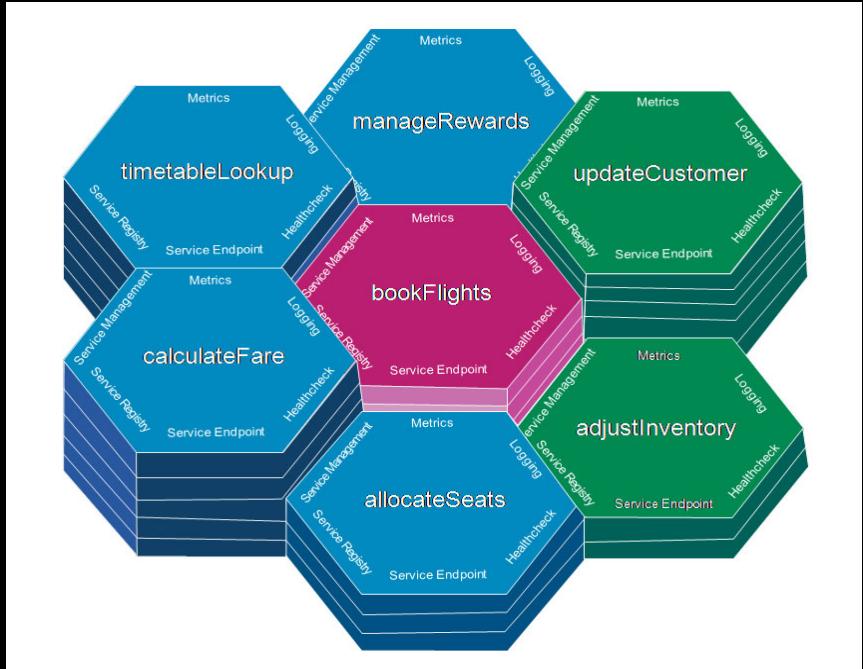
*Cloud native and legacy apps will co-exist for the next 10+ years*



# How do you achieve digital transformation?

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

# Microservices

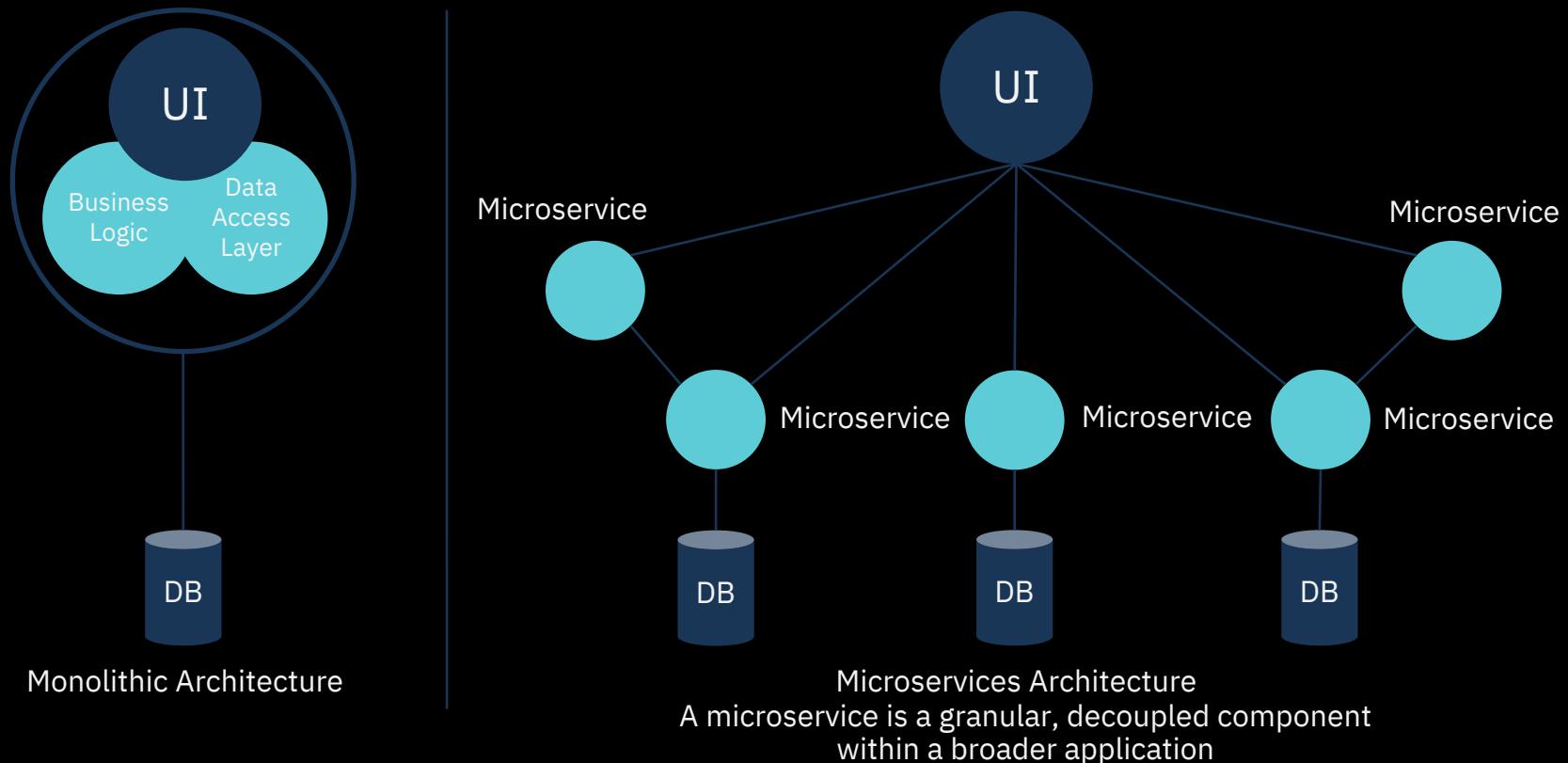


Decomposing an application into **single function modules** which are **independently deployed and operated**

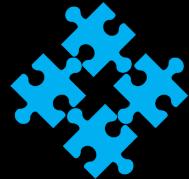
**Accelerate delivery** by minimizing communication and coordination between people

# Microservices architecture

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



# Microservices – key tenets



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

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



Services are **optimized for a single function**

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



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

- Avoid tight coupling introduced by communication through a database



**Per-service continuous delivery** (CI/CD)

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



**Per-service high availability** and clustering decisions

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

# Obvious rules for developing cloud applications

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

Read the article : [http://www.ibm.com/developerworks/websphere/techjournal/1404\\_brown/1404\\_brown.html](http://www.ibm.com/developerworks/websphere/techjournal/1404_brown/1404_brown.html)

That said....

# Microservices

are

hard

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

# QUESTIONS?



# Kubernetes Workshop Series

**Docker**

02



# Everybody Loves Containers



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

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

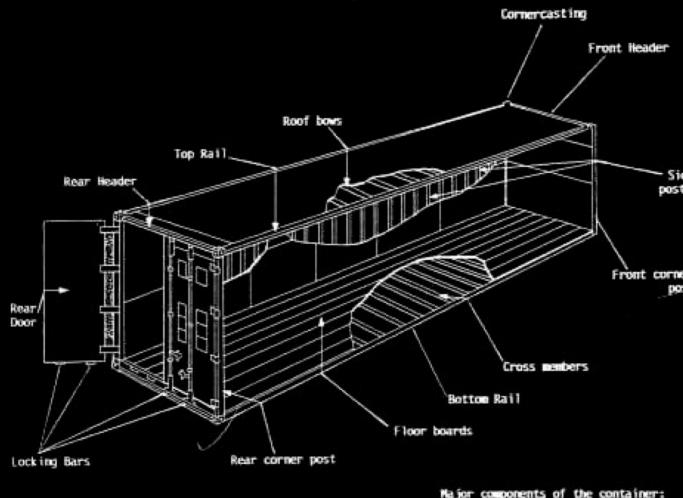
# Microservices implementation with Containers

---

## Why it works – separation of concerns

### Development

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

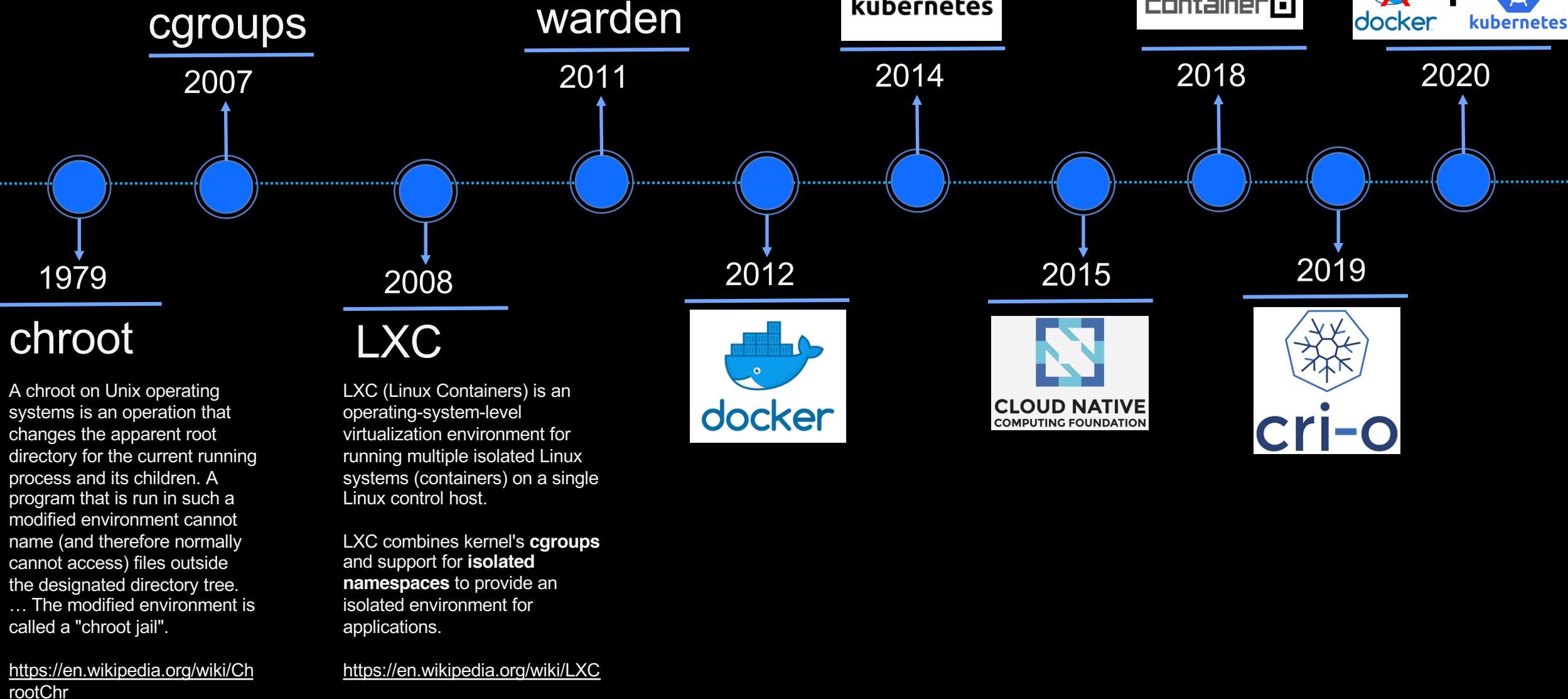


### Operations

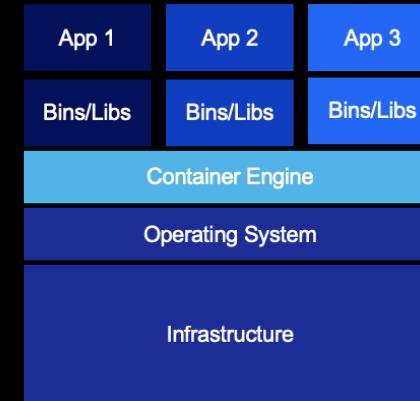
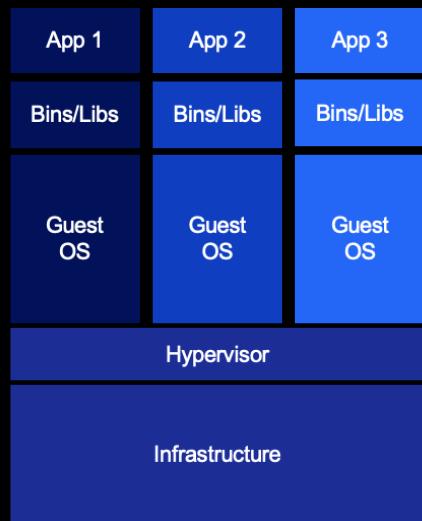
- Worries about what's “**outside**” the container
  - Logging
  - Remote Access
  - Monitoring
  - Network Config
- All containers start, stop, copy, attach, migrate, etc... the same way

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

# Container History



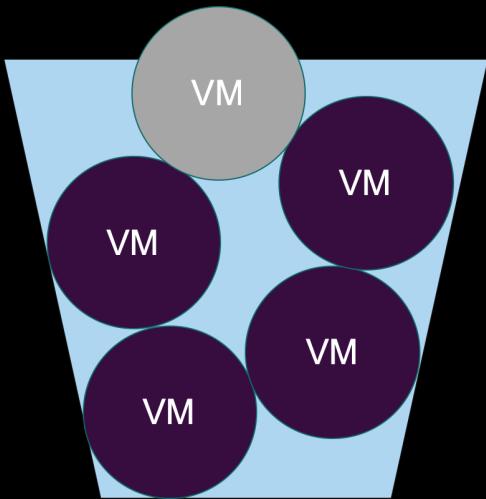
# VMs vs. Containers



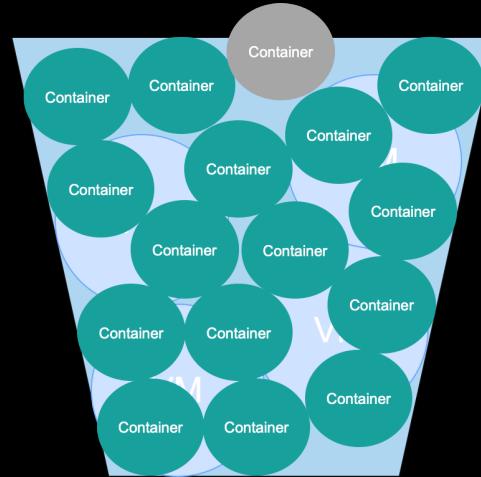
- ++ VM Isolation (OS)**
  - Complete OS
  - Static Compute
  - Static Memory

- + 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 are **portable**



Containers are **easy to manage**



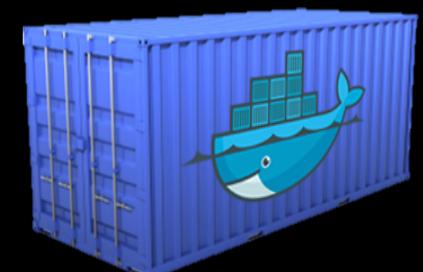
Containers provide “**just enough**” isolation



Containers use hardware **more efficiently**



Containers are **immutable**



# Docker Components

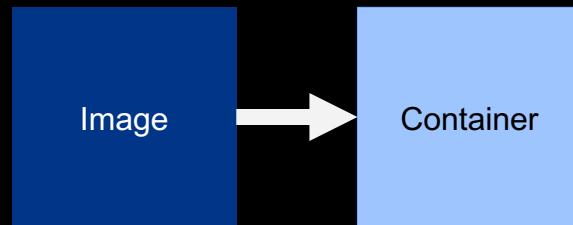
## Container

Smallest compute unit



# Docker Components

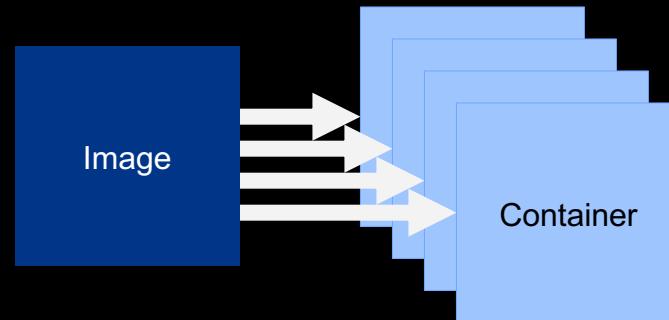
**Containers**  
are created from  
**Images**



one process per container

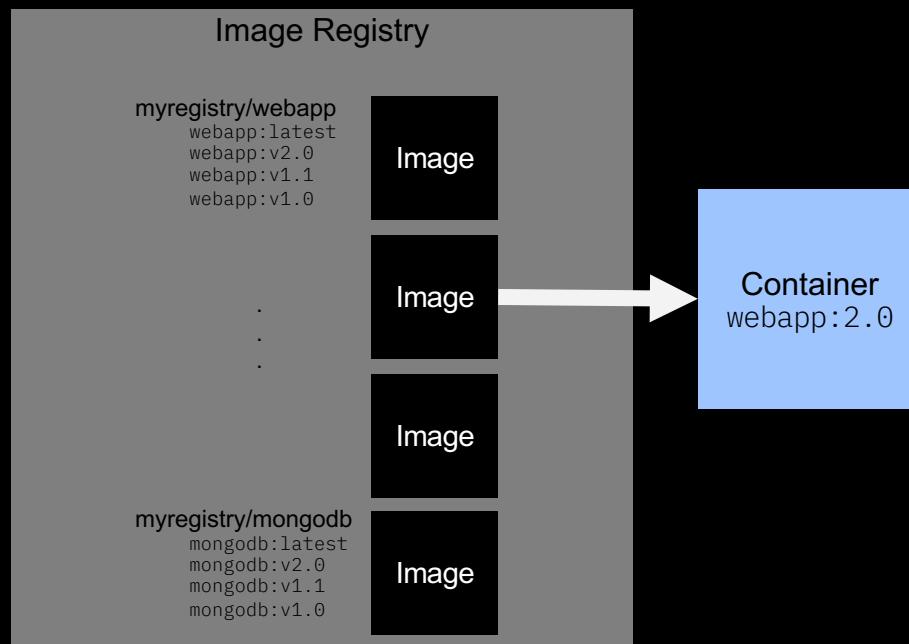
# Docker Components

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



# Docker Components

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



# Docker Components



## Image

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



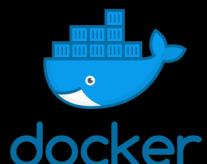
## Container

The standard unit in which the application service resides or transported



## Docker Hub/Registry

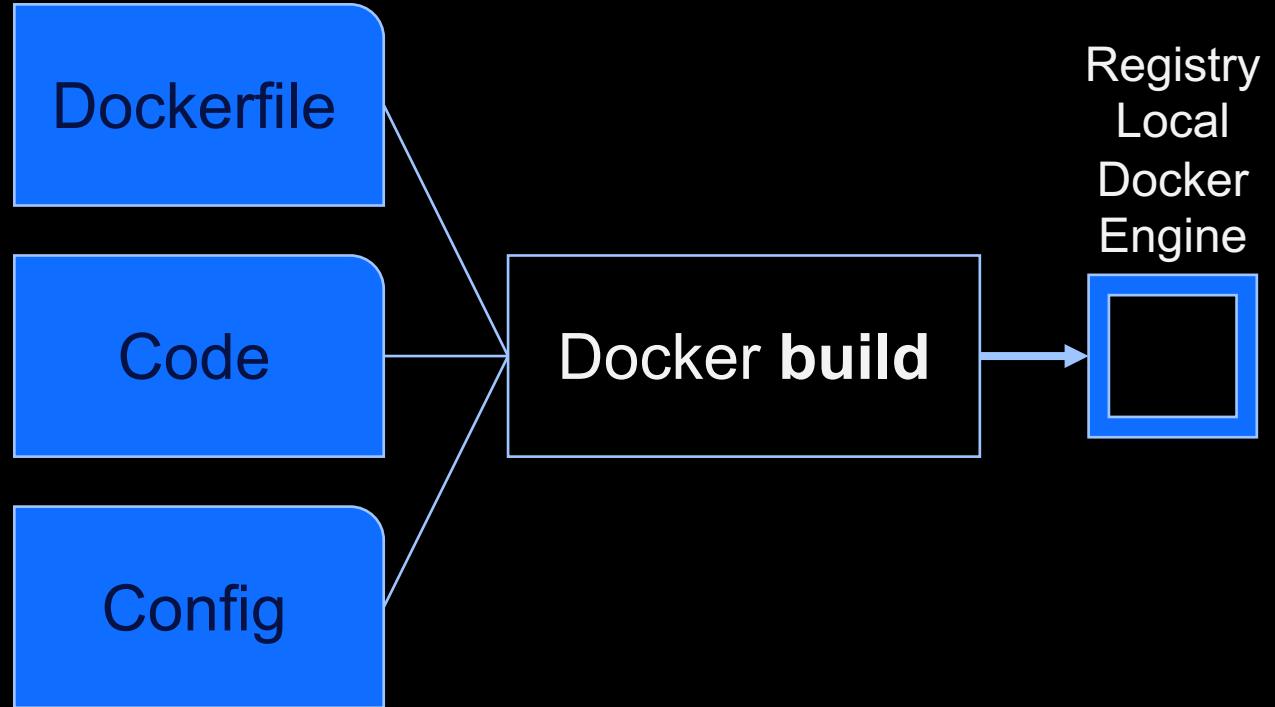
Available in SaaS or Enterprise to deploy anywhere you choose  
Stores, distributes, and shares container images



## Docker Engine

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

# Docker Basics – Build



# Docker Basics – Build - Dockerfile

## Dockerfile

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

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

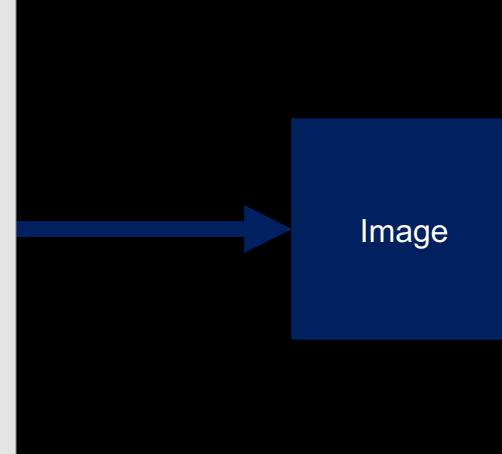
# Maintainer
MAINTAINER "youremailaddress"

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

# Set variables
ENV myName John Doe

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

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



# Docker Basics – Build - Dockerfile

- A text file that builds an image using Docker directives
- FROM
- RUN
- COPY
- ENTRYPOINT
- LABEL
- ENV
- ARG
- USER
- EXPOSE

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

# Maintainer
MAINTAINER "youremailaddress"

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

# Set variables
ENV myName John Doe

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

# Expose the port of the web application
EXPOSE 80

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

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

Base Image from:

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

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

Set environment variables  
Can be overridden 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

# Docker Basics – Layered File System

## Docker Layers Inheritance

- Build on the work of those who came before you

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

# Maintainer
MAINTAINER "youremailaddress"

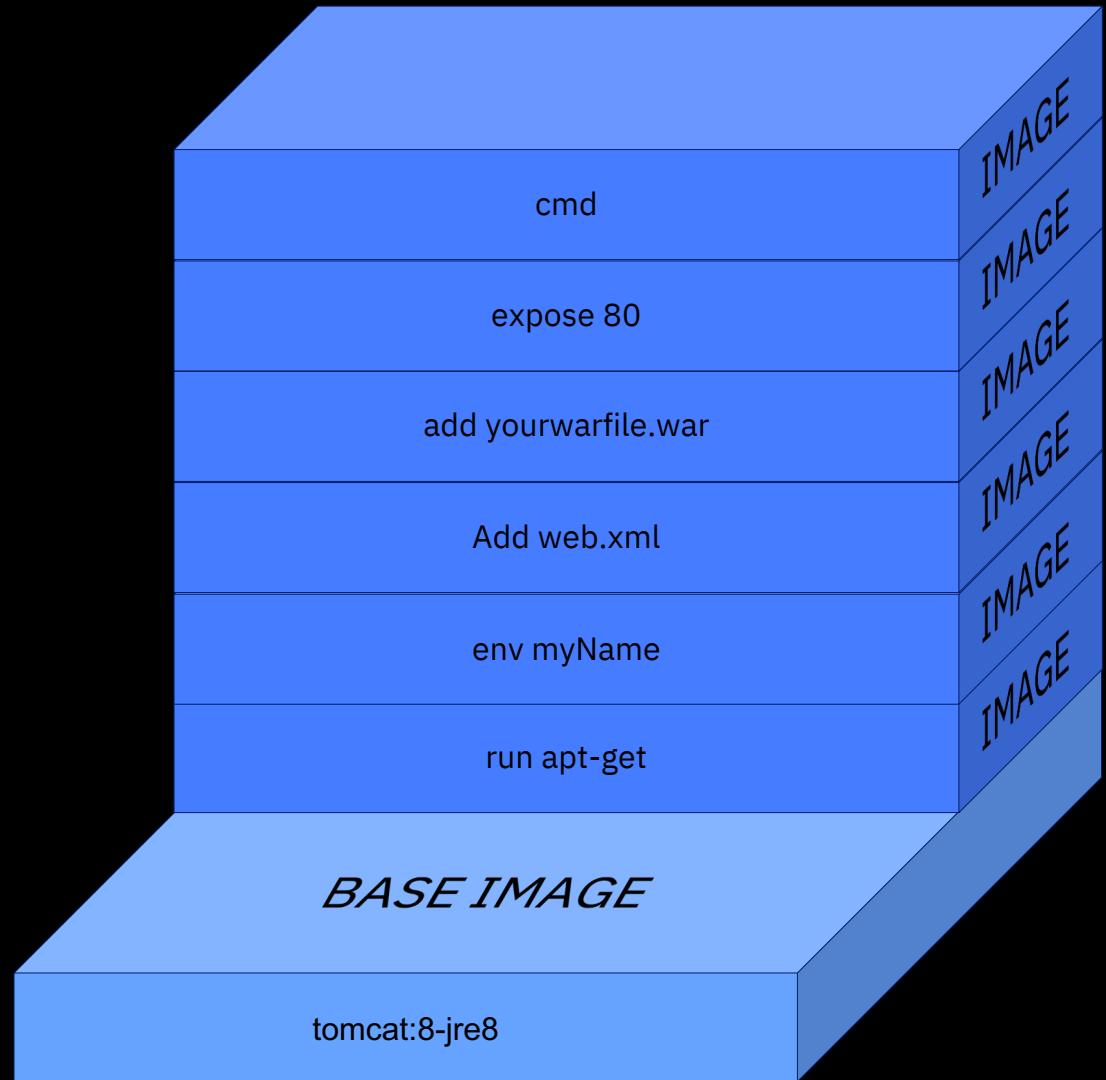
# Run command
RUN apt-get update && apt-get -y upgrade

# Set variables
ENV myName John Doe

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

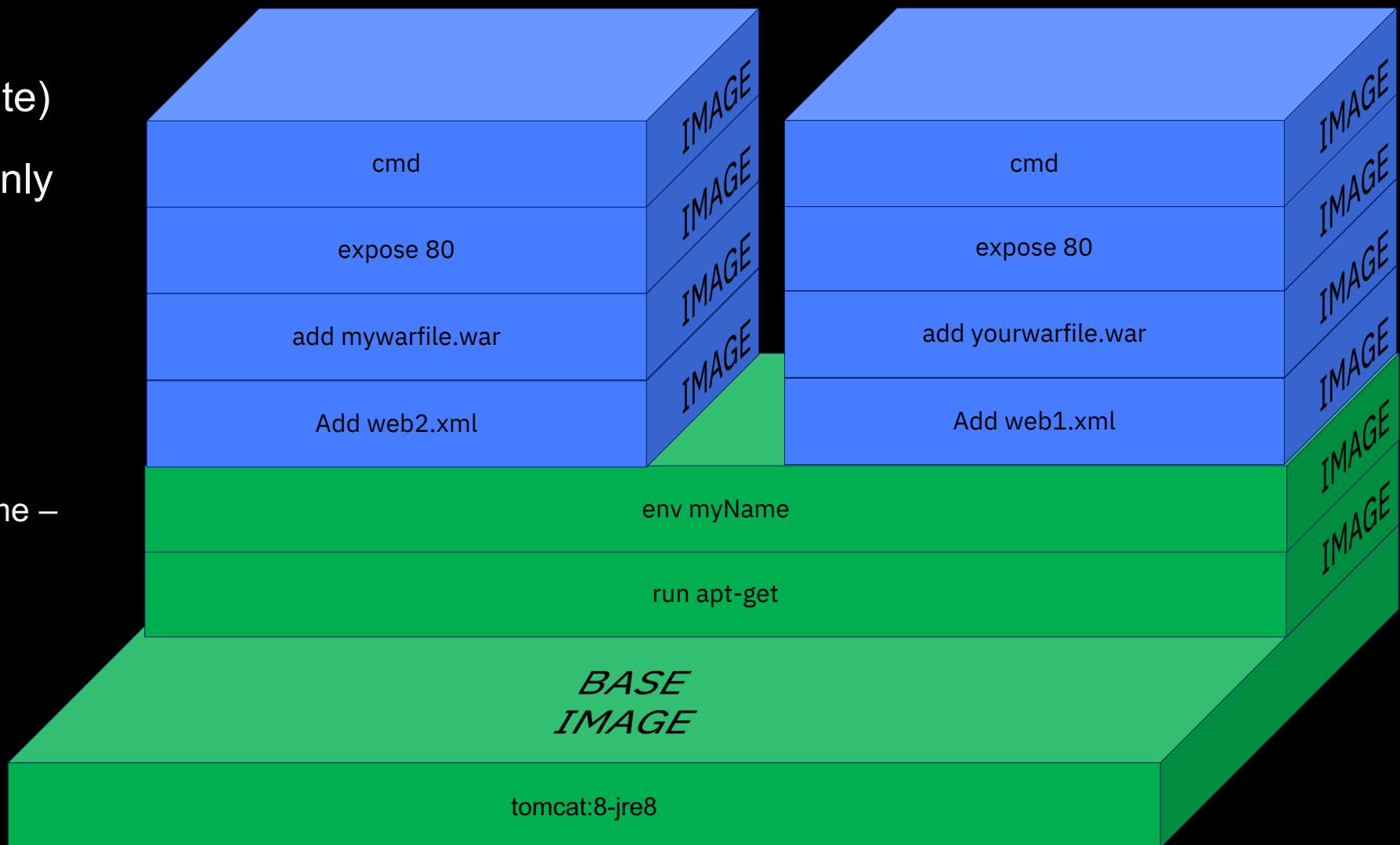
# Expose the port of the web application
EXPOSE 80

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

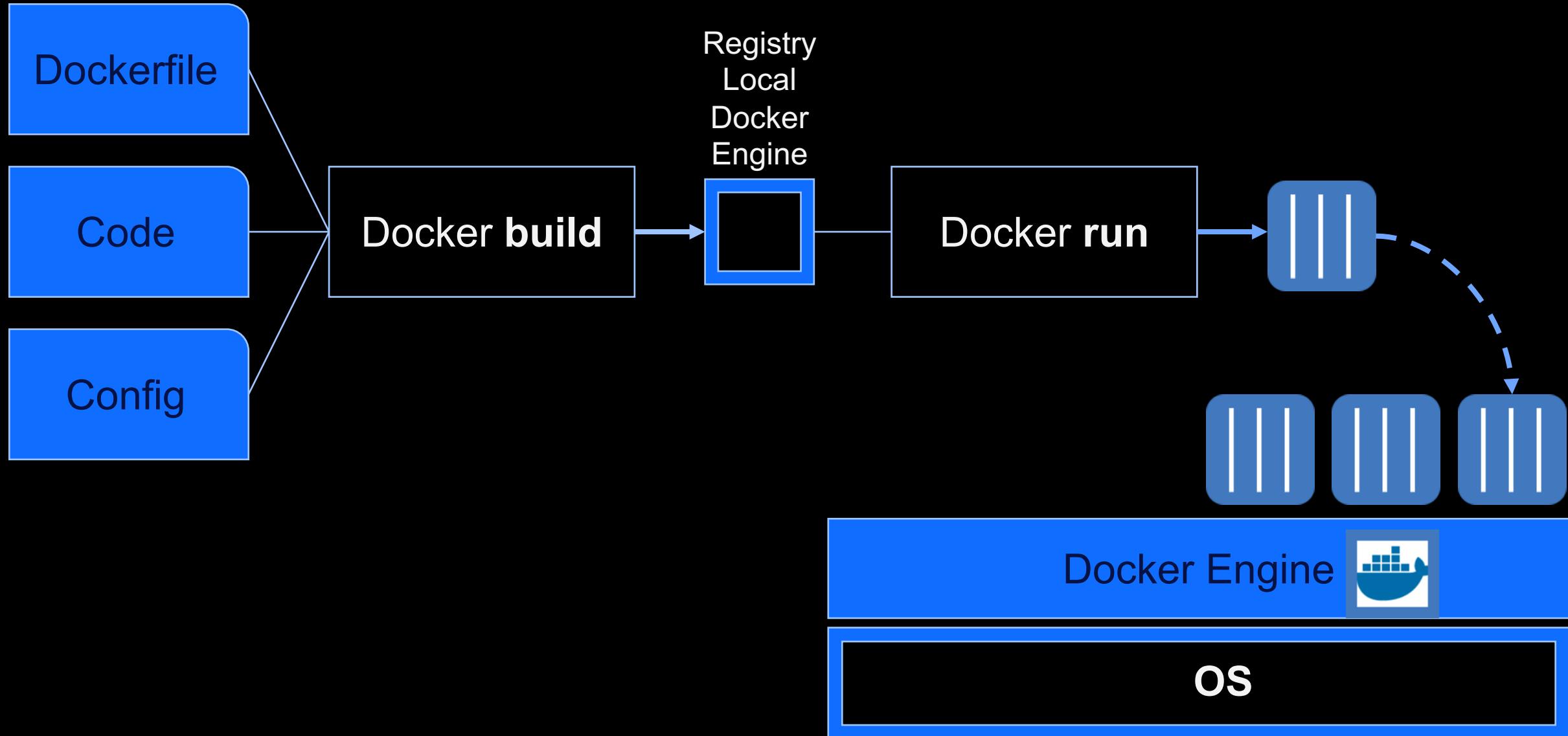


# Docker Basics – Layered File System

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



# Docker Basics – Run



# Docker Basics – Run

## Docker Layers

### Inheritance

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



# Docker Basics – Run

## Docker run

- Local (containerd)

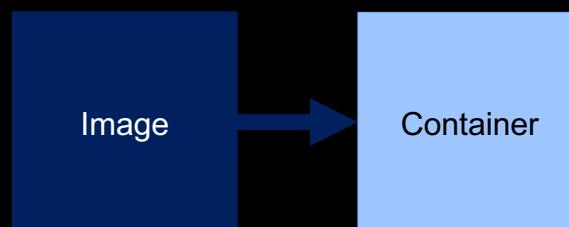
```
tpouyer at Laptop in ~/Development/workspace/docker/docker-jke-db on master
$ docker run -d postgres:latest
4eec29ae5eaa20798b1af8fa37873f7fc20cbb7cb789986b44f66cd94a784e0a

tpouyer at Laptop in ~/Development/workspace/docker/docker-jke-db on master
$ docker ps
CONTAINER ID        IMAGE               COMMAND      CREATED             STATUS              PORTS
ORTS                NAMES
4eec29ae5eaa        postgres:latest     "/docker-entrypoint.s"   5 seconds ago    Up 4 seconds          5432/tcp

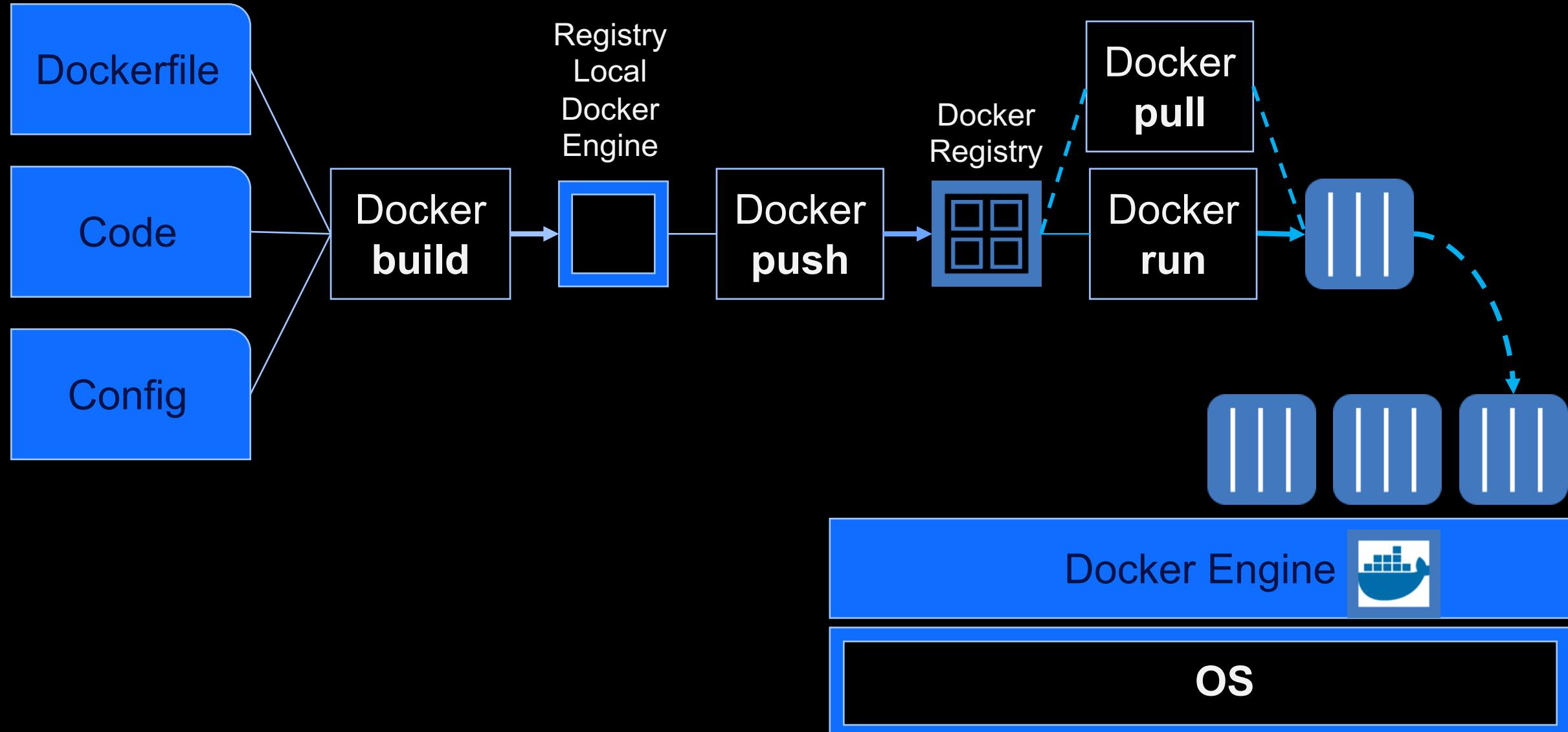
tpouyer at Laptop in ~/Development/workspace/docker/docker-jke-db on master
$ 
```

```
docker run -d -e MYVAR=foo -p 8080:80 myimage:1.0.0
docker run -ti myimage:1.0.0 /bin/bash
```

...



# Docker Basics – Store, Retrieve & Run with registry



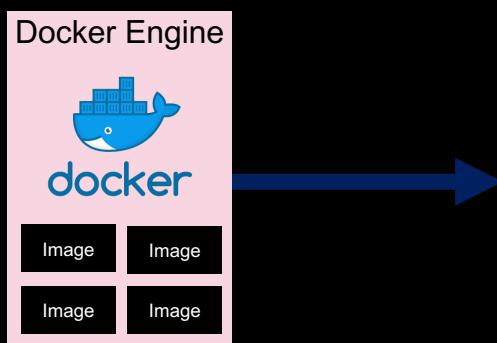
# Docker Basics – Store & Retrieve

## Docker Registry

- Private Local
- Public Docker Hub
- Private Shared

docker images

```
tpouyer at Laptop in ~/Development/workspace/docker/docker-jke-db on master
$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
stackinabox/demo-docker   dev      4e2a1f6cc1a4    25 hours ago  528.3 MB
stackinabox/demo-jke     1.1      a596fbf2c3b7    3 days ago   672 MB
stackinabox/demo-jke     dev      a596fbf2c3b7    3 days ago   672 MB
stackinabox/demo-jke     latest   a596fbf2c3b7    3 days ago   672 MB
stackinabox/demo-jke     1.1      a596fbf2c3b7    3 days ago   672 MB
stackinabox/demo-jke     latest   a596fbf2c3b7    3 days ago   672 MB
stackinabox/jke-web      latest   a4be0cdad8bc    4 days ago   462.3 MB
stackinabox/demo-jke-db   dev      54bdbf42b999    4 days ago   327.5 MB
stackinabox/jke-db        ddc     54bdbf42b999    4 days ago   327.5 MB
stackinabox/urbanbane-deploy-client  6.2.2.0  53c878fbf034    7 days ago   477 MB
websphere-liberty        crosservices: 7e27c3fb9c96   10 days ago  445.7 MB
mysql                   architecture 5.6      a896fd82dcd5   13 days ago  327.5 MB
mysql                   principles   latest   f3694c67abdb   13 days ago  400.1 MB
[...]
$ ls
[...]
Line 31, Column 17
Tab size: 4
```



myregistry/webapp

webapp:latest  
webapp:v2.0  
webapp:v1.1  
webapp:v1.0

myregistry/mongodb

mongodb:latest  
mongodb:v2.0  
mongodb:v1.1  
mongodb:v1.0

# Docker Basics – Store & Retrieve with registry

## Docker Registry

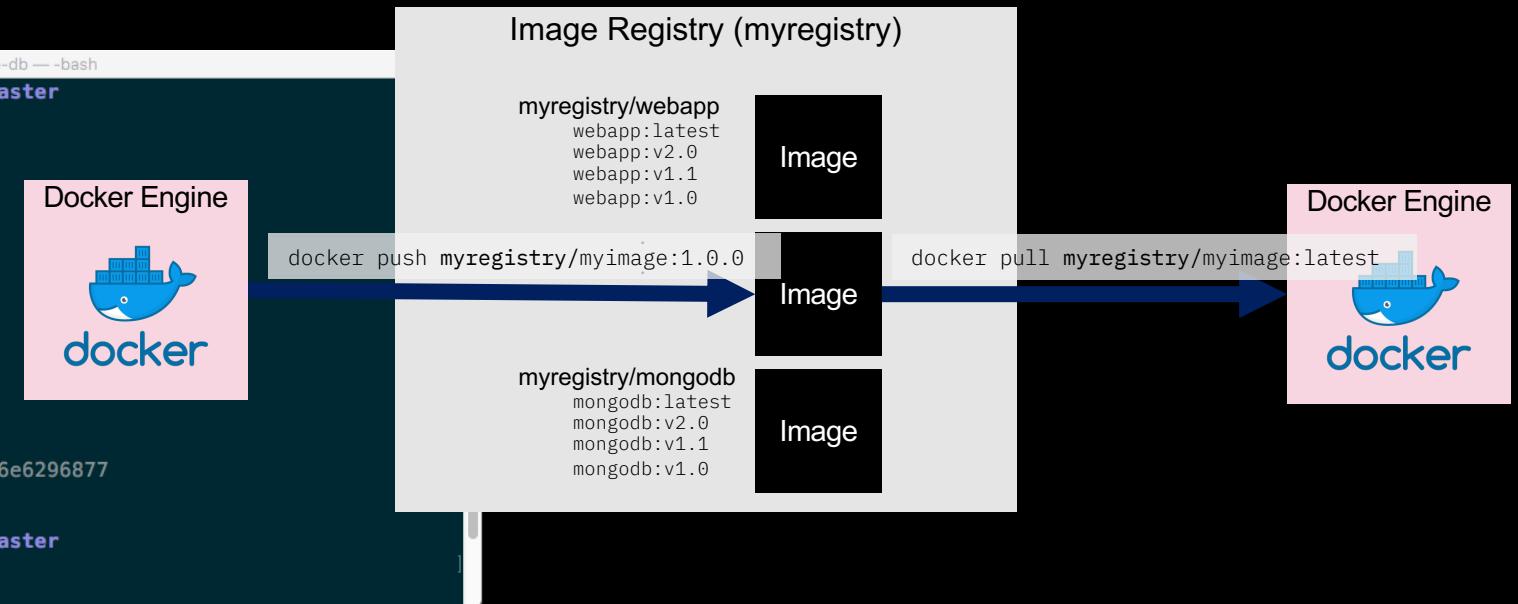
- Private Local
- Public Docker Hub
- Private Shared

```
docker tag myimage:1.0.0 myregistry/myimage:1.0.0
docker push myregistry/myimage:1.0.0

docker pull myregistry/myimage:latest
```

```
tpouyer at Laptop in ~/Development/workspace/docker/docker-jke-db on master
$ docker pull postgres:latest
latest: Pulling from library/postgres
  Digest: sha256:0842a7ef786aa2658623085160cb38451eb3d40856e7d222ae0069b6e6296877
Status: Downloaded newer image for postgres:latest

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



# Docker Basics – Registries

Hosting image repositories

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

Public and Private registries

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

Login into the registry

- Docker login domain:port



# Docker Recap

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

# QUESTIONS?



Kubernetes Workshop Series  
**Let's get real**

03

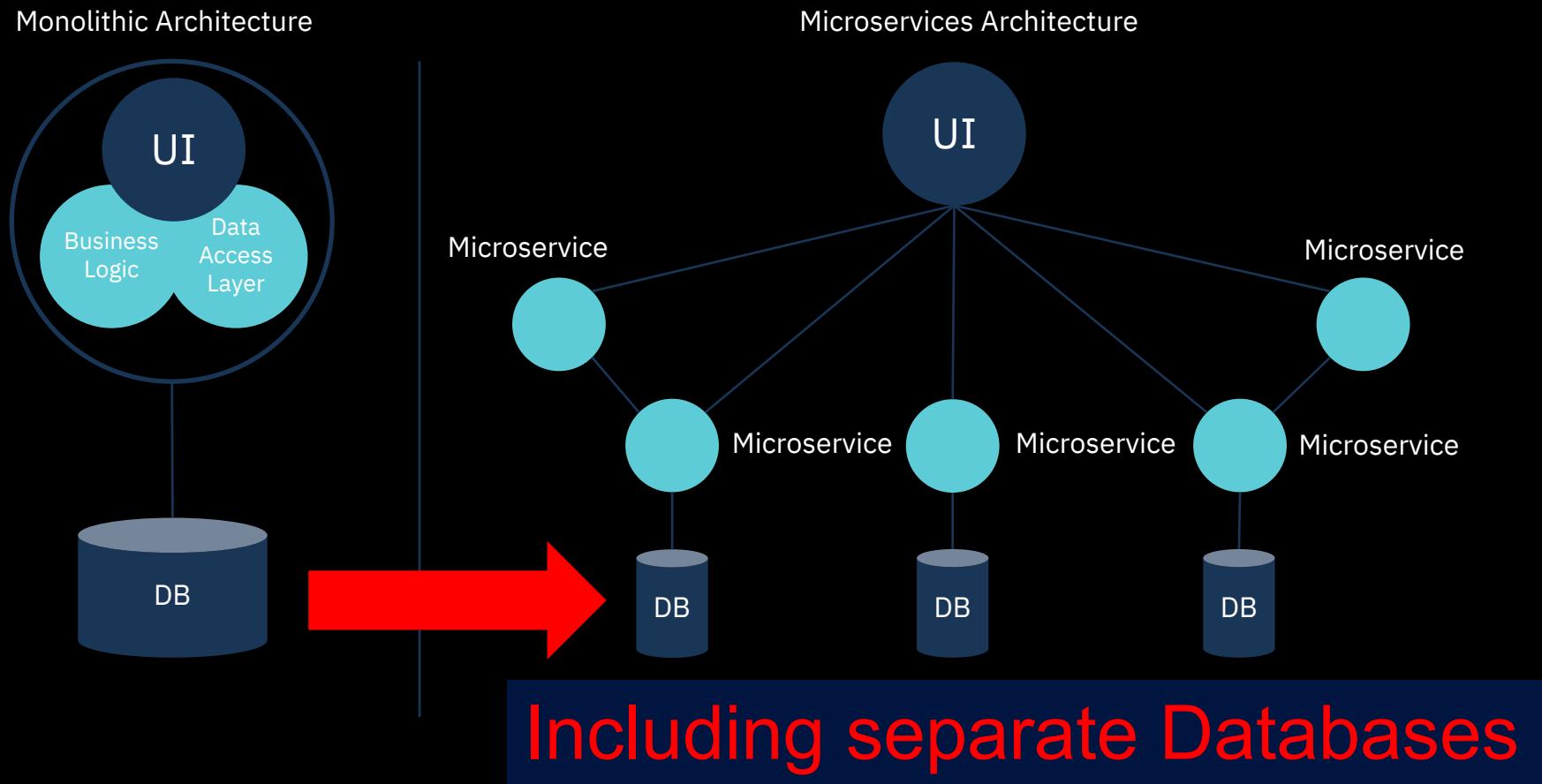


Containerizing your WAR file  
doesn't mean you're doing  
microservices.

- 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?

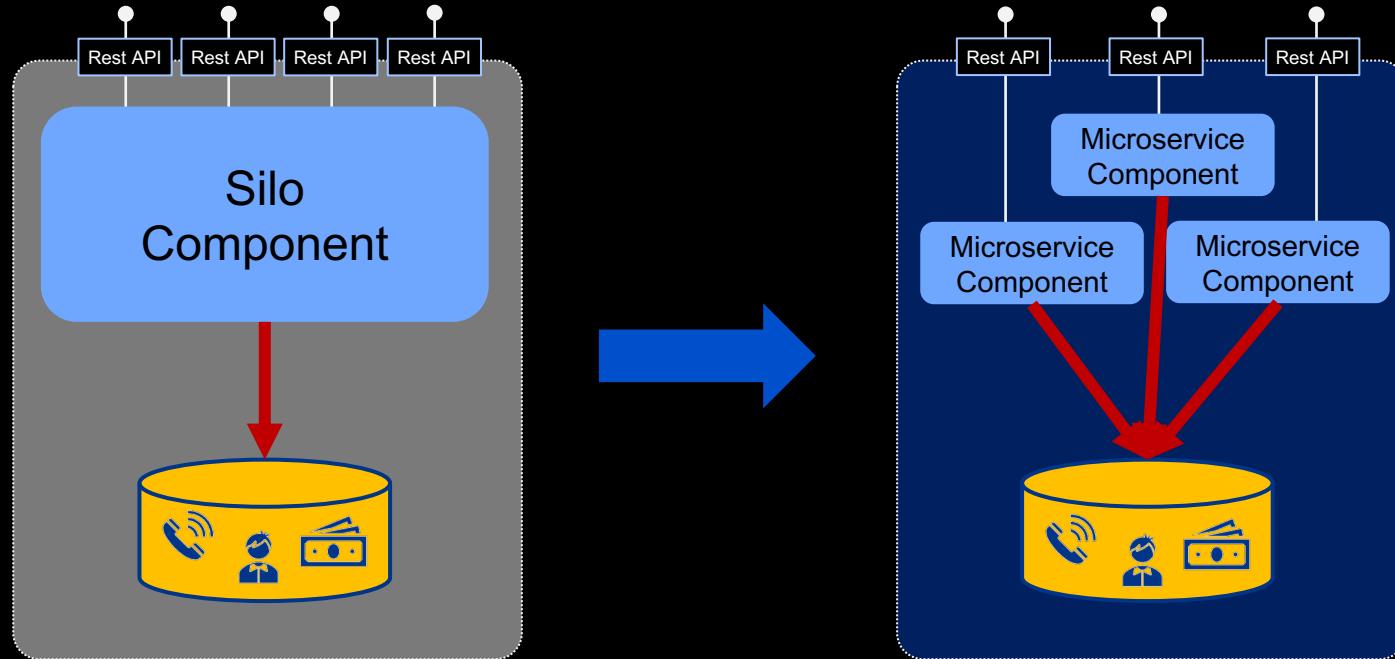
# Microservices – Database refactoring

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



# Microservices – Database refactoring

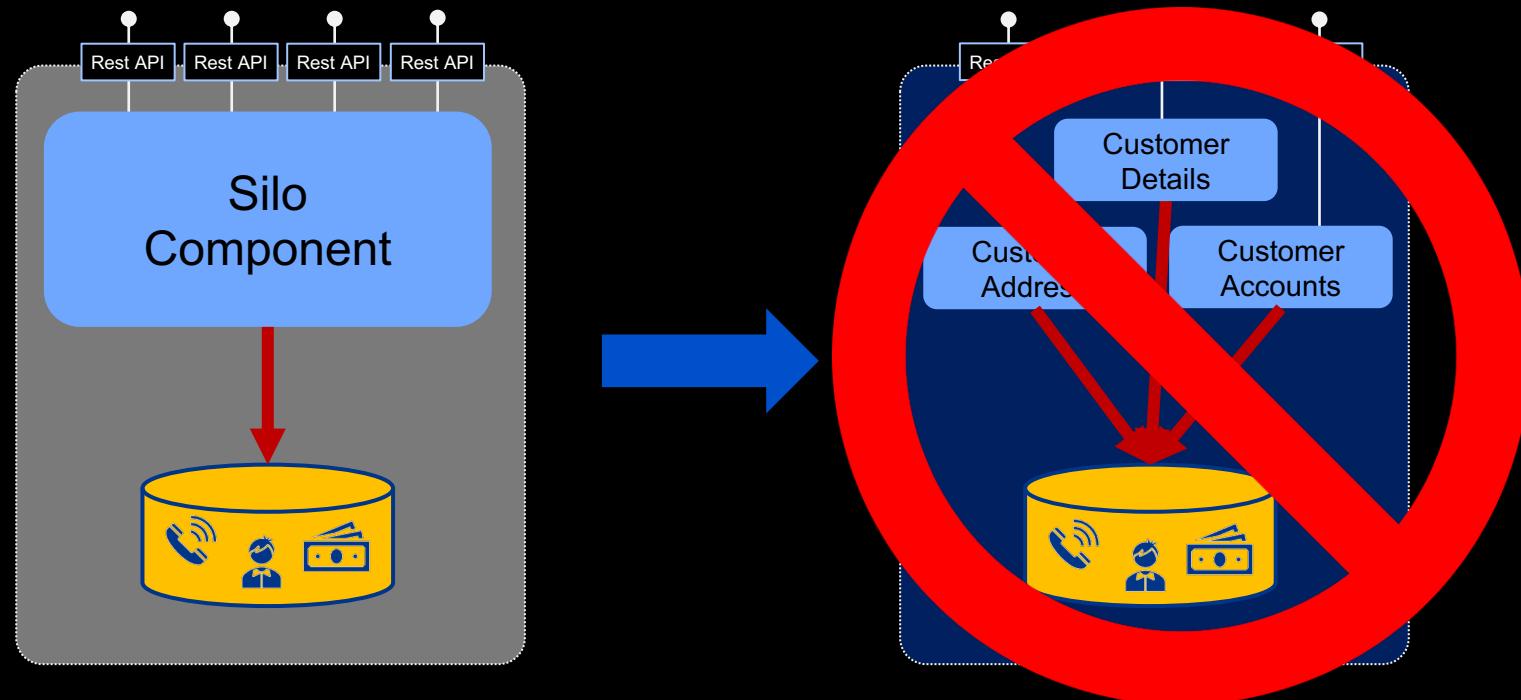
## Common mistake



# Microservices – Database refactoring

## Data Isolation

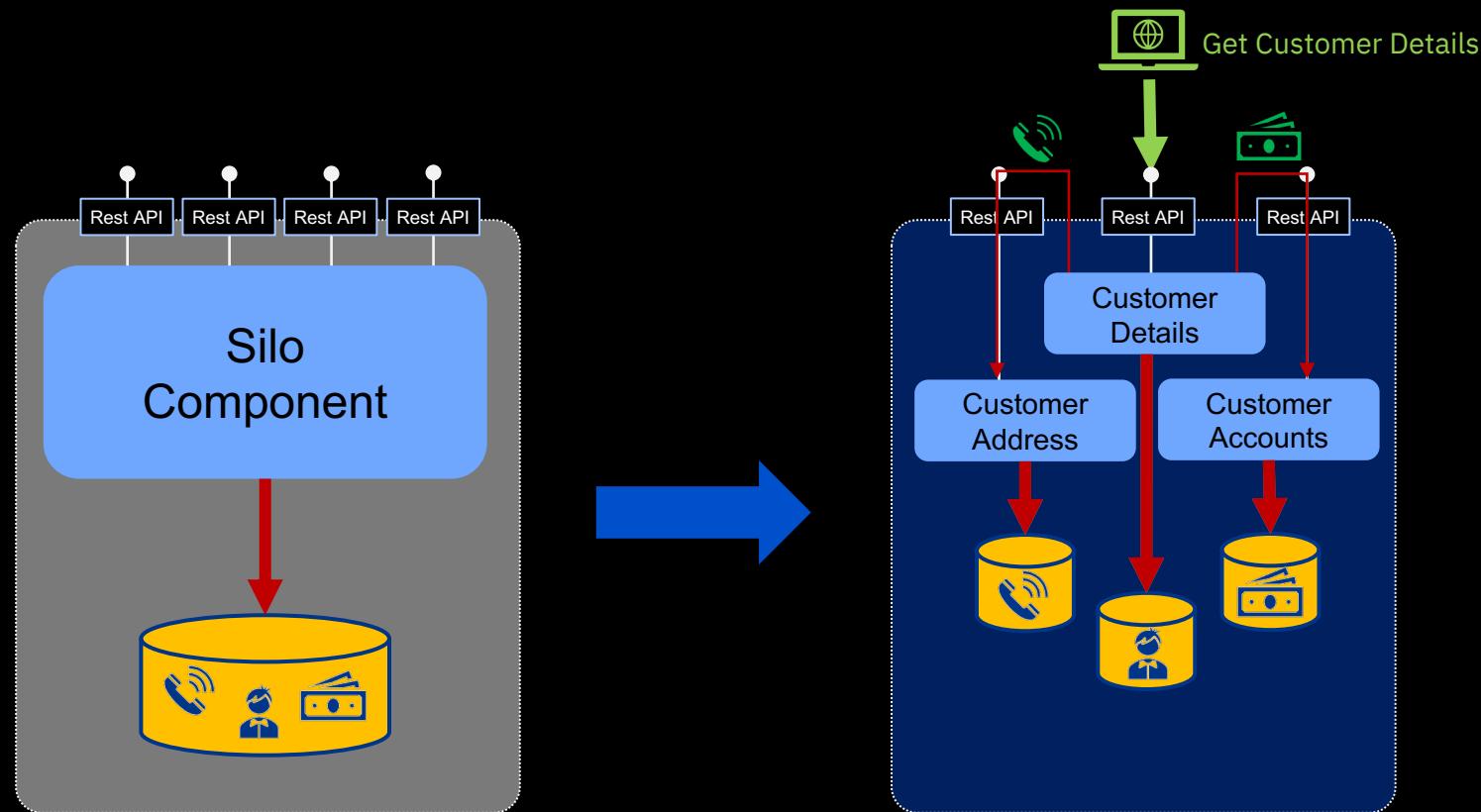
Each microservice must keep its own data. One typical error that happens in the beginning of a microservice transformation is the use of a centralized database, the same way it was used in the monolithic application. This creates a single point of failure and dependency between the microservices.



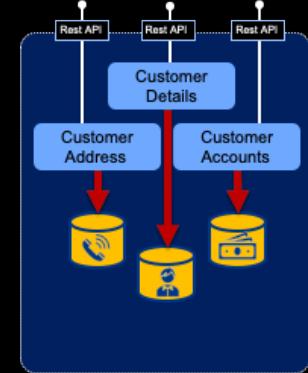
# Microservices – Database refactoring

## 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.



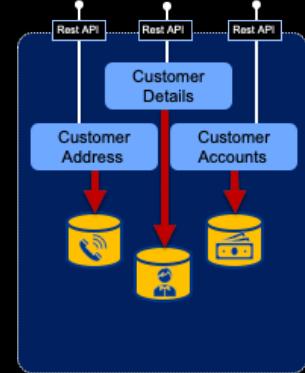
# Microservices – per-service database



## Benefits

- Avoid unexpected data modification – the API enforces consistency
- 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

# 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)

# Microservices Good or bad idea?

# Microservices – reasons not to use them

## Some thoughts

- If speed is your goal, microservices aren't the solution (MVP, doesn't have to scale)
- Applications that require tight integration between individual components and services (real-time processing, ...)
- At what point is it in its lifecycle? Mission-critical?
- Not all applications are large enough to break down into microservices

# QUESTIONS?

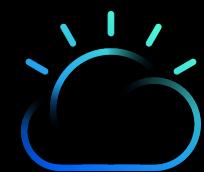




◦◦ Kubernetes Workshop Series  
**Docker - Hands-On**

04





# Starting Course JTC01 Docker

Select JTC01 course to begin



Welcome to the Kubernetes Course | Kubernetes Training - Mozilla Firefox

Wed 11 Nov 08:32

Welcome to the Kubernetes Course

Kubernetes Training

Search

- Introduction
- JTC01 Docker**
- JTC02 Kubernetes Labs
- JTC10 Istio
- JTC14 Kubernetes Ansible Operators
- JTC16 Kubernetes Security Labs
- JTC17 Kubernetes Advanced Security

Welcome to my courses around Kubernetes.

I have developed them over the last year and have held several online and in-person meetups to teach fellow geeks about the awesome world of containers and orchestration.

You can find the sources here

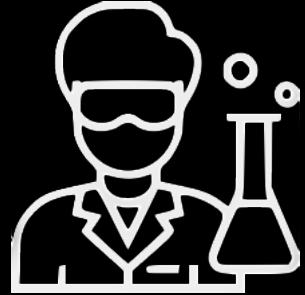
Videos <https://www.youtube.com/channel/UCIS0jmGOQrG2AKKPktJYj9w/>

Training Documents [https://github.com/niklaushirt/k8s\\_training\\_public](https://github.com/niklaushirt/k8s_training_public)

Training files <https://github.com/niklaushirt/training.git>

©2020 Niklaus Hirt

Introduction >



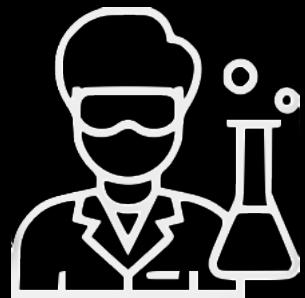
JTC01 Labs

Lab 1 : Docker basics

Lab 2 : Docker – Working with Containers

- Build a Docker image
- Run a Docker image
- Use the Portainer tool
- Deploy a more complex Docker application
- Push a Docker image into a registry

Lab 3 : Docker internals



READY  
SET  
GO!!!!

Duration: 60 mins

# QUESTIONS?



# Kubernetes Workshop Series

## **Kubernetes**

05



# 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

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

# Everybody Loves Containers

## But when you go



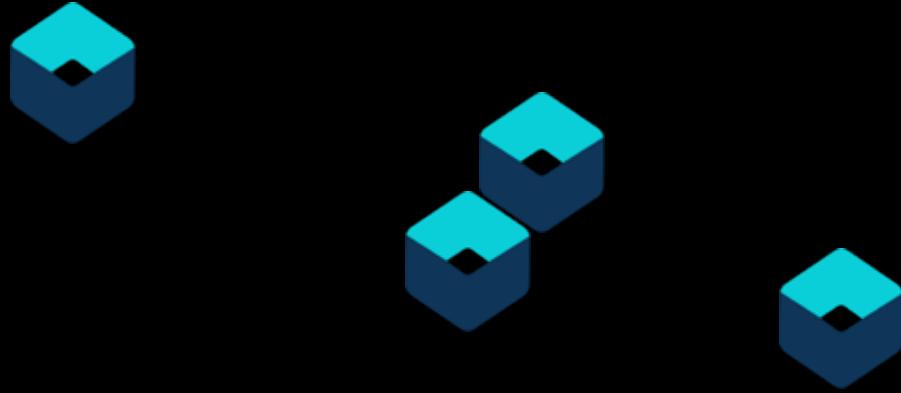
From this....



To this....



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



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

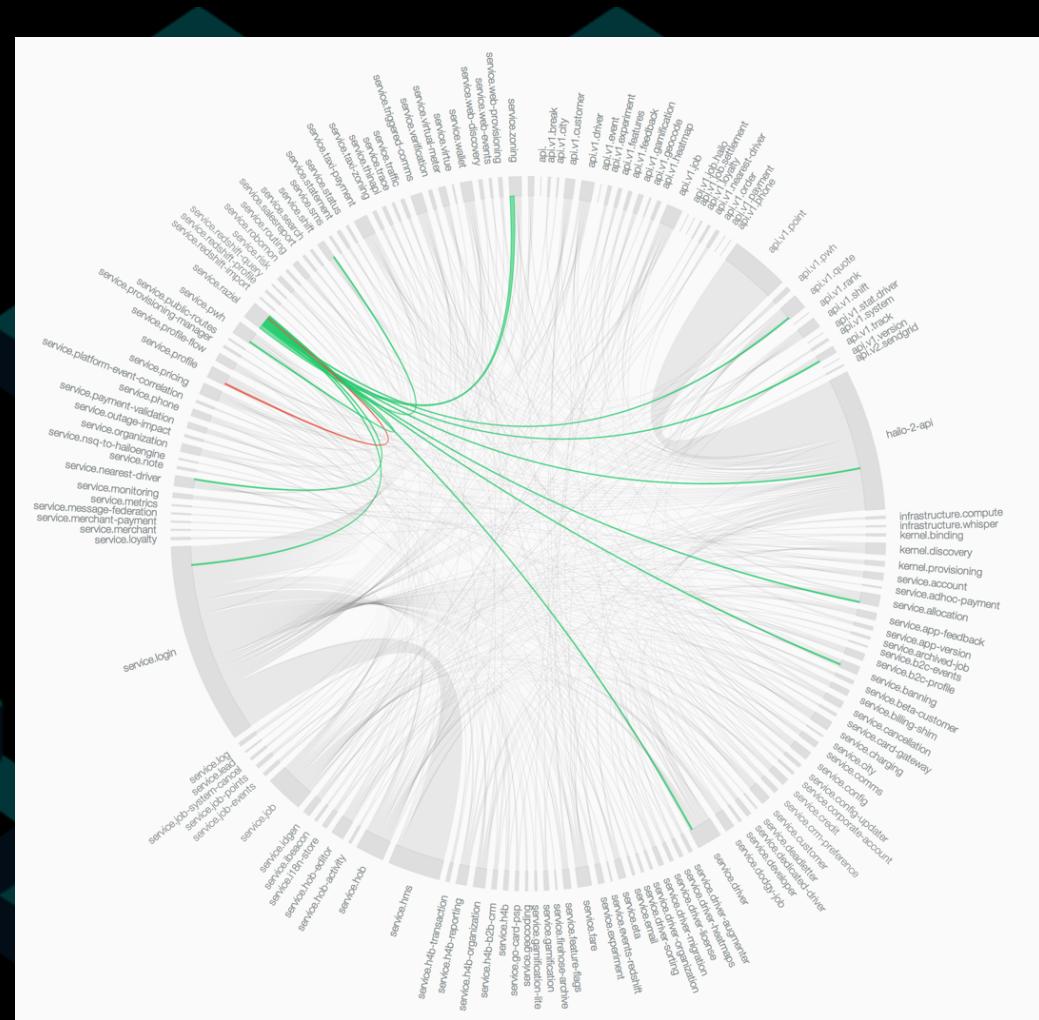


# Pets vs Cattle

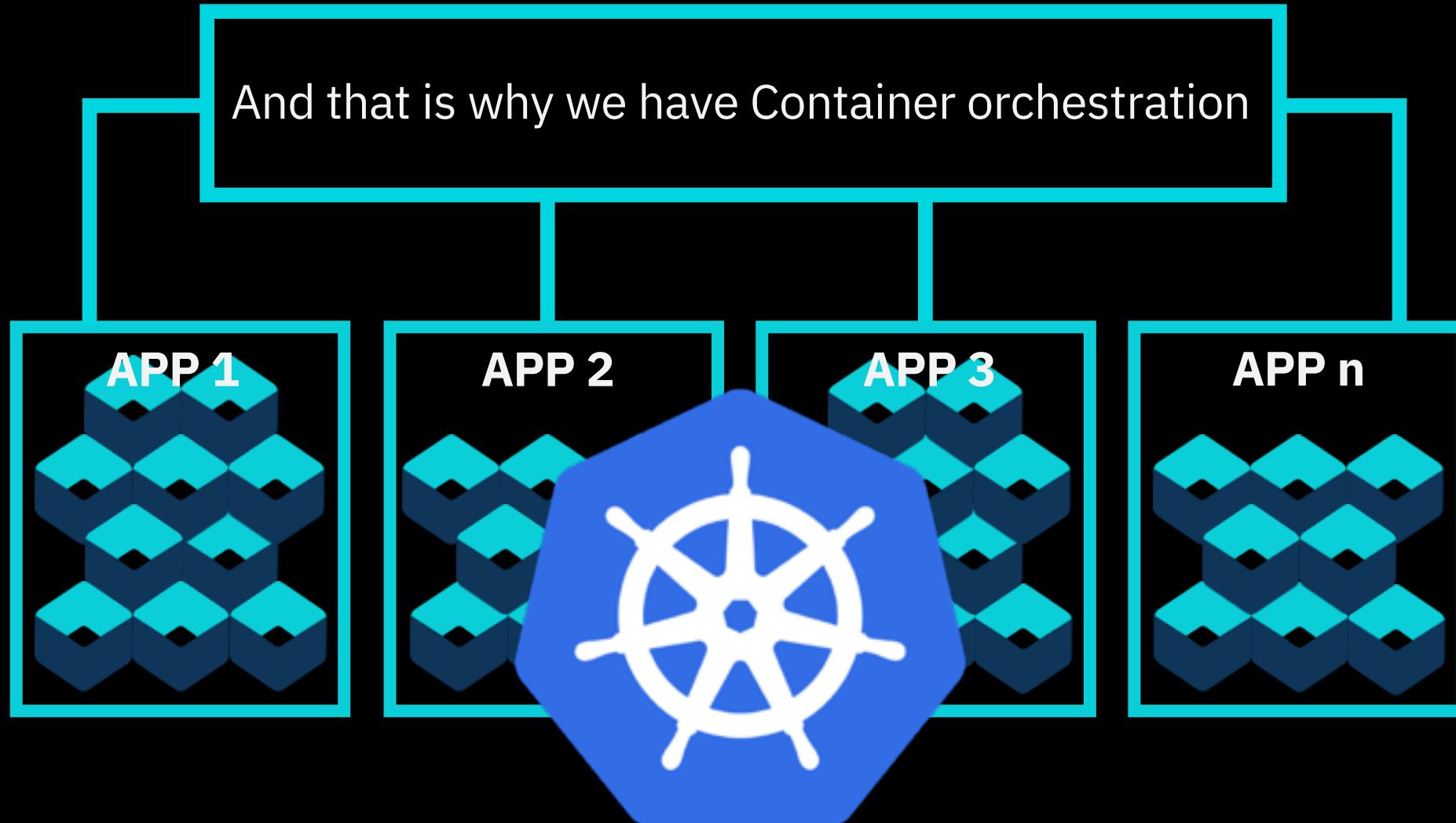
But soon you have many applications, many instances...

# The trade off

**Improved delivery velocity**  
in exchange for  
**increased operational complexity**



# Enter Kubernetes (K8s)





## 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

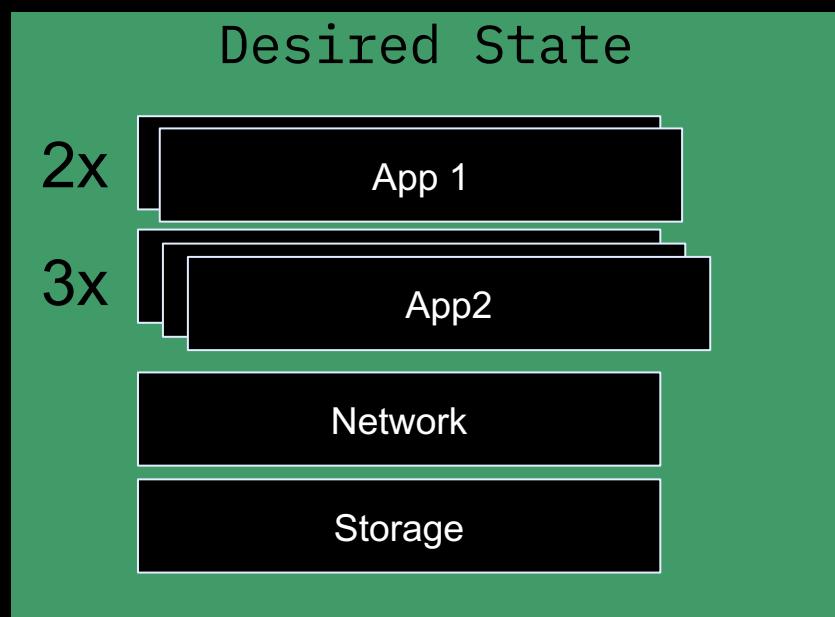
By contrast, in a declarative system, **the user** knows the desired state, supplies a representation of the desired state to the system, then **the system** reads the current state and determines the sequence of commands to transition the system to the desired state.

# Kubernetes – Declarative System

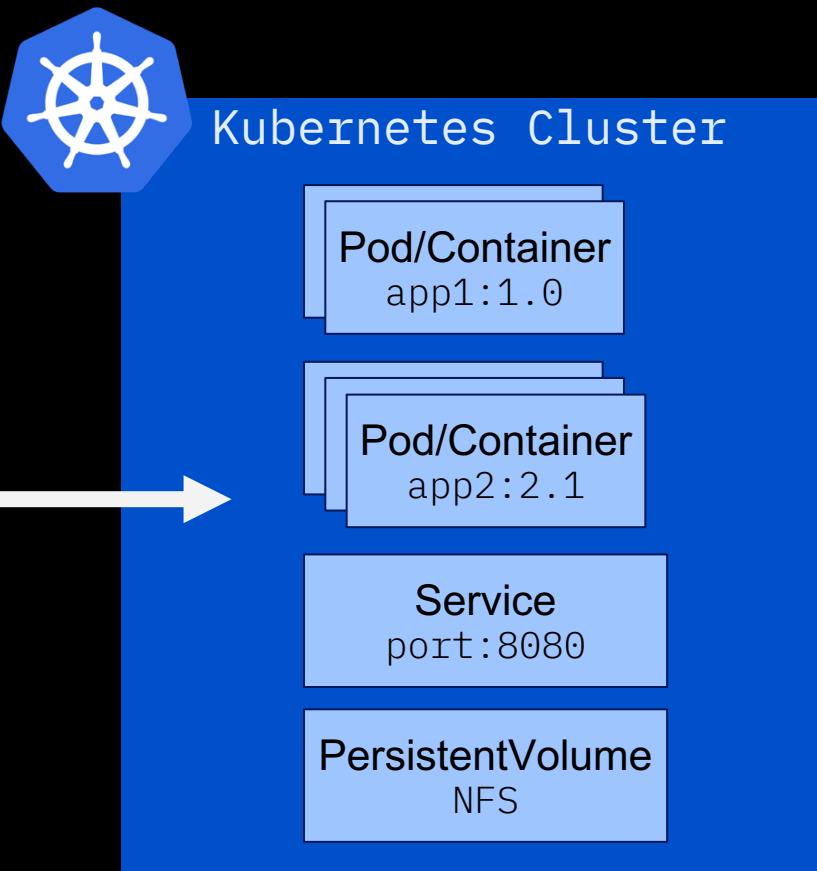


## The Desired State

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



```
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
      component: klusterlet
      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: "354b8990aab44c9988a0edfd101b128"
        productVersion: "3.1.2"
    spec:
```

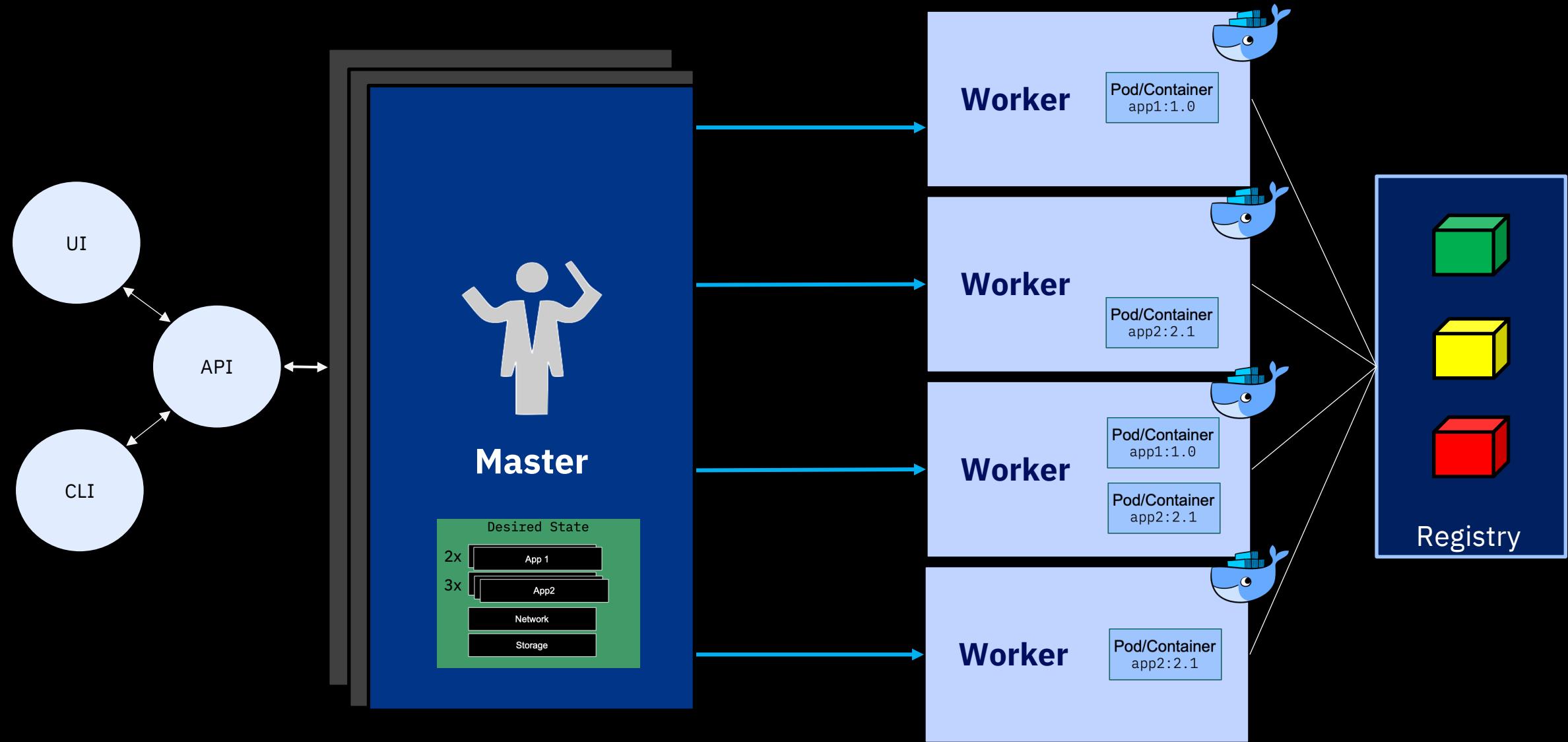


# Kubernetes Strengths



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

# Kubernetes Management Architecture



# Kubernetes Management Architecture



**Nodes** – hosts that run Kubernetes applications

API

## Master nodes

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

- Controller Manager Server
- Scheduler

## Worker nodes

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



# Kubernetes Management Architecture

## Pods

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

## Deployment

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

## ReplicaSet

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

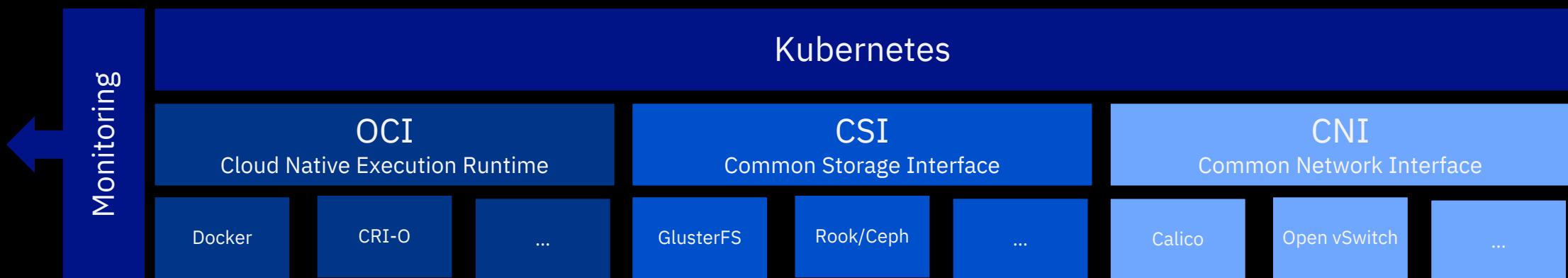
Master Node

Worker Node 3

## Service

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

# Kubernetes Cluster Layers



# Kubernetes – Why 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. Every container run using the Docker CLI 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.



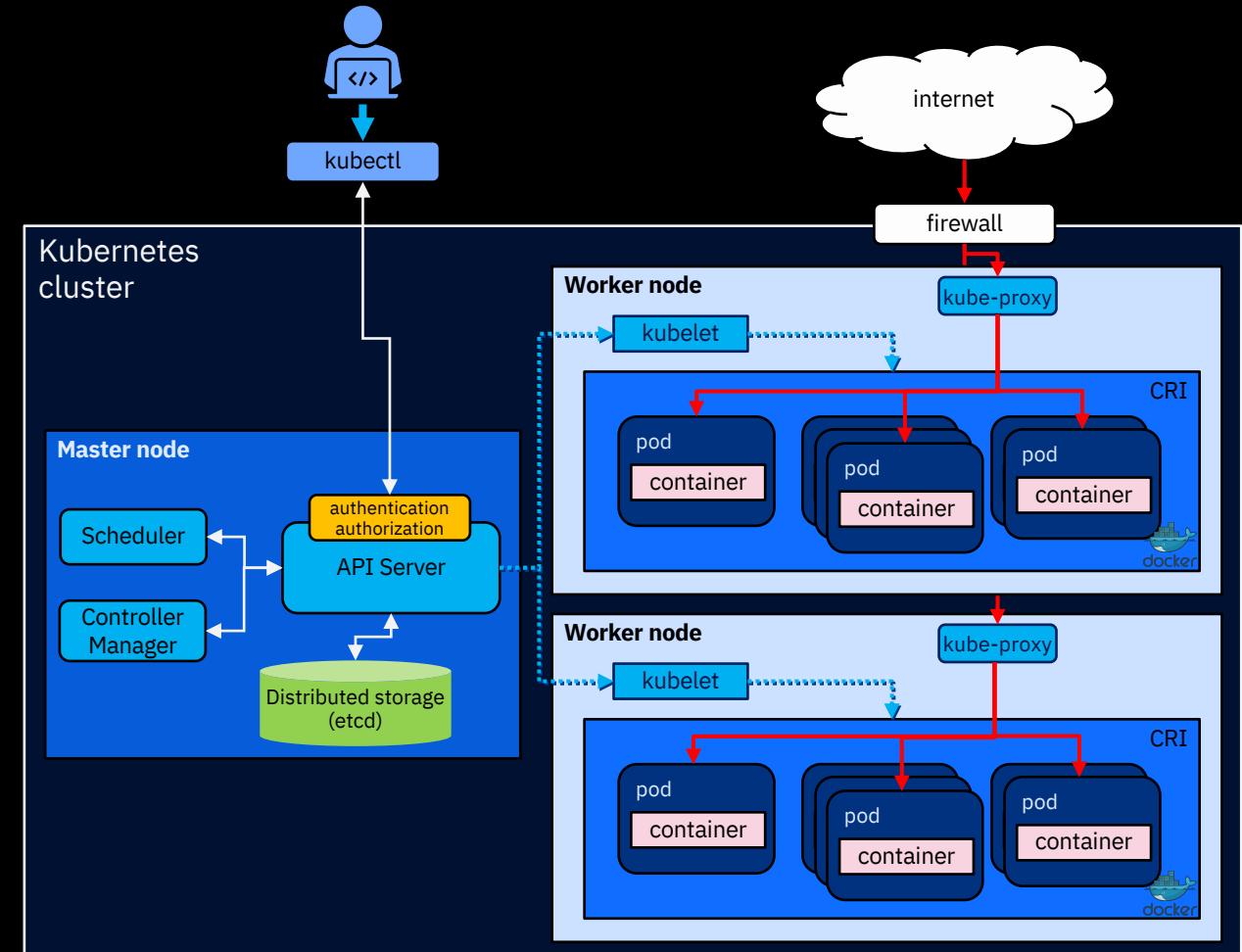
# Kubernetes Cluster Architecture

## Master node

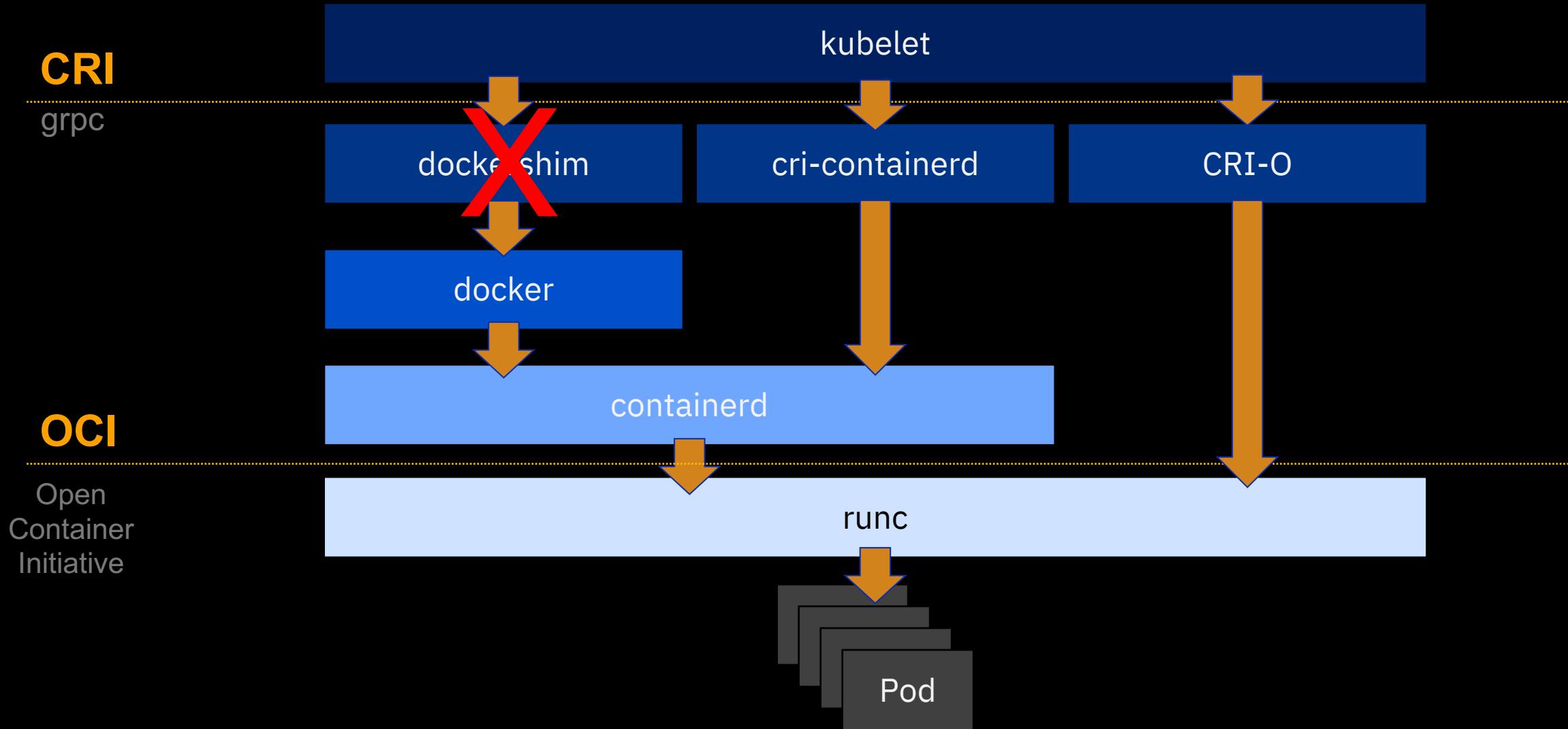
- Node that manages the cluster
- Scheduling, replication & control
- Multiple nodes for HA

## 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 Runtime Interface



# kubectl – talking to the Cluster

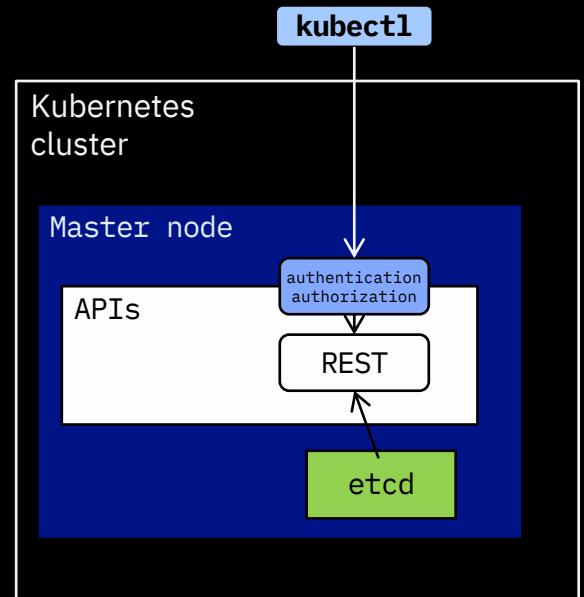


`kubectl` is a command line interface for running commands against Kubernetes clusters (read state, create objects, ...).

`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
```

To start, open Terminal and run `./welcome.sh`



# 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`

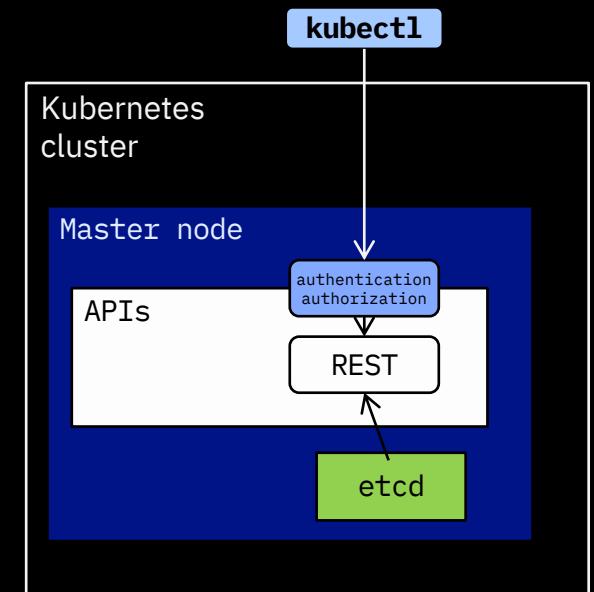
List Pods in Namespace

`kubectl describe nodes <node-name>`

Details about K8s object

`kubectl logs <pod-name>`

Get the logs for a Pod

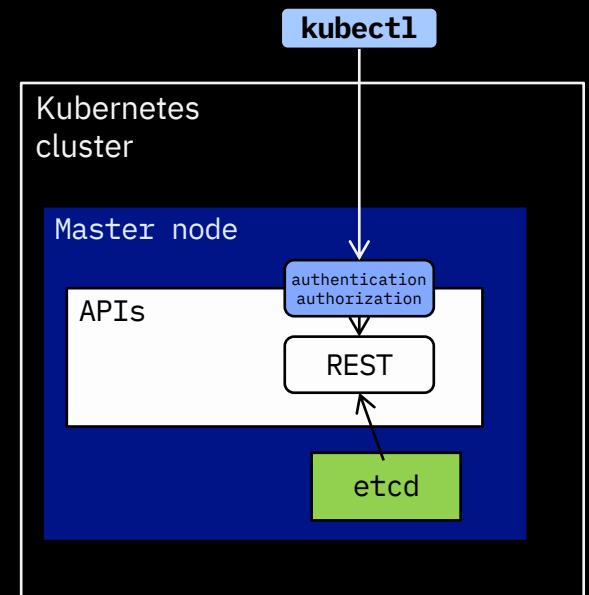


# kubectl – talking to the Cluster



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

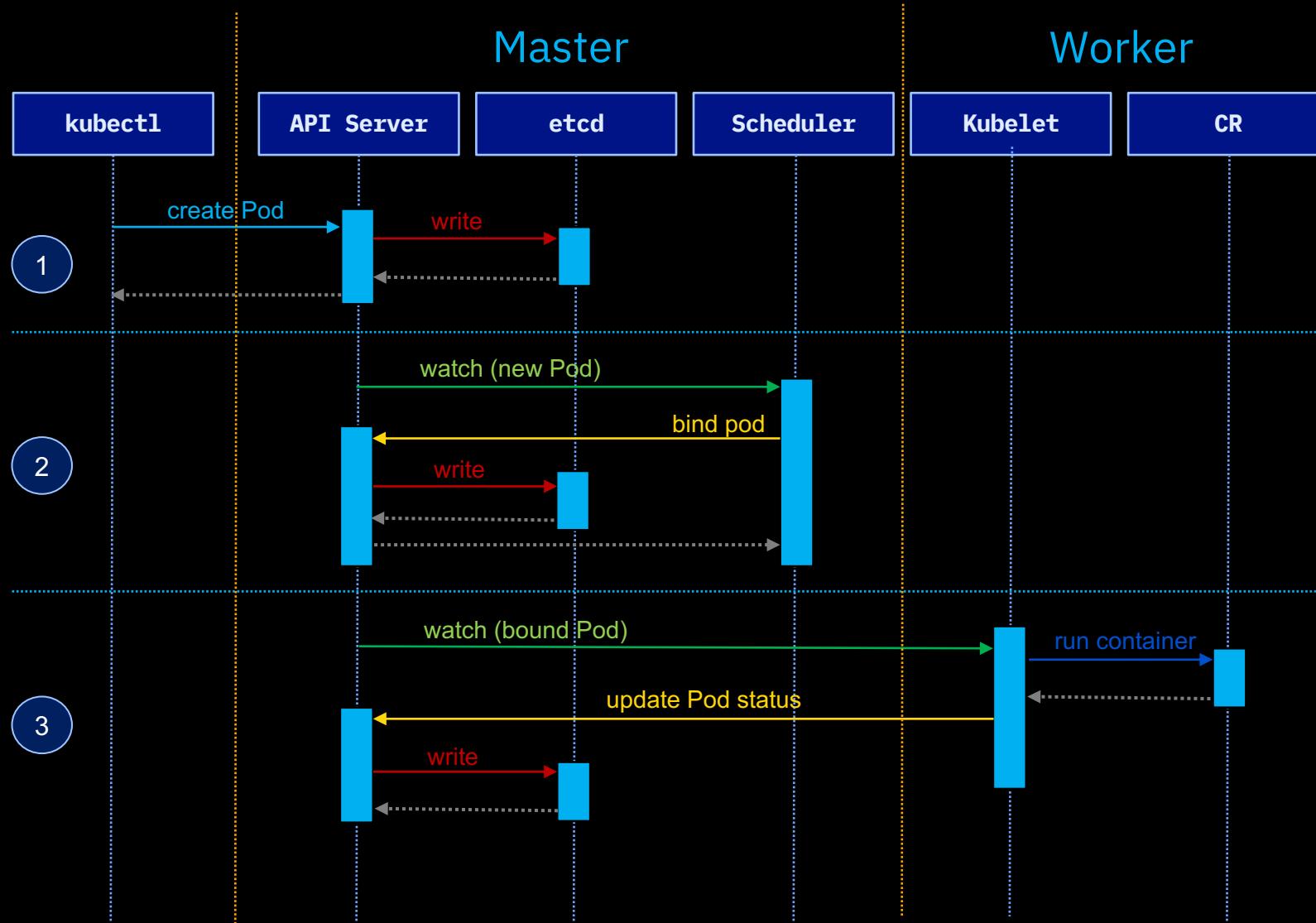
- Kubernetes uses `etcd` as a key-value database store.
- It stores the `configuration` of the Kubernetes cluster in etcd.
- It also stores the `actual state` of the system and the `desired state` of the system in etcd.
- Anything you might read from a `kubectl get xyz` command is coming from etcd.
- 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

# Pod



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

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



10.0.0.1



10.0.0.2

# Creating a Pod



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

Or

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

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

example.yaml

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



10.0.0.1



10.0.0.2



## Name

- Each resource object by type has a unique name

## Namespace

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

# 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

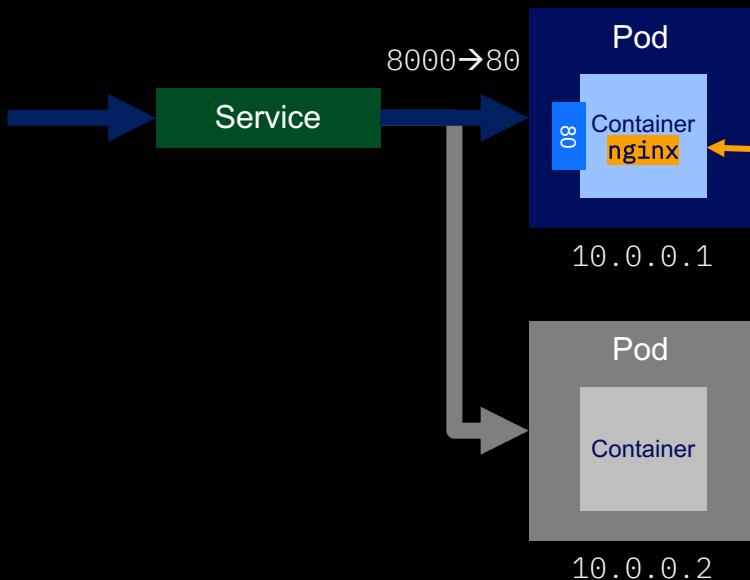
Deployment details	
Type	Detail
Name	cam-ui-basic
Namespace	services
Created	39 minutes ago
Labels	app=cam-ibm-cam,chart=ibm-cam-1.3.0,heritage=Tiller,name=cam-ui-basic,release=cam
Selector	name=cam-ui-basic
Replicas	Desired: 1   Total: 1   Updated: 1   Available: 1
RollingUpdateStrategy	Max unavailable: 1   Max surge: 1
MinReadySeconds	0

Service details	
Type	Detail
Name	cam-ui-basic
Namespace	services
Created	40 minutes ago
Type	ClusterIP
Labels	app=cam-ibm-cam,chart=ibm-cam-1.3.0,heritage=Tiller,name=cam-ui-basic,release=cam
Selector	name=cam-ui-basic
Cluster IP	10.0.0.165
External IP	-
Port	cam-ui-basic 39002/TCP
Node port	None
Session affinity	None

# 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

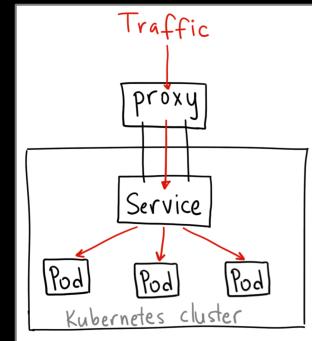


```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  ports:
    - port: 8000
      targetPort: 80
      protocol: TCP
  selector:
    app: nginx
```

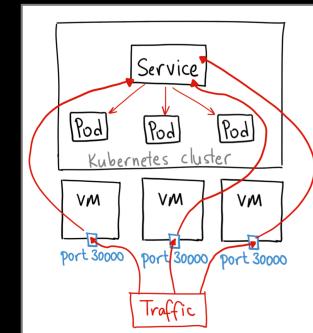


# Service

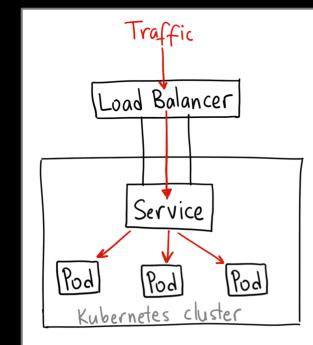
**ClusterIP**: This type exposes the service on the cluster internal IP. This means that the service is only reachable from within the cluster.



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



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



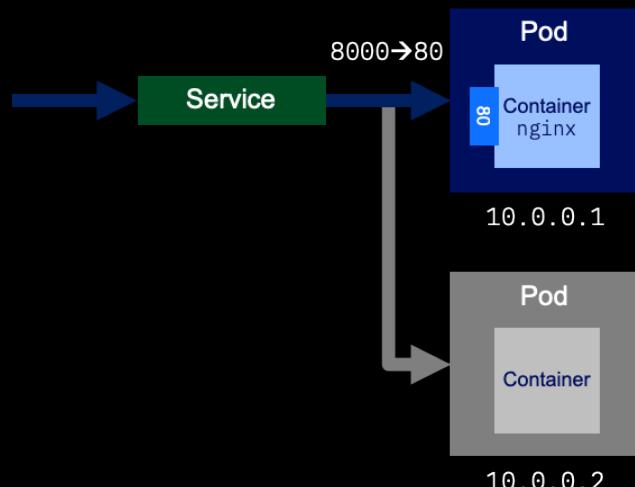


# Creating a Service

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

```
kubectl get services
```

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



example.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  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

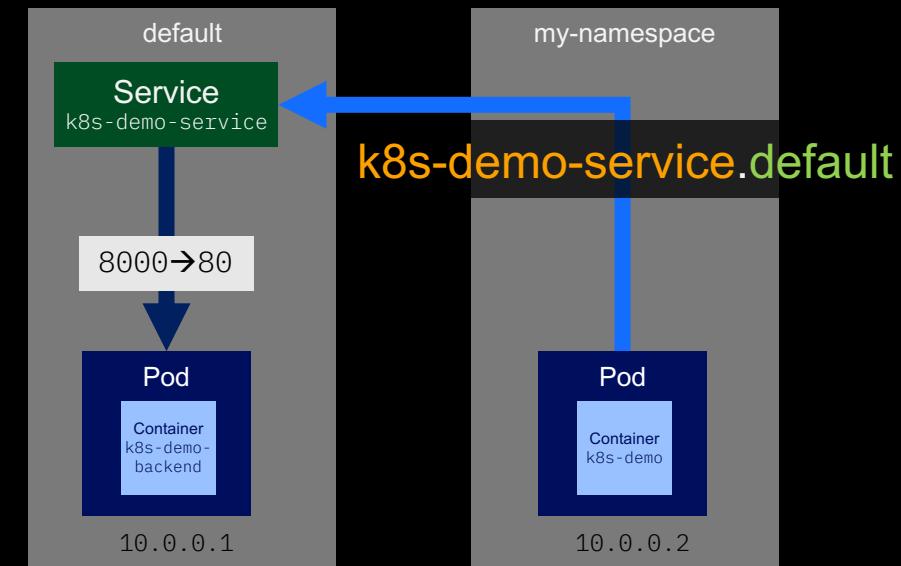
**k8s-demo-service**

When referred from another namespace

**k8s-demo-service.default**

Fully qualified name

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

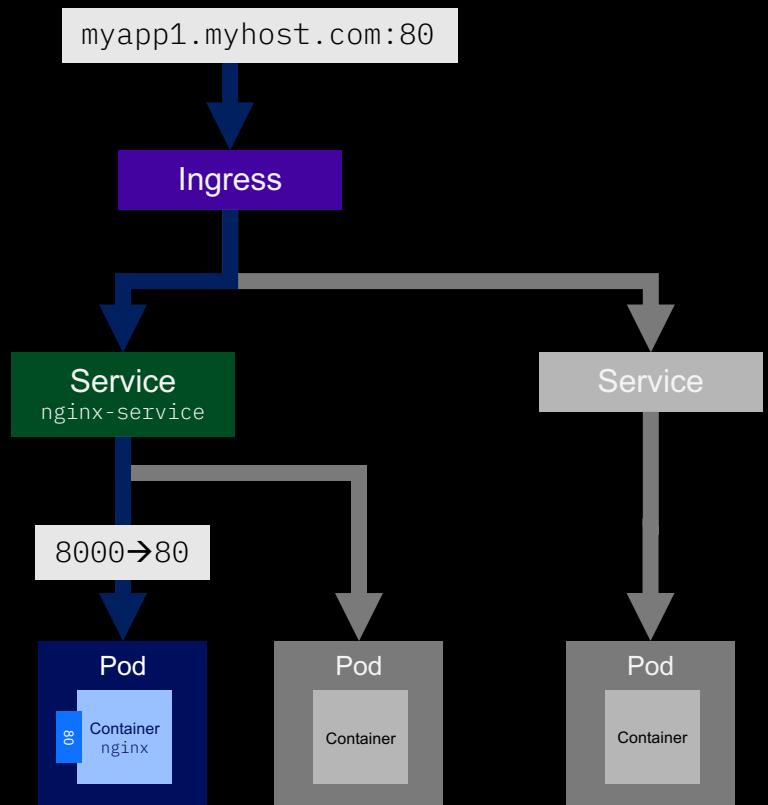


# Ingress



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 a loadbalancer.

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

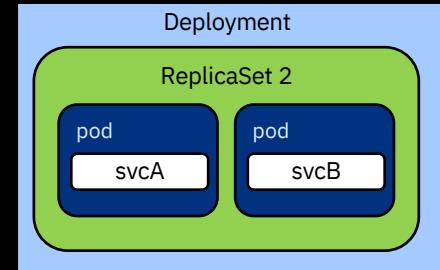


# Deployments & ReplicaSets



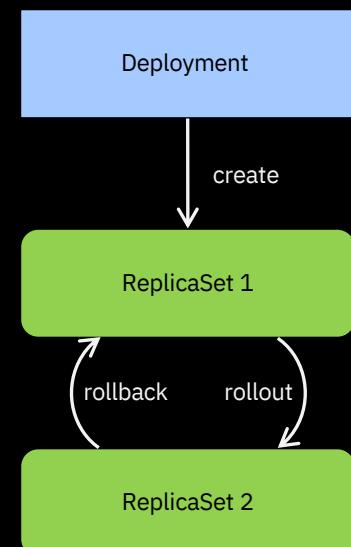
## Deployment

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



## ReplicaSet

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

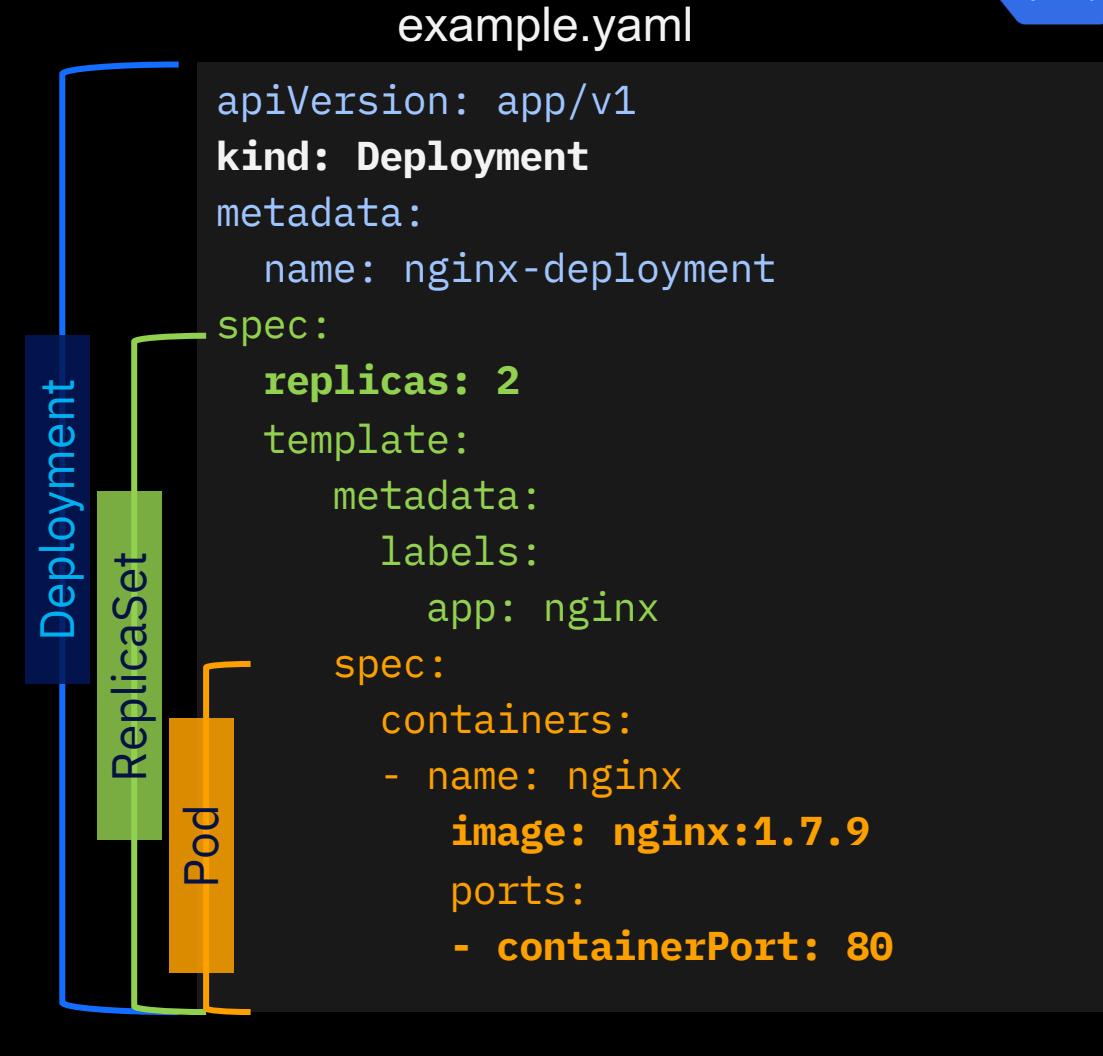


# Deployment



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

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



# StatefulSet, DaemonSet, Job...



## StatefulSet

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

## DaemonSet

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

## Job

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

## Cron Job

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

# 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

# Configuring Resources and Containers



## ConfigMap

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

## Secret

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

# Configuring Resources and Containers



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

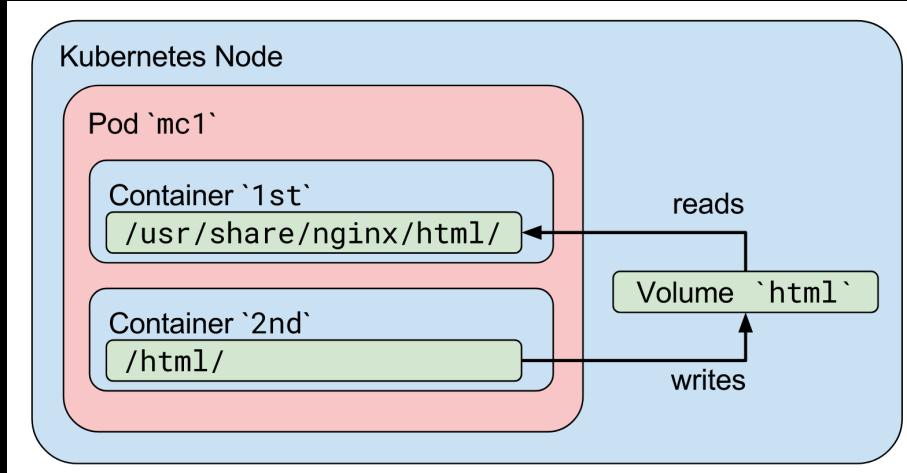
```
kind: Deployment
spec:
  containers:
    - name: test-container
      image: xxx
      ...
  env:
    - name: SPECIAL_LEVEL_KEY
      valueFrom:
        configMapKeyRef:
          name: example-config
          key: enemies
```

Can be used as normal environment variable (here **SPECIAL\_LEVEL\_KEY=aliens**)

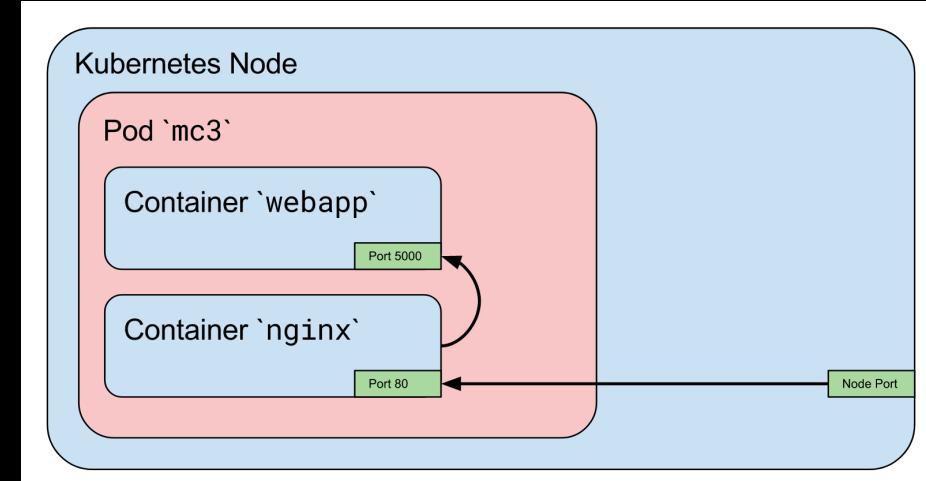
# Multi-Container Pod Design Patterns



## Shared Volumes

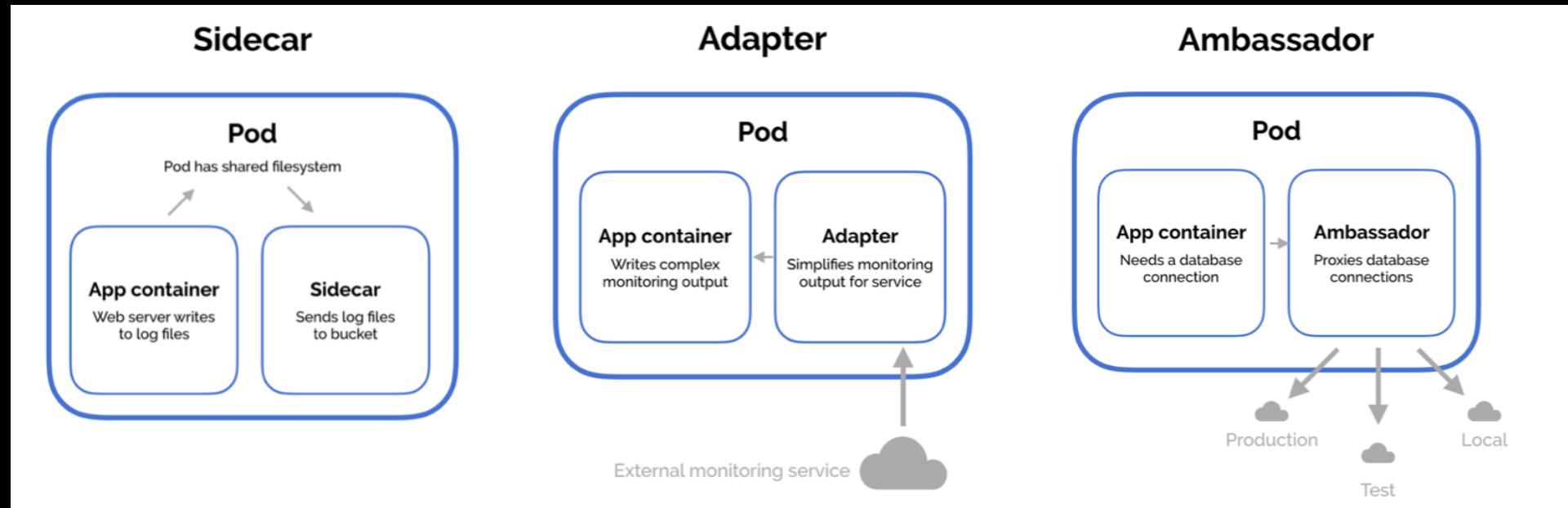


## No Network isolation



**In what order containers are being started in a Pod?**

# Multi-Container Pod Design Patterns



## Sidecar pattern

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

## Adapter pattern

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

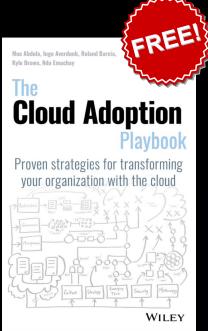
## Ambassador pattern

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

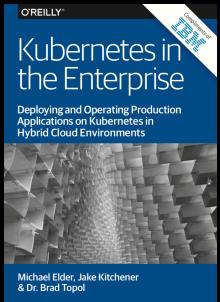
# QUESTIONS?



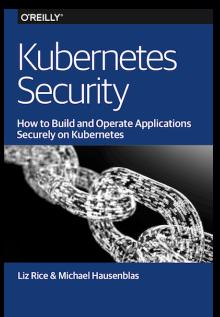
# Kubernetes – Some Reading Tips



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/>

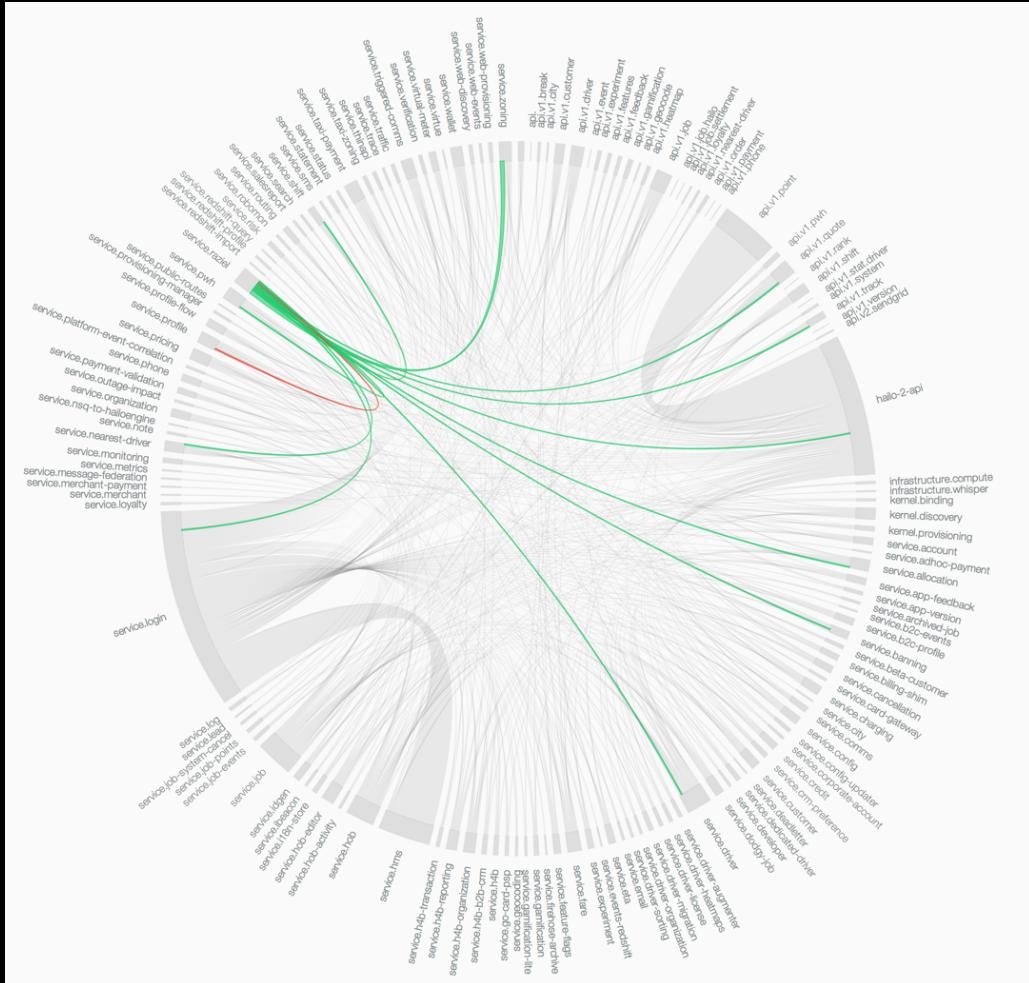
# Kubernetes Workshop Series

## **Mesh Networking**



# The trade off

**Improved delivery velocity**  
in exchange for  
**increased operational complexity**



# Addressing DevOps Challenges



#	CHALLENGE
<b>CHALLENGE 1</b>	ROLL OUT NEW VERSION WITHOUT DOWNTIME OR CHANGING CODE
<b>CHALLENGE 2</b>	HOW TO DO CANARY TESTING
<b>CHALLENGE 3</b>	HOW TO DO A/B TESTING
<b>CHALLENGE 4</b>	THINGS DON'T ALWAYS GO CORRECTLY IN PRODUCTION...
<b>CHALLENGE 5</b>	HOW CAN I LIMIT RATE FOR SOME OF MY SERVICES?
<b>CHALLENGE 6</b>	I NEED TO VIEW AND MONITOR WHAT IS GOING ON WHEN CRISIS ARISES
<b>CHALLENGE 7</b>	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

EDITION: EU

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

f t in e

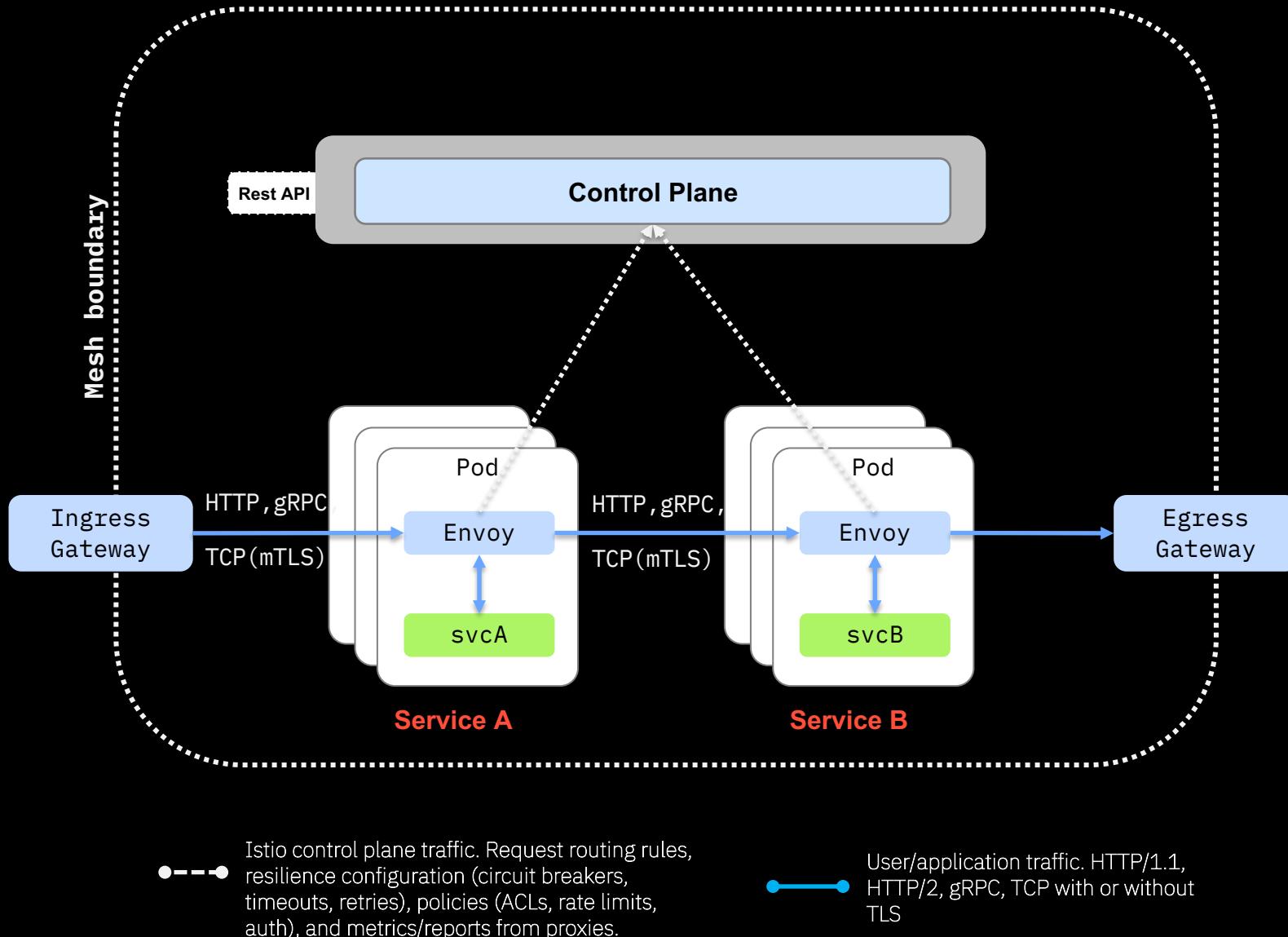
Today Google, IBM and Lyft announced the alpha release of Istio: a new open-source project that provides a uniform way to help connect, secure, manage and monitor microservices.

Launched in May 2017 by Google, Lyft and IBM

Open Source

Zero Code Changes

# ISTIO - Architecture



# ISTIO

## Custom resource definitions

```
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

## 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
```



<http://myapp.demo.com/demo>

**POD**  
helloworld  
version = v1

**SERVICE**  
helloworld  
selector  
app: helloworld

helloworld.namespace.svc.cluster.local

# 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
```



<http://myapp.demo.com/demo>

### URL Routing

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

### Traffic Splitting

```
kind: DestinationRule
metadata:
  name: helloworld-destination
spec:
  host: helloworld
  subsets:
    - name: v1
      labels:
        version: v1
    - name: v2
      labels:
        version: v2
```



# 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
```



<http://myapp.demo.com/demo>

### URL Routing

```
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
```

### Traffic Splitting

```
kind: DestinationRule
metadata:
  name: helloworld-destination
spec:
  host: helloworld
  subsets:
    - name: v1
      labels:
        version: v1
    - name: v2
      labels:
        version: v2
```

helloworld.namespace.svc.cluster.local



# 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: /
      route:
        - destination:
            host: reviews
            subset: v1
            weight: 100
        - destination:
            host: reviews
            subset: v2
            weight: 100
    - match:
        - headers:
            end-user:
              exact: jason
      route:
        - destination:
            host: reviews
            subset: v1
        - destination:
            host: reviews
            subset: v2
```

### Traffic Splitting

```
kind: DestinationRule
metadata:
  name: helloworld-destination
spec:
  host: helloworld
  subsets:
    - name: v1
      labels:
        version: v1
    - name: v2
      labels:
        version: v2
```

`helloworld.namespace.svc.cluster.local`

**POD**  
helloworld  
version = **v1**

**POD**  
helloworld  
version = **v2**

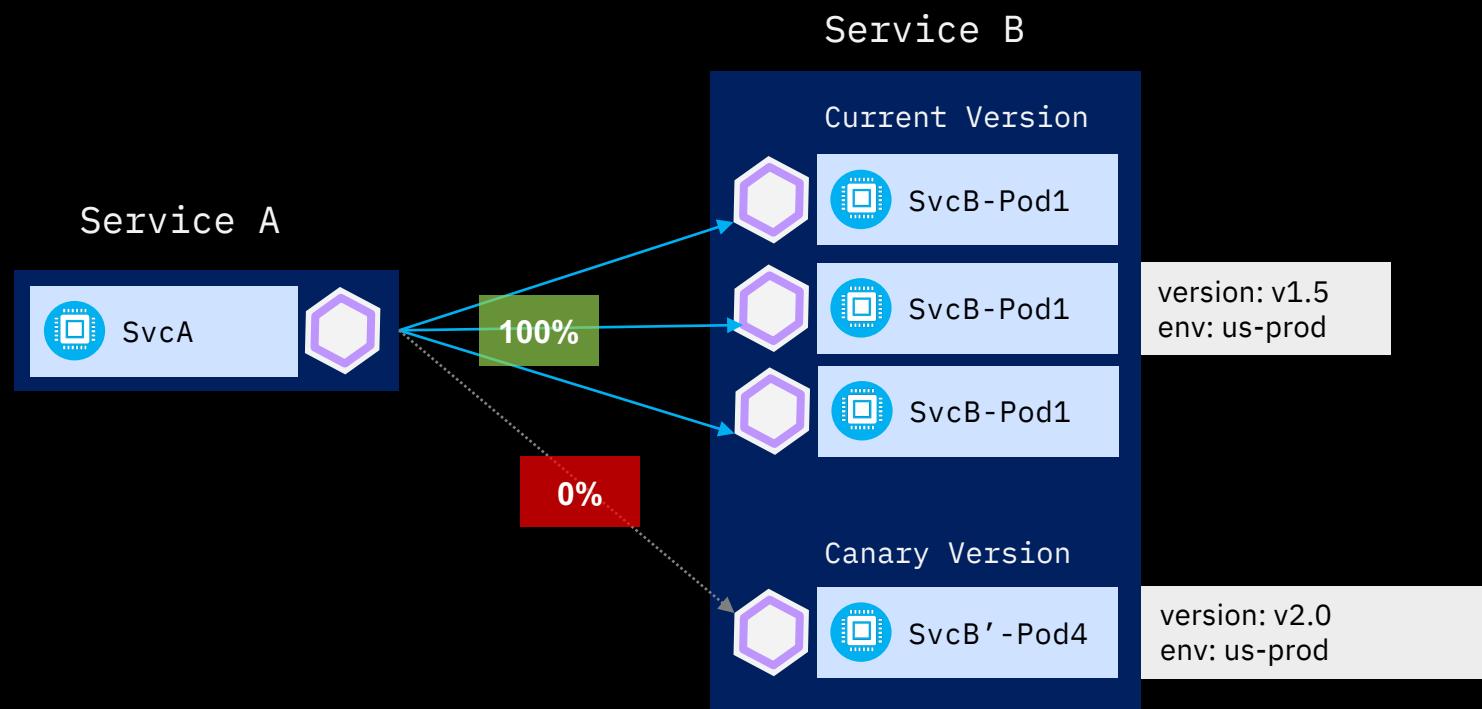
**SERVICE**  
helloworld  
selector  
app: helloworld

# Addressing DevOps Challenges



#	CHALLENGE	ISTIO SOLUTION
<b>CHALLENGE 1</b>	ROLL OUT NEW VERSION WITHOUT DOWNTIME OR CHANGING CODE	TRAFFIC CONTROL
<b>CHALLENGE 2</b>	HOW TO DO CANARY TESTING	TRAFFIC SPLITTING
<b>CHALLENGE 3</b>	HOW TO DO A/B TESTING	TRAFFIC STEERING
<b>CHALLENGE 4</b>	THINGS DON'T ALWAYS GO CORRECTLY IN PRODUCTION...	TRAFFIC MIRRORING RESILIENCY RESILIENCY TESTING
<b>CHALLENGE 5</b>	HOW CAN I LIMIT RATE FOR SOME OF MY SERVICES?	RATE LIMITING
<b>CHALLENGE 6</b>	I NEED TO VIEW AND MONITOR WHAT IS GOING ON WHEN CRISIS ARISES	TELEMETRY
<b>CHALLENGE 7</b>	HOW CAN I SECURE MY SERVICES?	AUTHENTICATION AUTHORIZATION CALICO

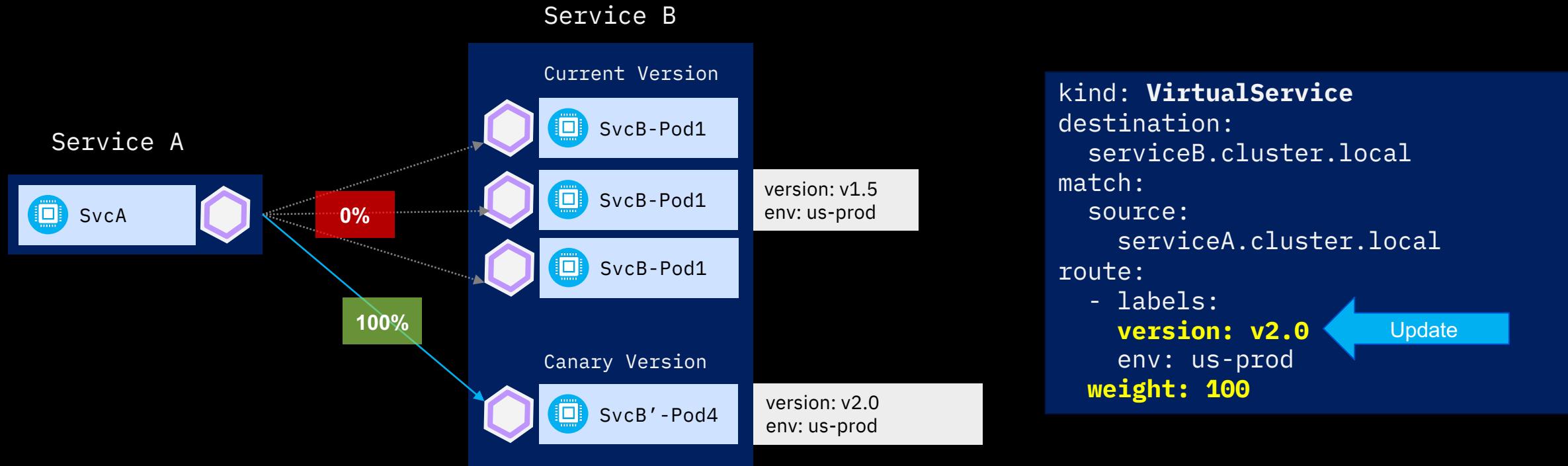
# Traffic Control



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

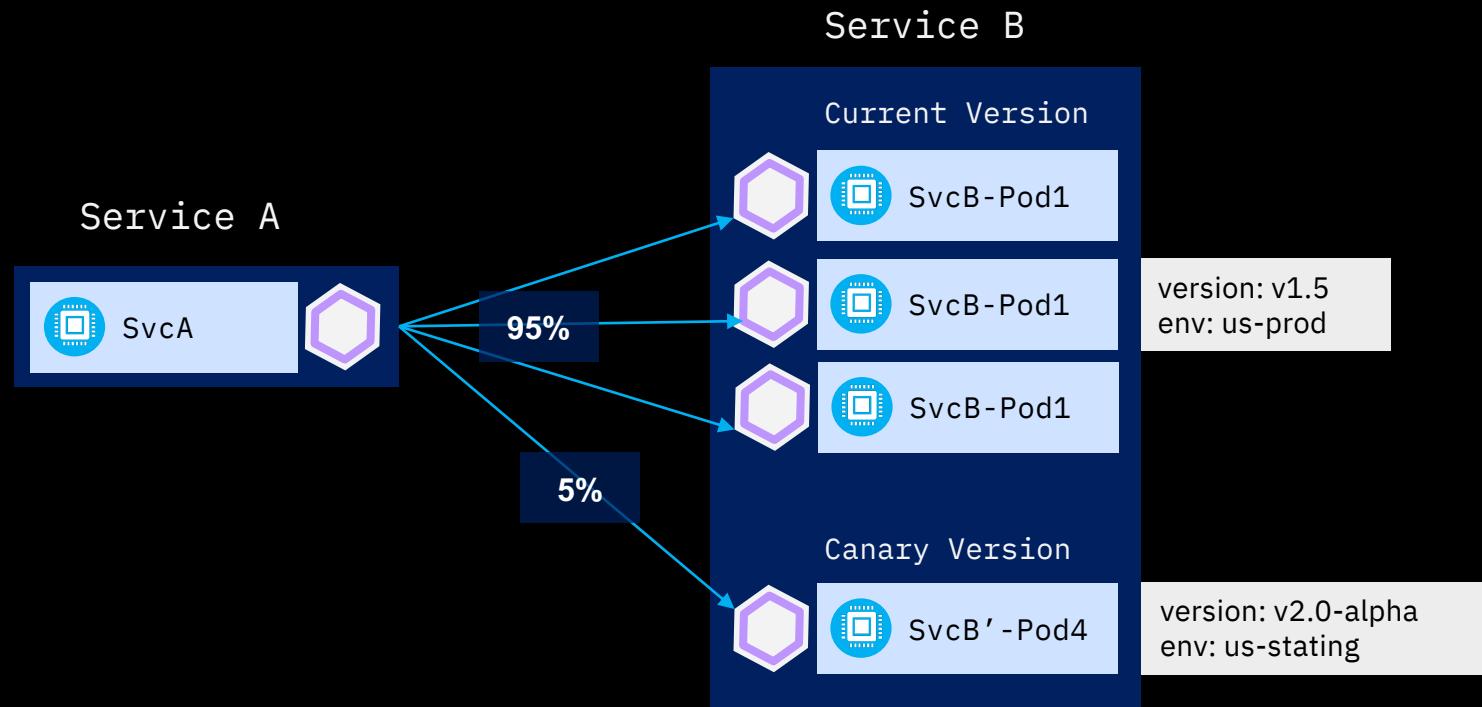
**CHALLENGE 1**  
**ROLL OUT NEW VERSION WITHOUT DOWNTIME OR CHANGING CODE**

# Traffic Control



**CHALLENGE 1**  
ROLL OUT NEW VERSION WITHOUT DOWNTIME OR CHANGING CODE

# Traffic Splitting

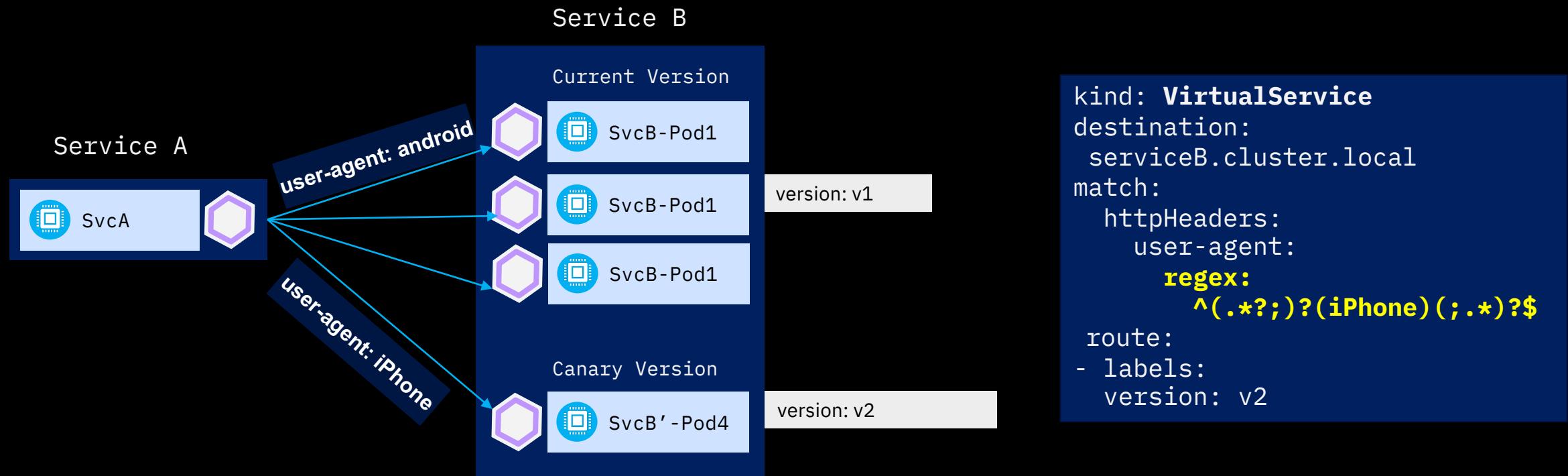


```
kind: VirtualService
destination:
  serviceB.cluster.local
match:
  source:
    serviceA.cluster.local
route:
  - labels:
      version: v1.5
      env: us-prod
  weight: 95
  - labels:
      version: v2.0-alpha
      env: us-staging
  weight: 5
```

## CHALLENGE 2 HOW TO DO CANARY TESTING

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

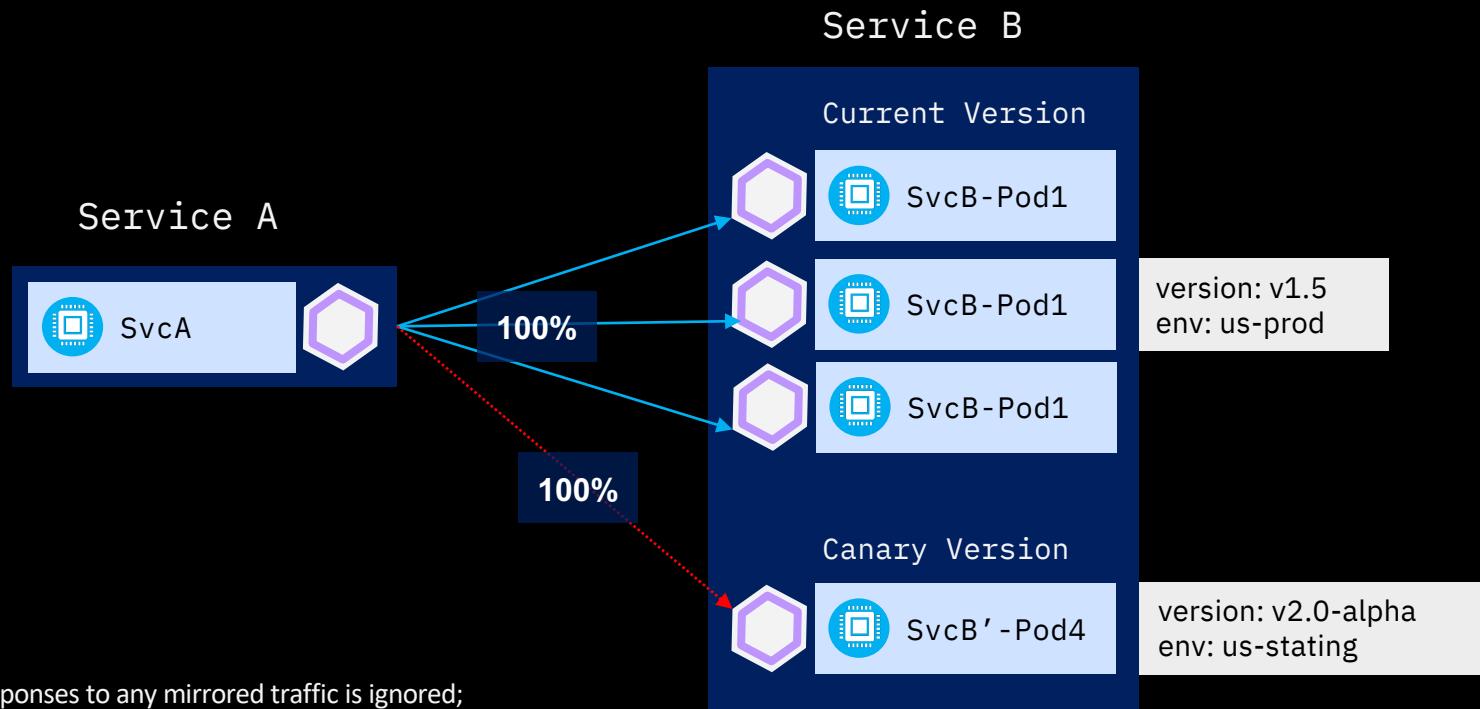
# Traffic Steering



Routing based on the request content

**CHALLENGE 3**  
**HOW TO DO A/B TESTING**

# Traffic Mirroring



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

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

# Resiliency

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

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

## Resilience features

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

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

# Rate limiting

Istio protects your application from rogue actors by imposing ratelimits

## Quotas:

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

## Rate limit

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

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

# Telemetry

- Monitoring & tracing should not be an afterthought in the infrastructure
- Goals
  - Metrics without instrumenting apps
  - Consistent metrics across fleet
  - Trace flow of requests across services
  - Portable across metric backend providers



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

# Kiali

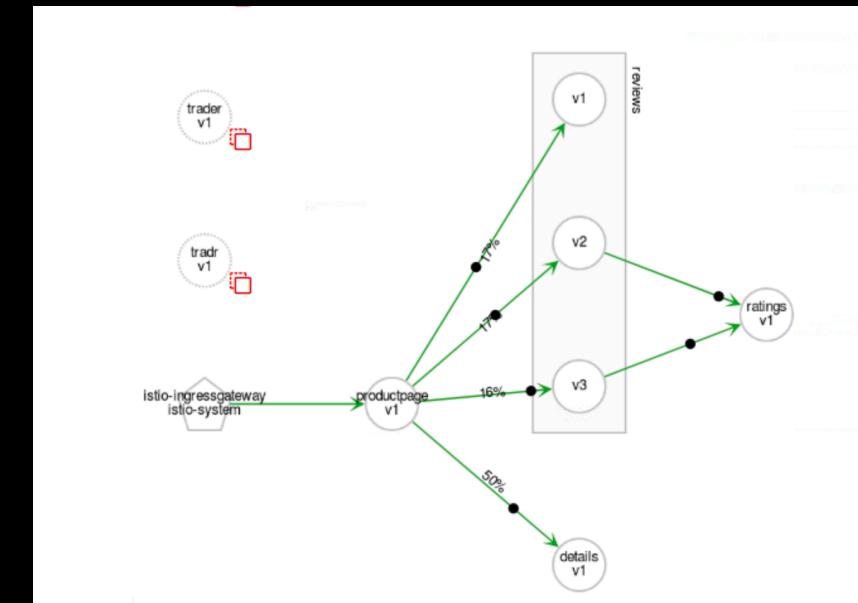
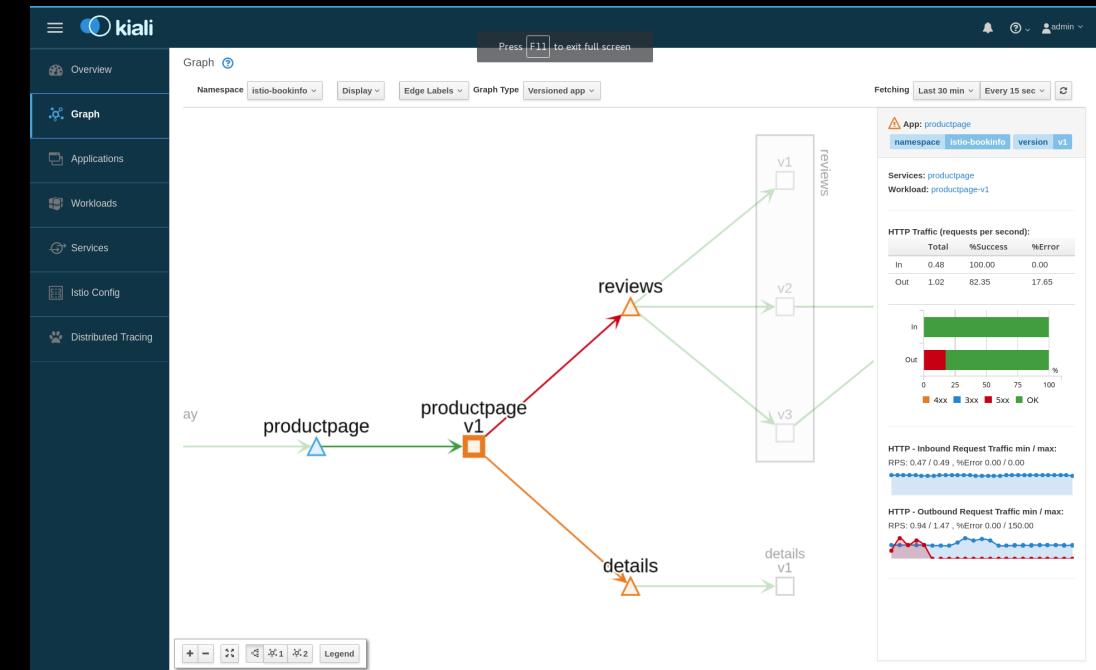
**Kiali** (greek κιάλι)  
*monocular or spyglass*

Visualise the service mesh topology, features like circuit breakers or request rates

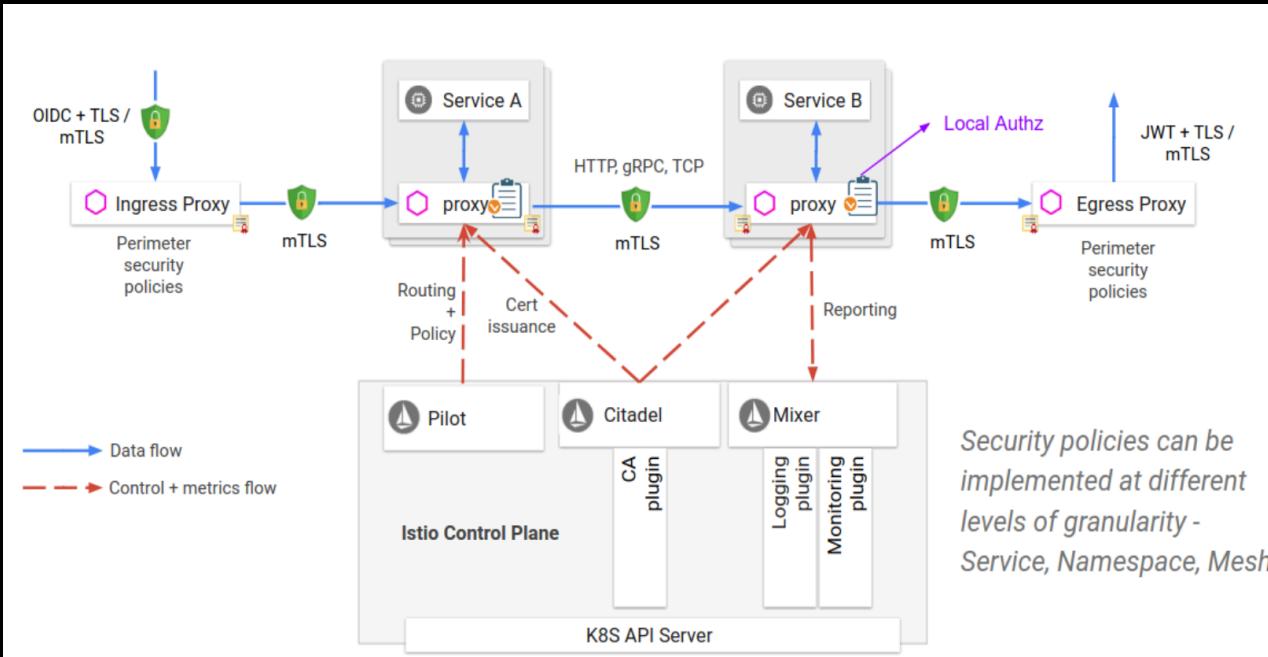
## Features

- **Graph**
  - Health
  - Types
  - Side Panel
  - Traffic Animation
- **Applications, Workloads and Services**
  - Detailed Metrics
- **Traffic Routing**
- **Istio compliance**
- **Istio Configuration**

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



# Security



- **Authentication**

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

- **Authorization**

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

**CHALLENGE 7**  
**HOW CAN I SECURE MY SERVICES?**

# Service Mesh - Bad Idea ?

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

- ▶ **Service Meshes are Opinionated**

They are a *platform* solution. “Work their way”

- ▶ **Service Meshes are Complex**

Adds considerable complexity with sidecars and control plane

- ▶ **Service Meshes can be Slow**

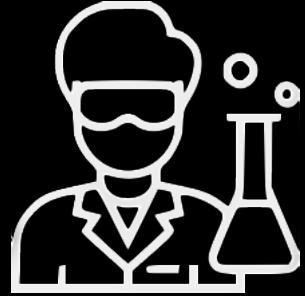
Routing traffic through a series of proxies can get painfully slow (about 700 nodes → reflector)

- ▶ **Service Meshes are for Developers**

Focused primarily on Developer view.

# QUESTIONS?





Sources and documentation will be available here:

Online Course

[https://niklaushirt.github.io/k8s\\_training\\_web/](https://niklaushirt.github.io/k8s_training_web/)

Course Documents

[https://github.com/niklaushirt/k8s\\_training\\_public](https://github.com/niklaushirt/k8s_training_public)

Course Files

<https://github.com/niklaushirt/training>



# Do you want to go further?

Online Course

[https://niklaushirt.github.io/k8s\\_training\\_web/](https://niklaushirt.github.io/k8s_training_web/)

Course Videos – Kubernetes: from zero to hero

<https://www.youtube.com/channel/UCIS0jmGOQrG2AKKPkJYj9w/videos>



# Niklaus Hirt



nikh@ch.ibm.com



@nhirt

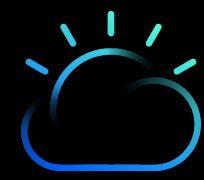


• Kubernetes Workshop Series

**Kubernetes - Hands-On**

07



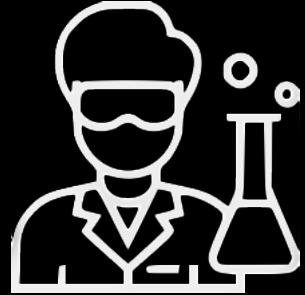


# Starting Course JTC02 Kubernetes

Select JTC02 course to begin



The screenshot shows a Firefox browser window with the title "Welcome to the Kubernetes Course | Kubernetes Training - Mozilla Firefox". The URL in the address bar is [https://niklaushirt.github.io/k8s\\_training\\_web/](https://niklaushirt.github.io/k8s_training_web/). The page content includes a sidebar with links like "Introduction", "JTC01 Docker", "JTC02 Kubernetes Labs" (which is highlighted in blue), "JTC10 Istio", "JTC14 Kubernetes Ansible Operators", "JTC16 Kubernetes Security Labs", and "JTC17 Kubernetes Advanced Security". The main content area features a large heading "Welcome to the Kubernetes Course" and a section titled "Welcome to my courses around Kubernetes." It lists resources such as "Videos" (<https://www.youtube.com/channel/UCIS0jmGOQrG2AKKPktJYj9w/>), "Training Documents" ([https://github.com/niklaushirt/k8s\\_training\\_public](https://github.com/niklaushirt/k8s_training_public)), and "Training files" (<https://github.com/niklaushirt/training.git>). A copyright notice "©2020 Niklaus Hirt" is at the bottom, and a blue button at the bottom right says "Introduction >".



## JTC02 Labs

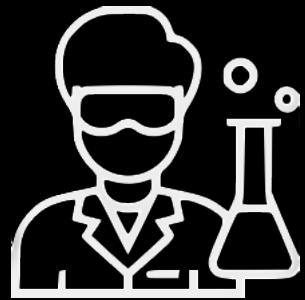
Lab 1: Refresher/overview over Kubernetes.

Lab 2: Running your first Pod on Kubernetes.

Lab 3: Deploy a Web Application

Lab 4: Scale an application on Kubernetes

Lab 5: Basics of persisting data in Kubernetes

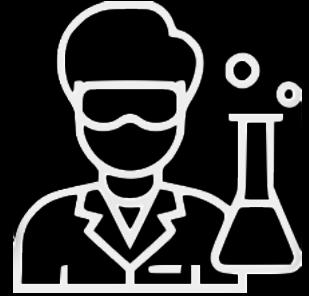


READY  
SET  
GO!!!!

Duration: 60 mins

# QUESTIONS?





Sources and documentation will be available here:

Online Course

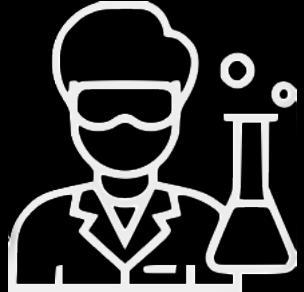
[https://niklaushirt.github.io/k8s\\_training\\_web/](https://niklaushirt.github.io/k8s_training_web/)

Course Documents

[https://github.com/niklaushirt/k8s\\_training\\_public](https://github.com/niklaushirt/k8s_training_public)

Course Files

<https://github.com/niklaushirt/training>



# Do you want to go further?

Online Course

[https://niklaushirt.github.io/k8s\\_training\\_web/](https://niklaushirt.github.io/k8s_training_web/)

Course Videos – Kubernetes: from zero to hero

<https://www.youtube.com/channel/UCIS0jmGOQrG2AKKPkJYj9w/videos>



# Niklaus Hirt



nikh@ch.ibm.com



@nhirt

# ANNEX



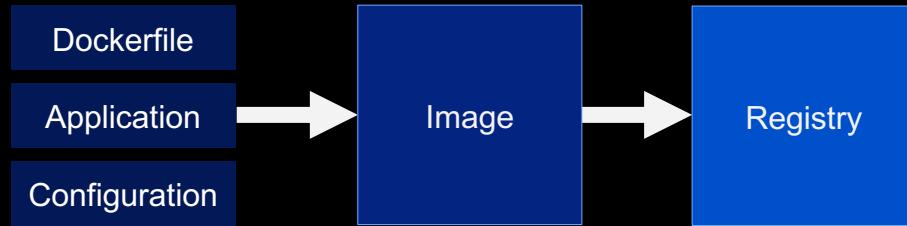
◦◦ Kubernetes Workshop Series  
**Kubernetes - Applied**

08



..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 . /app

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

EXPOSE 3000

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

Minimum components you need:

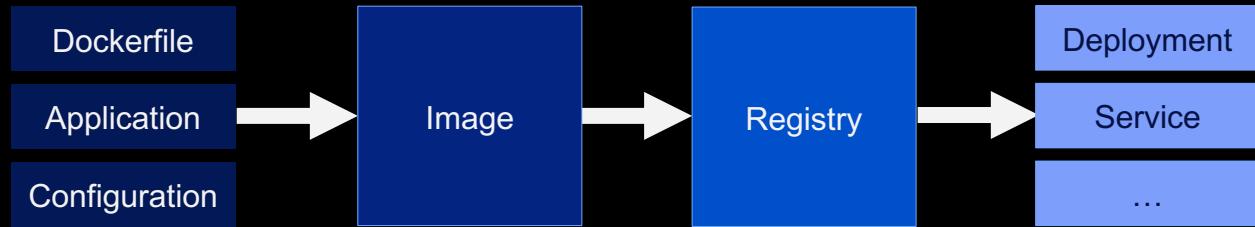
- Built Application or source  
(NodeJS, Java, Golang, ...)
- Dockerfile  
→ Use the right base image for your app

```

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



# Minimal Application – Kubernetes Manifests



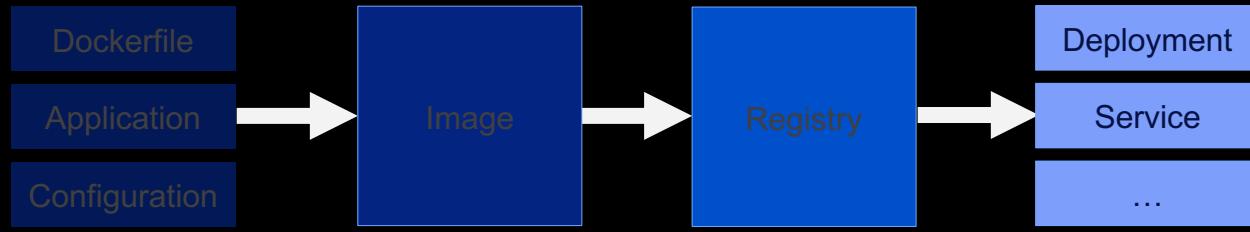
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

Minimum components you need:

- Deployment
- Service



# Minimal Application – Kubernetes Manifests

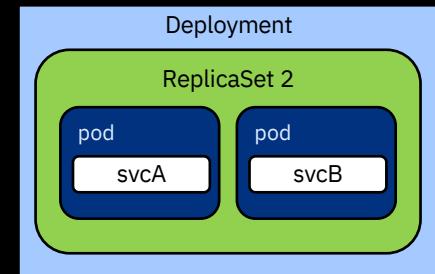


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

Deployment

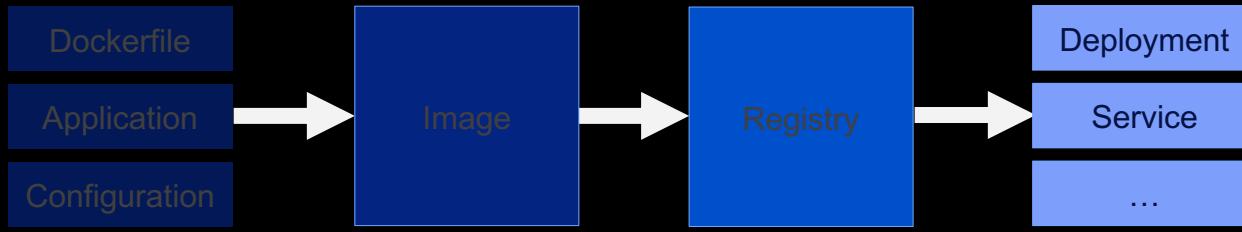
Minimum components you need:

- Deployment
- Service





# Minimal Application – Kubernetes Manifests



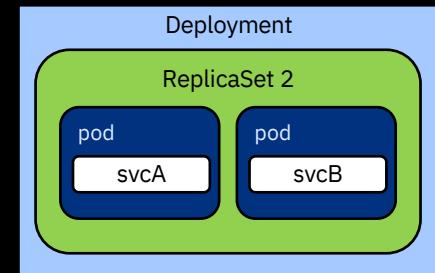
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

Deployment

ReplicaSet

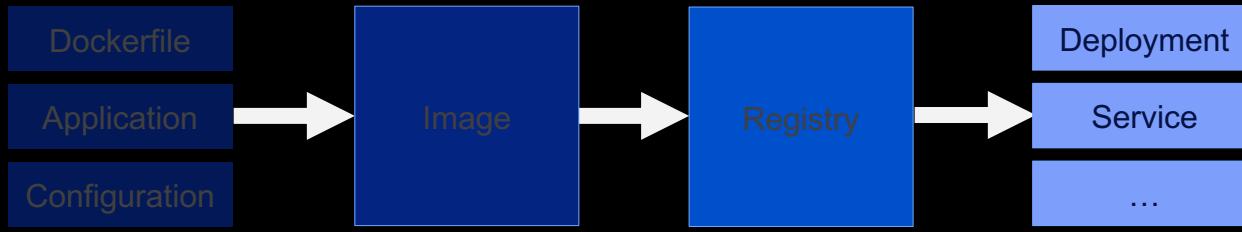
Minimum components you need:

- Deployment
- Service





# Minimal Application – Kubernetes Manifests



```
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
          imagePullPolicy: Always
          image: niklaushirt/k8s-demo:1.0
          ports:
            - containerPort: 80
```

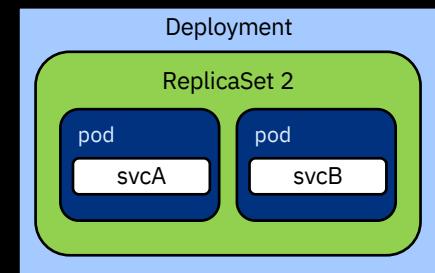
Deployment

ReplicaSet

Pod

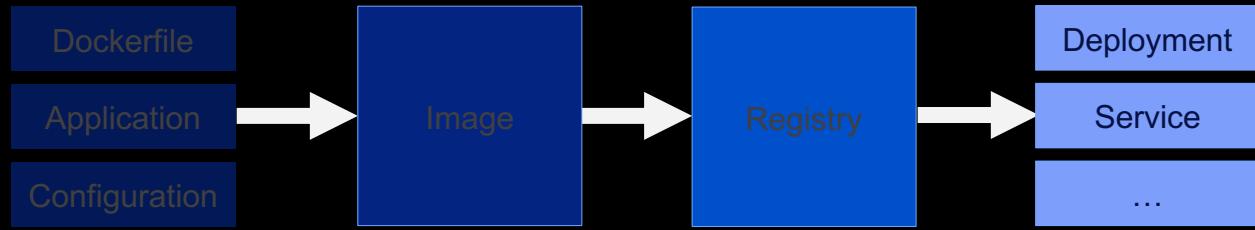
Minimum components you need:

- Deployment
- Service





# Minimal Application – Kubernetes Manifests



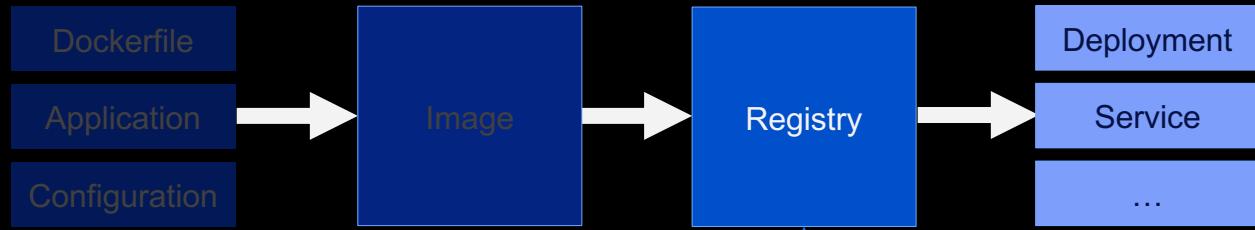
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
template:
  metadata:
    labels:
      app: k8s-demo
spec:
  containers:
  - name: k8s-demo
    imagePullPolicy: Always
   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

Minimum components you need:

- Deployment
- Service

Selectors must use the label of the deployment!!

# Minimal Application – Kubernetes Manifests



```
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
          imagePullPolicy: Always
          image: niklaushirt/k8s-demo:1.0
      ports:
        - containerPort: 80
```

Minimum components you need:

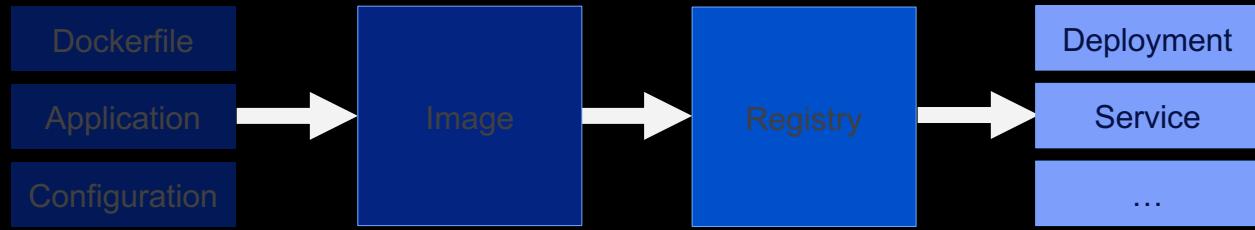
- Deployment
- Service

## imagePullPolicy:

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



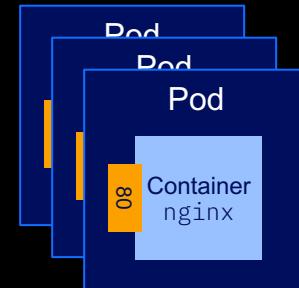
# Minimal Application – Kubernetes Manifests



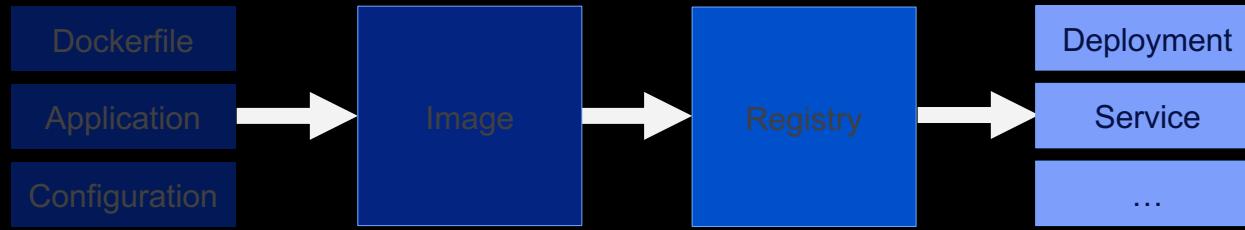
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

Minimum components you need:

- Deployment
- Service



# Minimal Application – Kubernetes Manifests

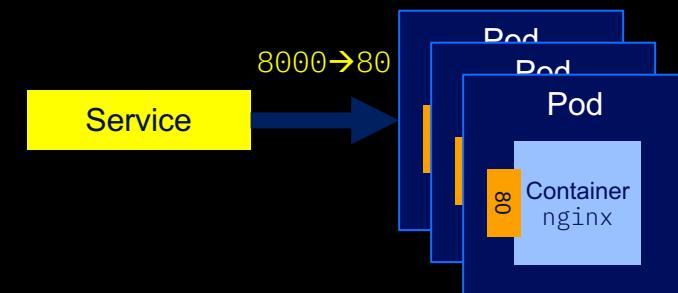


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

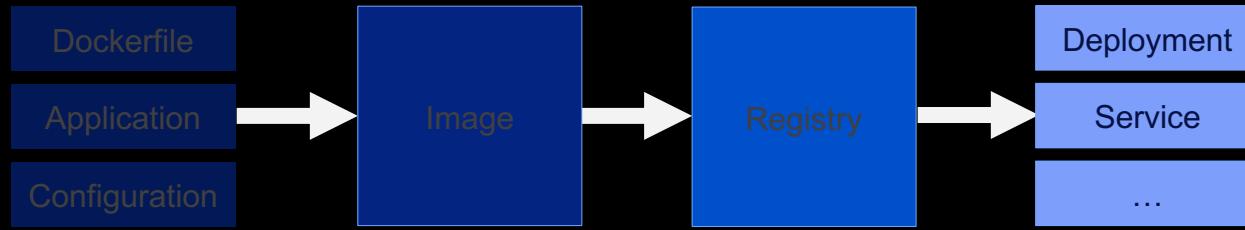
```
apiVersion: v1
kind: Service
metadata:
  name: k8s-demo-service
spec:
  ports:
    - port: 8000
      targetPort: 80
      protocol: TCP
    selector:
      app: k8s-demo
```

Minimum components you need:

- Deployment
- Service



# Minimal Application – Kubernetes Manifests

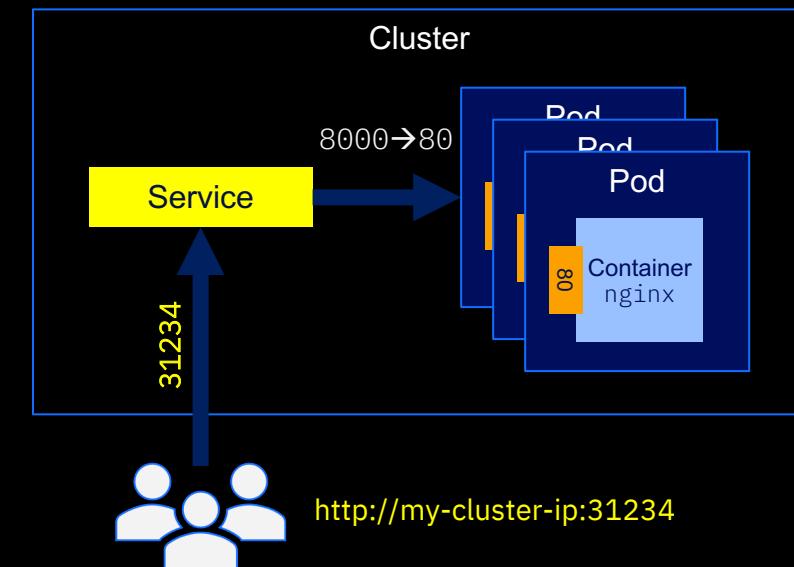


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

```
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
```

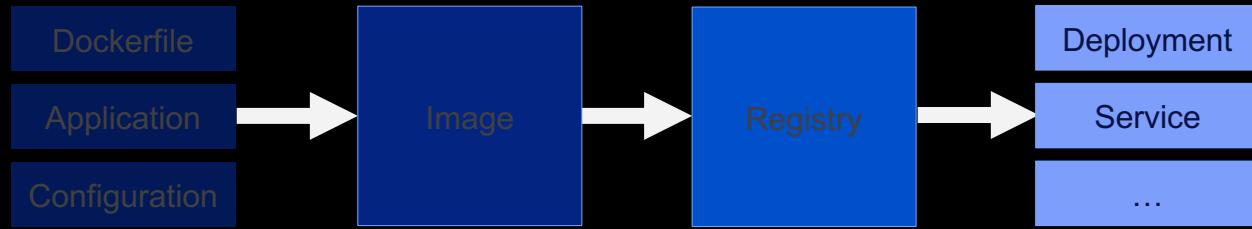
Minimum components you need:

- Deployment
- Service





# Minimal Application – Kubernetes Manifests



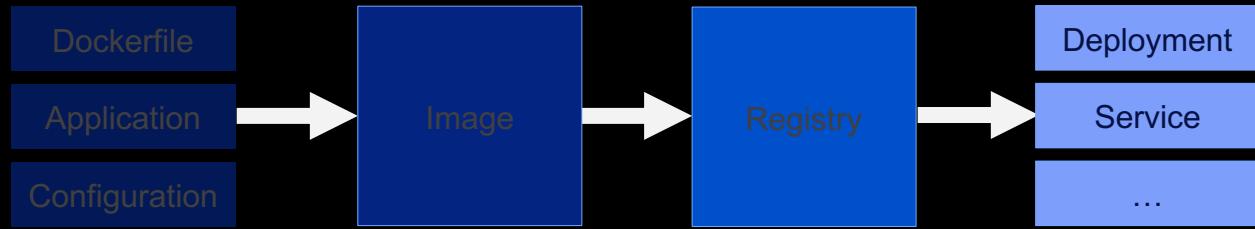
```
apiVersion: apps/v1
kind: Deployment
metadata:
...
spec:
...
template:
...
resources:
  requests:
    memory: 60Mi
    cpu: 10m
  limits:
    memory: 500Mi
    cpu: 100m
```

Minimum components you need:

- Deployment
- Service



# Minimal Application – Kubernetes Manifests

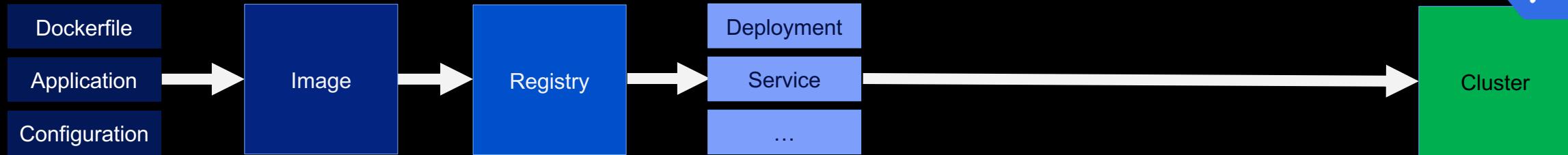


```
apiVersion: apps/v1
kind: Deployment
metadata:
...
spec:
...
template:
...
env:
- name: PORT
  value : "3000"
- name: APPLICATION_NAME
  value: k8sdemo
- name: BACKEND_URL
  value: http
```

Minimum components you need:

- Deployment
- Service

# Minimal Application – Kubernetes Manifests



```
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
```

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

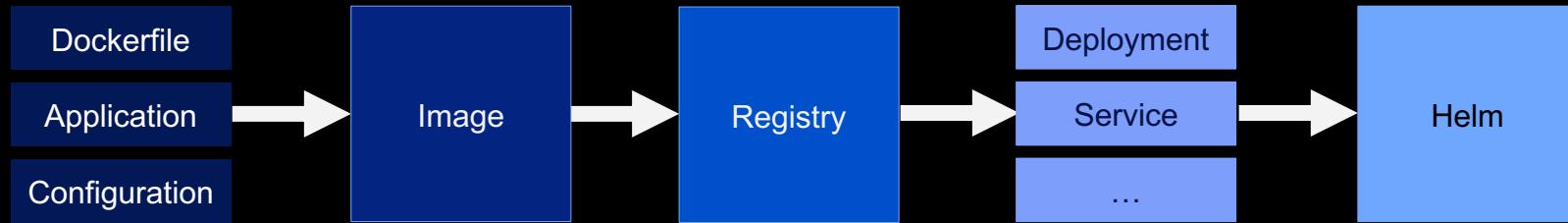
Minimum components you need:

- Deployment
- Service

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

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

# Minimal Application – Helm



```
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

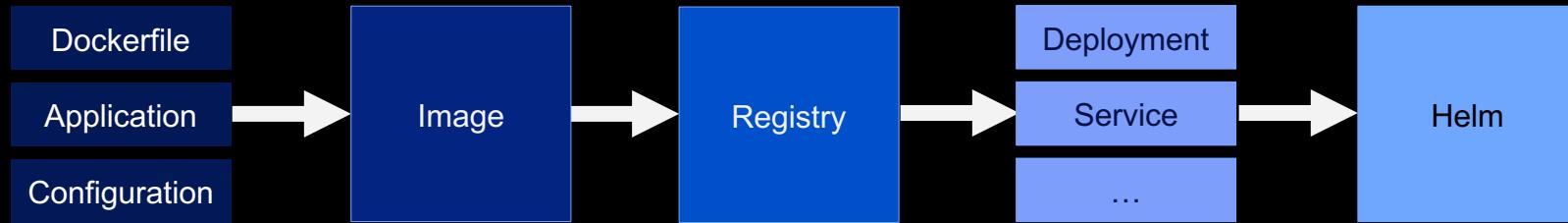
Templates (yaml)

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



```
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



```
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

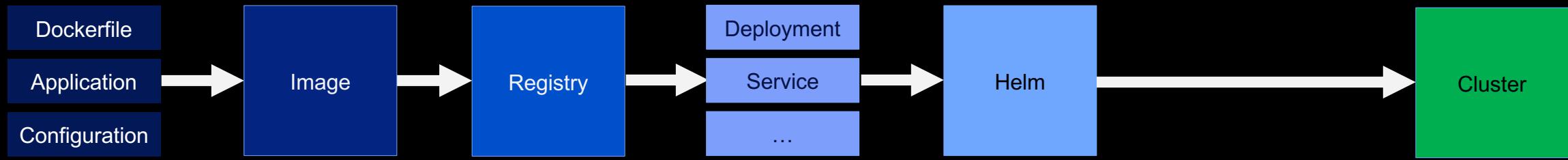
Templates (yaml)

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



```

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
  
```

```

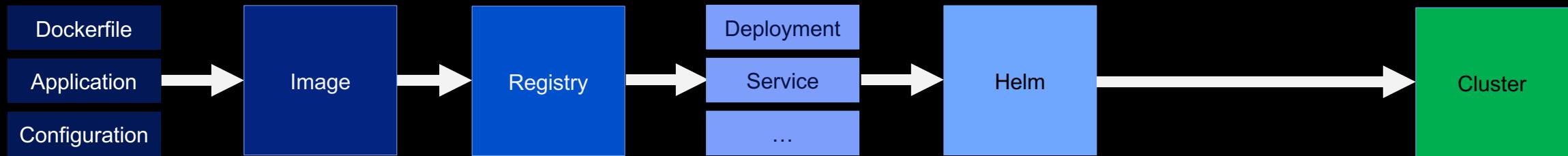
apiVersion: v1
image:
  name:
    - niklaushirt/k8s-demo
  tag: 1.0
  pullPolicy: IfNotPresent
replicas: 1
  
```

helm install k8s-demo

```

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



```
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
```

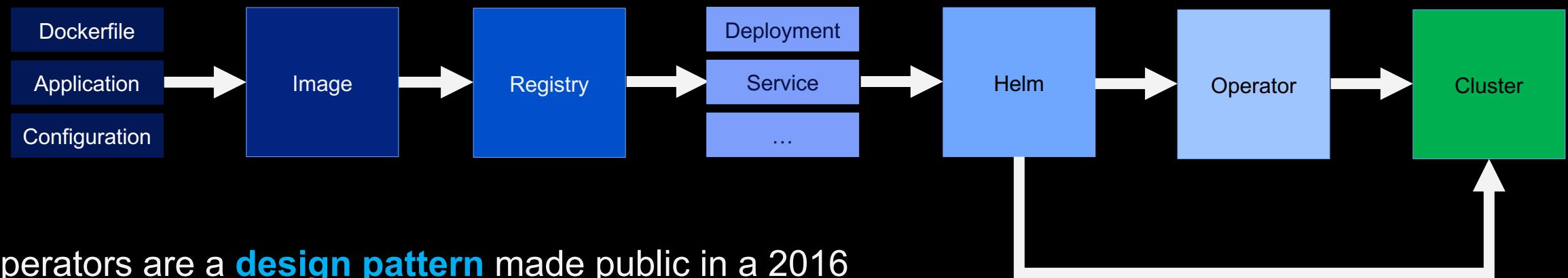
```
apiVersion: v1
image:
  name:
    - niklaushirt/k8s-demo
  tag: 1.0
  pullPolicy: IfNotPresent
replicas: 1
```

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

```
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.

More about this in three weeks JTC14 – Kubernetes Operators

# K9s – the Norton Commander of Kubernetes



```
Context: services/c107-e-us-south-containers-cloud-ibm-com:3_ <0> all      <ctrl-d> Delete      <y> YAML
Cluster: c107-e-us-south-containers-cloud-ibm-com:30958   <1> services    <d> Describe
User: IAMnikh@ch.ibm.com/c107-e-us-south-containers-clou... <2> default    <e> Edit
K9s Rev: 0.14.0                                         <ctrl-k> Kill
K8s Rev: v1.16.2                                         <1> Logs      <shift-1> Logs Previous
CPU: 24%                                                 <s> Shell
MEM: 55%                                                 Pod(default) [8]

NAME          READY   STATUS    RS   CPU    MEM   %CPU/R   %MEM/R   %CPU/L   %MEM/L IP           NODE   QOS   AGE
dashboard-78dcdbfd49-wp2r2   1/1   Terminated   0   0     21     0       21       0       0 172.30.153.228 10.94.80.44 BU   6m58s
dashboard-78dcdbfd49-wrwzz  1/1   Running    0   n/a   n/a   n/a   n/a   n/a   n/a 172.30.211.203 10.94.80.49 BU   7s
kubetoy-deployment-866fc8c687-gv9vl  1/1   Running   23   0     16     0       25       0       0 12 172.30.212.45 10.94.80.49 BU   35d
nginx-policy-deployment-9dd74c6bf-ggnfm  1/1   Running   0   0     1     0       0       0       0 0 172.30.212.60 10.94.80.49 BE   5d14h
nginx-policy-deployment-9dd74c6bf-t6nrs  1/1   Running   0   0     1     0       0       0       0 0 172.30.31.248 10.94.80.27 BE   5d14h
openldap-96d6c5f88-fw7lm   1/1   Running   0   0     19     0       0       0       0 0 172.30.153.207 10.94.80.44 BE   43d
openldap-admin-78cfcbb5d5-bf68p  1/1   Running   0   1     83     0       0       0       0 0 172.30.153.205 10.94.80.44 BE   43d
student-ui-68c59b4cd4-vcsd5  1/1   Running   0   0     21     0       21       0       0 0 172.30.211.199 10.94.80.49 BU   27d

<pod>
```

Preinstalled in the training VM

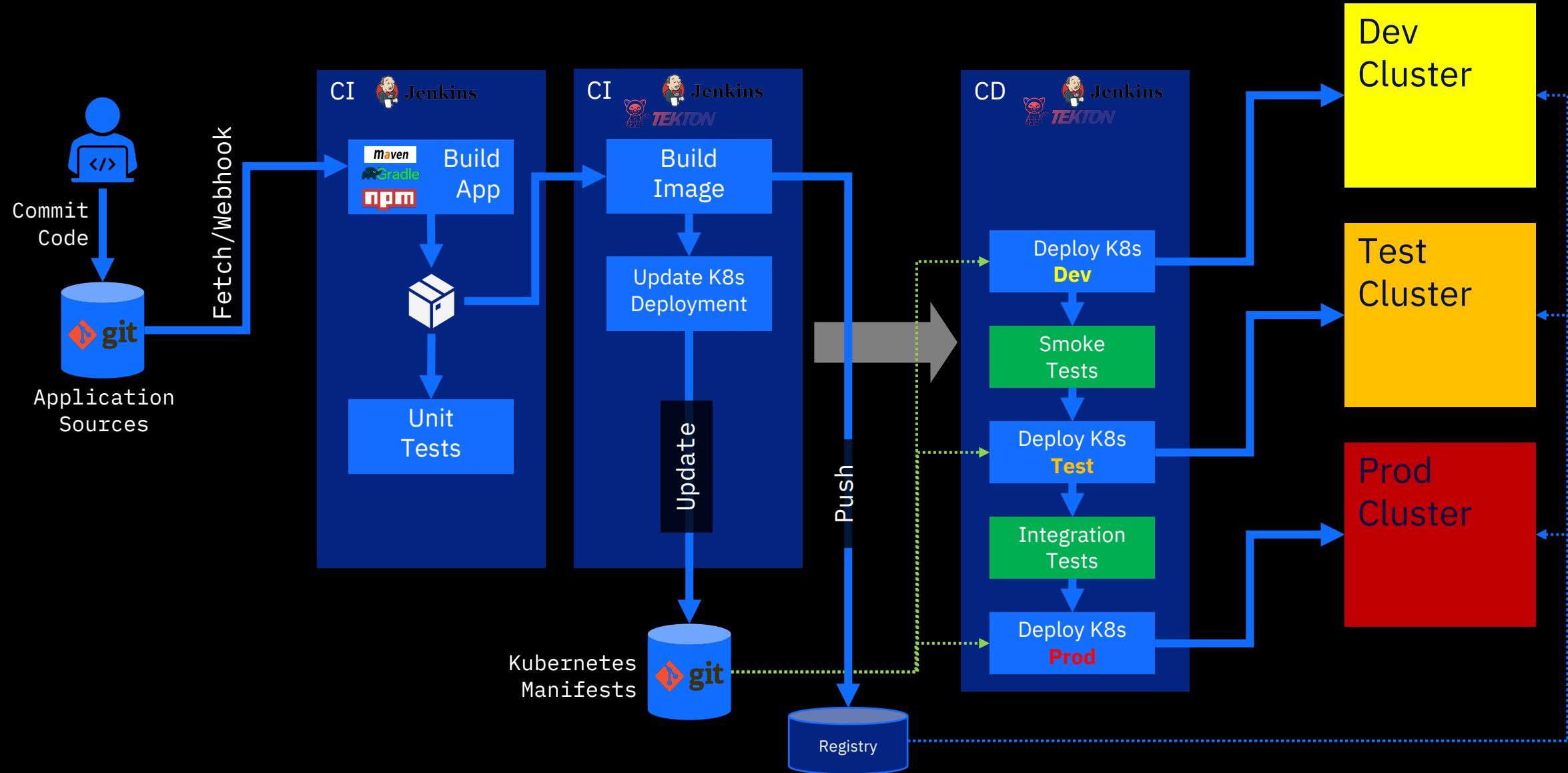
For dumb people like me  
(who just don't get vi)

`KUBE_EDITOR="nano" k9s`

I did create an alias in .bashrc/.zshrc  
`alias k9s='KUBE_EDITOR="nano" k9s'`

<https://github.com/derailed/k9s>

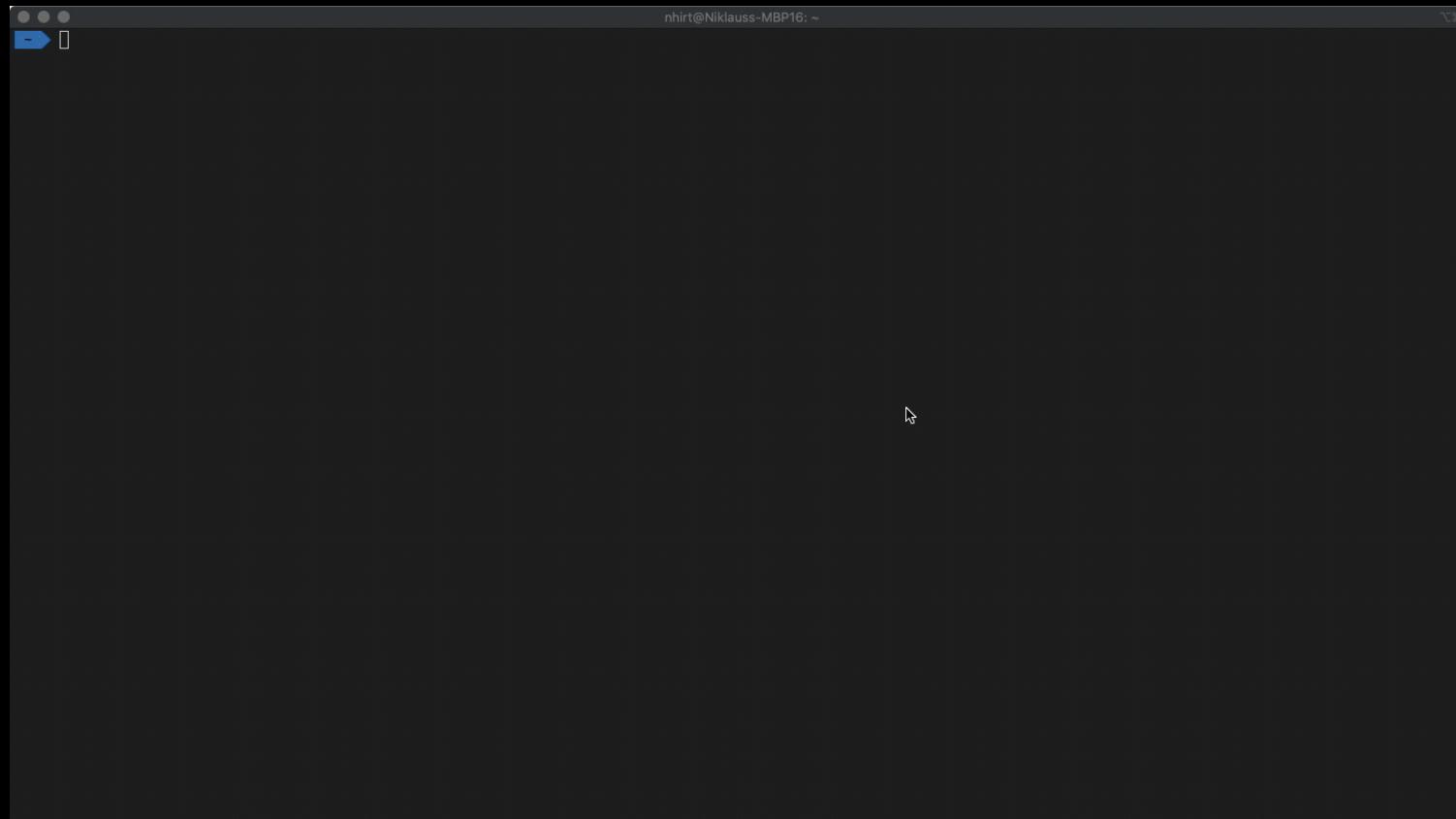
# Kubernetes – CI/CD Pipeline - GitOps



# K9s – the Norton Commander of Kubernetes



```
nhirt@Niklauss-MBP16: ~
```



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?



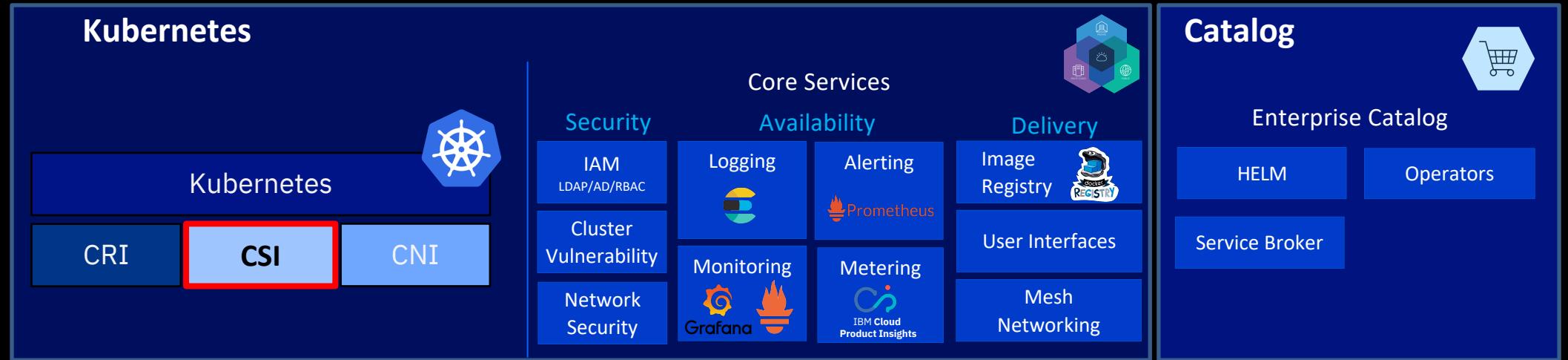
# Kubernetes Workshop Series

## **Container State**

08



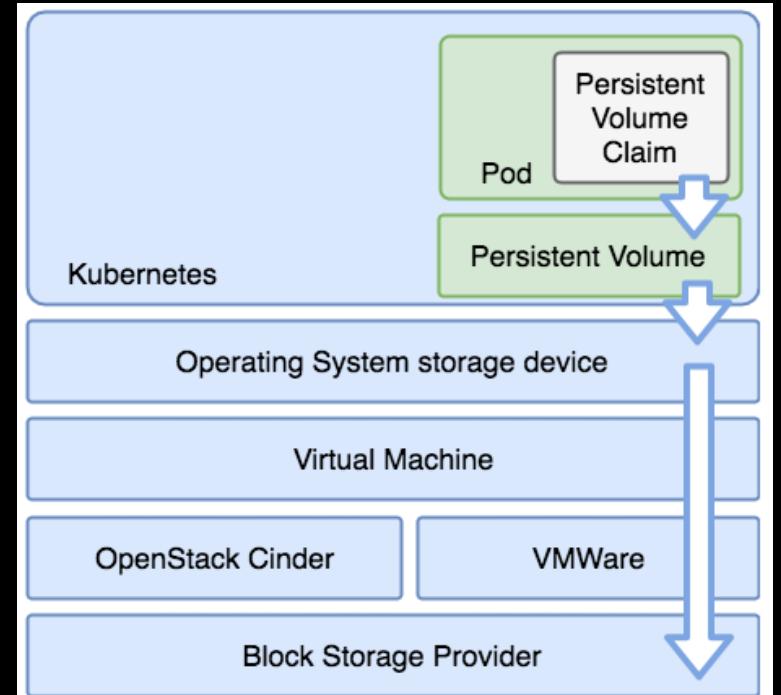
# Kubernetes – Storage



**Kubernetes Persistent Storage Support:**  
HostPath, NFS, GlusterFS, vSphereVolume, ....

## 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



The **PersistentVolume** subsystem provides an API for users and administrators that abstracts details of how storage is provided from how it is consumed.

**PersistentVolume** (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes

**PersistentVolumeClaim** (PVC) is a request for storage by a user. It is similar to a Pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific levels of resources (CPU and Memory). Claims can request specific size and access modes (e.g., they can be mounted once read/write or many times read-only).

## Provisioning

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.

### Binding

A user creates a `PersistentVolumeClaim` with a specific amount of storage requested and with certain access modes.

The cluster watches for new PVCs, tries to find a matching PV and binds them together.

Once bound, `PersistentVolumeClaim` binds are exclusive, regardless of how they were bound.

### Using

Pods use claims as volumes.

### Deleting

PersistentVolume (PVs) that are bound to PVCs are/can not removed from the system.

If a user deletes a PVC in active use by a Pod, the PVC is not removed until all Pods that are using the PVC are being terminated.

### Reclaiming

The reclaim policy for a PersistentVolume tells the cluster what to do with the volume after it has been released of its claim.

#### Retain

When the PersistentVolumeClaim is deleted, the PersistentVolume still exists and the volume is considered “released”.

An administrator can manually reclaim the volume.

#### Delete

Removes both the PersistentVolume object from Kubernetes, as well as the associated storage asset in the external infrastructure

#### Recycle (deprecated)

Performs a basic scrub (`rm -rf /thevolume/*`) on the volume and makes it available again for a new claim.

## 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, ...

### VolumeBindingMode

Volume binding and dynamic provisioning occurs once the PersistentVolumeClaim is created.

This may result in unschedulable Pods

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fast
provisioner: kubernetes.io/vsphere-volume
parameters:
  diskformat: zeroedthick
  datastore: VSANDatastore
reclaimPolicy: Retain (or Delete, Reclaim)
volumeBindingMode: Immediate (or WaitForFirstConsumer)
```

## Kubernetes – Common Storage Interface

- GlusterFS
- Rook/Ceph
- OpenEBS
- NFS



## GlusterFS

**Gluster** is a scalable, distributed network filesystem. Using common off-the-shelf hardware, you can create large, distributed storage solutions for media streaming, data analysis, and other data- and bandwidth-intensive tasks. Gluster is free.

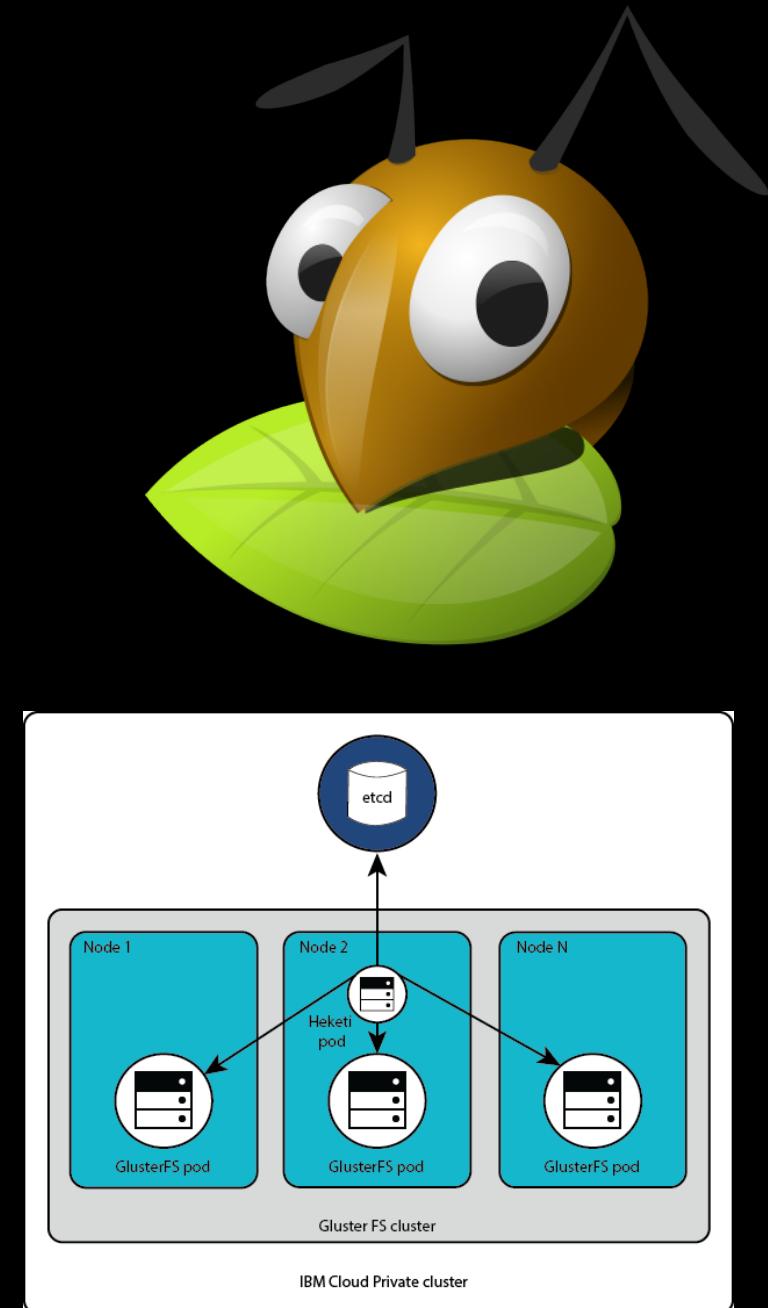
### Physical install

- Format and mount the bricks
- Installing GlusterFS
- Configure the trusted pool
- Set up a GlusterFS volume
- Install Heketi & Topology (for dynamic provisioning)
- Create K8s Storage Class

### Or container based

- [https://hub.docker.com/r/glusterfs/gluster\\_centos](https://hub.docker.com/r/glusterfs/gluster_centos)
- Must run as privileged → OpenShift?

<https://docs.gluster.org/en/latest/Quick-Start-Guide/Quickstart/>



## Rook/Ceph

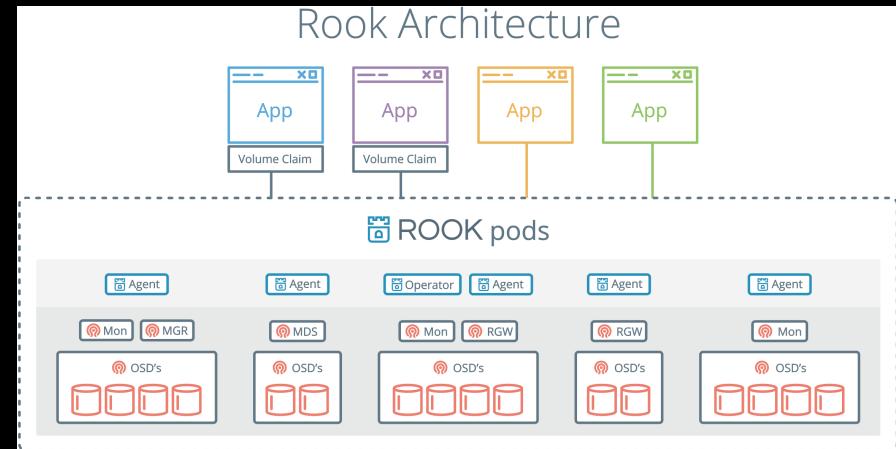
**Rook** turns distributed storage systems into self-managing, self-scaling, self-healing storage services.

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

### Install on K8s

- git clone <https://github.com/rook/rook.git>
- kubectl create -f common.yaml
- kubectl create -f operator(-openshift).yaml
- kubectl create -f cluster-test.yaml
- kubectl create -f csi/rbd/storageclass-test.yaml

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



■ 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

## OpenEBS

**OpenEBS** is truly Kubernetes native.

It adopts Container Attached Storage (CAS) approach, where each workload is provided with a dedicated storage controller. It implements granular storage policies and isolation that enable users to optimize storage for each specific workload.

Install on K8s

- sudo apt-get install open-iscsi
- kubectl apply -f <https://openebs.github.io/charts/openebs-operator.yaml>
- kubectl create -f csi/rbd/storageclass-test.yaml
- kubectl apply -f openebs-storageclasses.yaml

<https://docs.openebs.io/docs/next/installation.html>



# QUESTIONS?

