

Kubernetes Workshop Series

JTC02 Kubernetes Basic Concepts

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



Welcome to the
Kubernetes
Workshop Series



Housekeeping



Meeting is being recorded to be shared on Social Media



Meeting Mute All: Unmute to speak



Breaks: every 60mins (interrupt me if I forget ;-)



Questions:

In Slack # (not in Webex!)

Addressed at the end of the Module

Additional questions: unmute to speak



We will monitor the Slack channel during the Labs

→ Feel free to answer other participants questions

Who am I?

Niklaus Hirt

Passionate about tech for over 35 years

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



✉ nikh@ch.ibm.com

🐦 @nhirt

Agenda - Kubernetes Basic Concepts

Module 0: Prepare the Labs

Module 1: Kubernetes

Module 2: Use Kubernetes in the cloud

Module 3: Kubernetes Applied

Module 4: Kubernetes Hands-On



Sources and documentation will be available here:

Recording from JTC01

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

https://github.com/niklaushirt/k8s_training_public

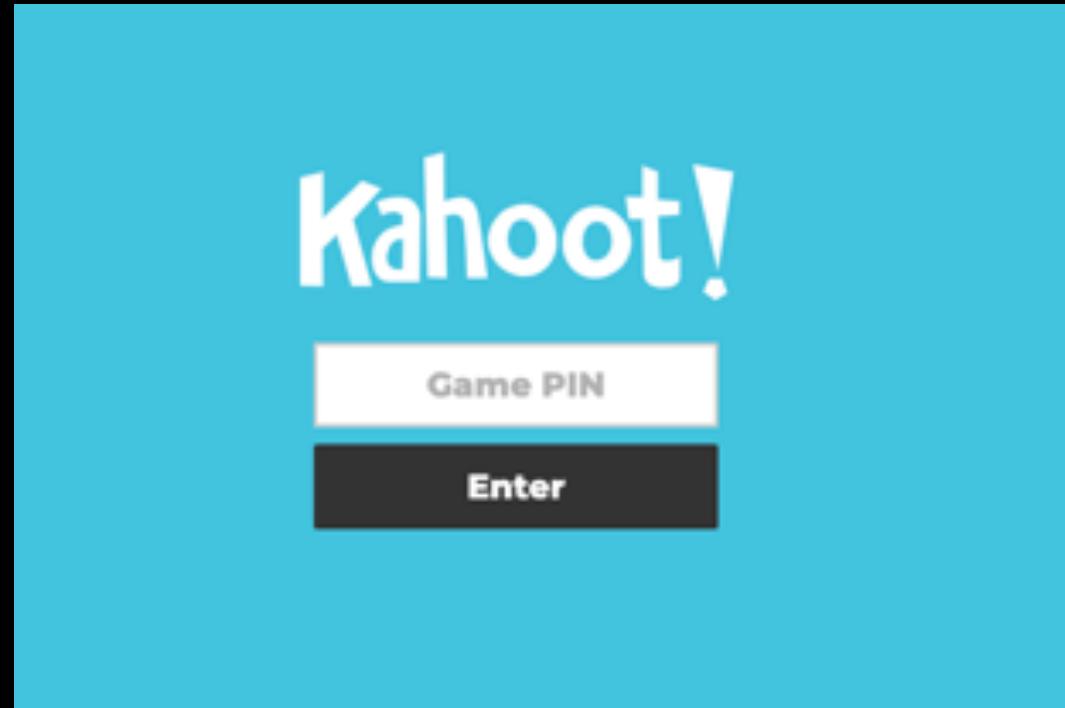
<https://github.com/niklaushirt/training>

Session Quiz & Feedback

We will collect some **feedback** and run a **quiz** or two.

Please make sure you can access <https://kahoot.it/>
either on your PC or Phone.

You will get the Game PIN
later in the training.



QUIZZ!!!

<https://kahoot.it/>





Kubernetes Workshop Series

Prepare the Labs

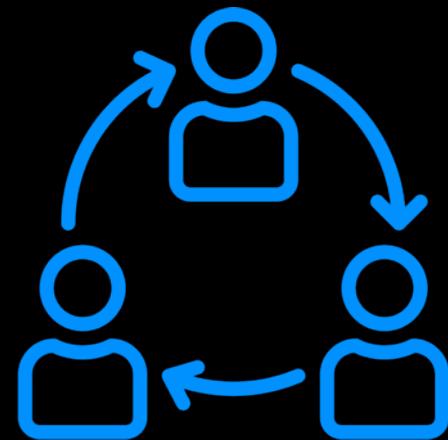


Session Objectives

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



Following some lectures will be ***hands-on*** work that each participant can to complete.





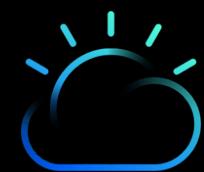
JTC90 Lab Setup

Task 1: Download Training VM

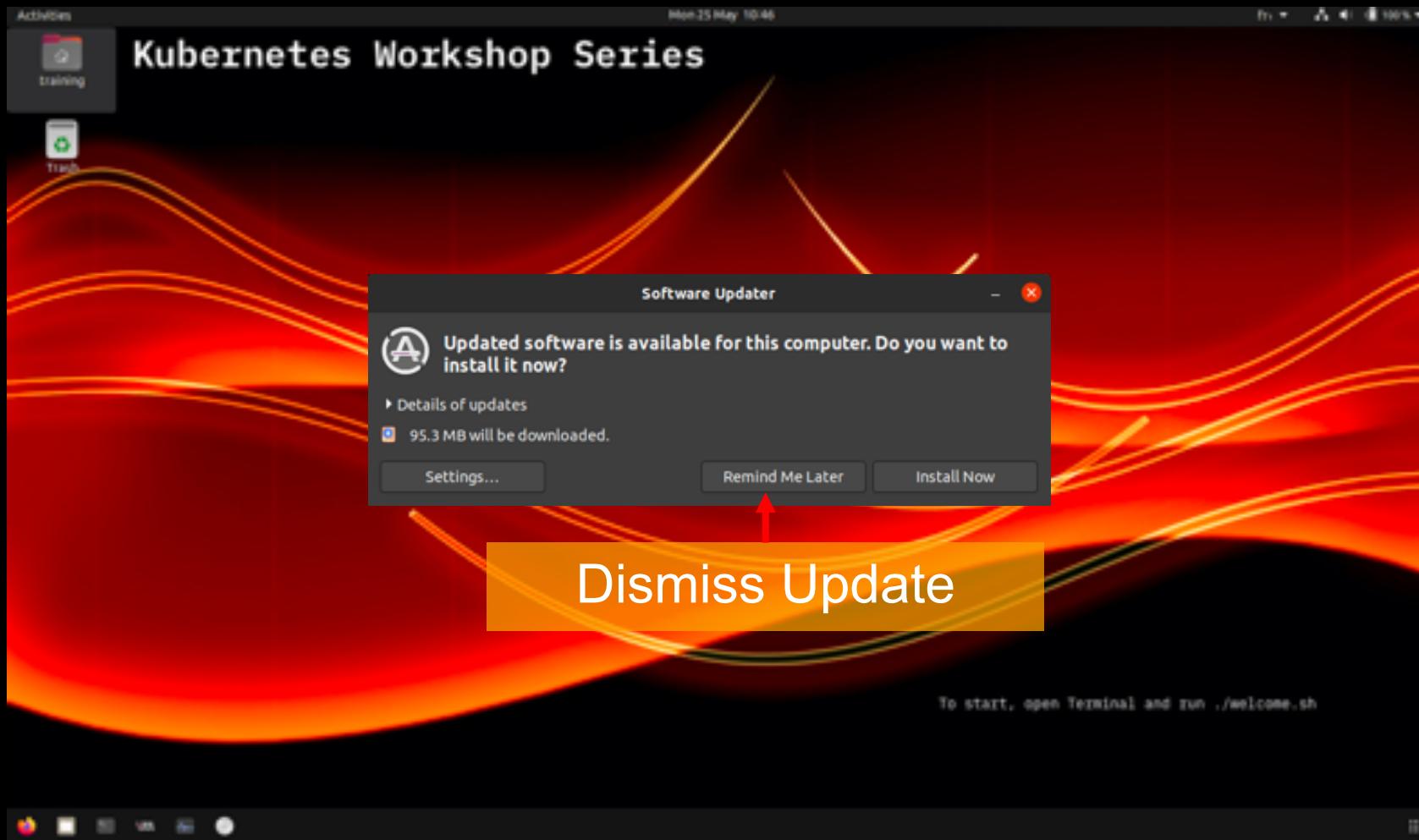
Task 2: Setup VMWare / VirtualBox

Task 3: Start Training VM

Task 4: Login / Check

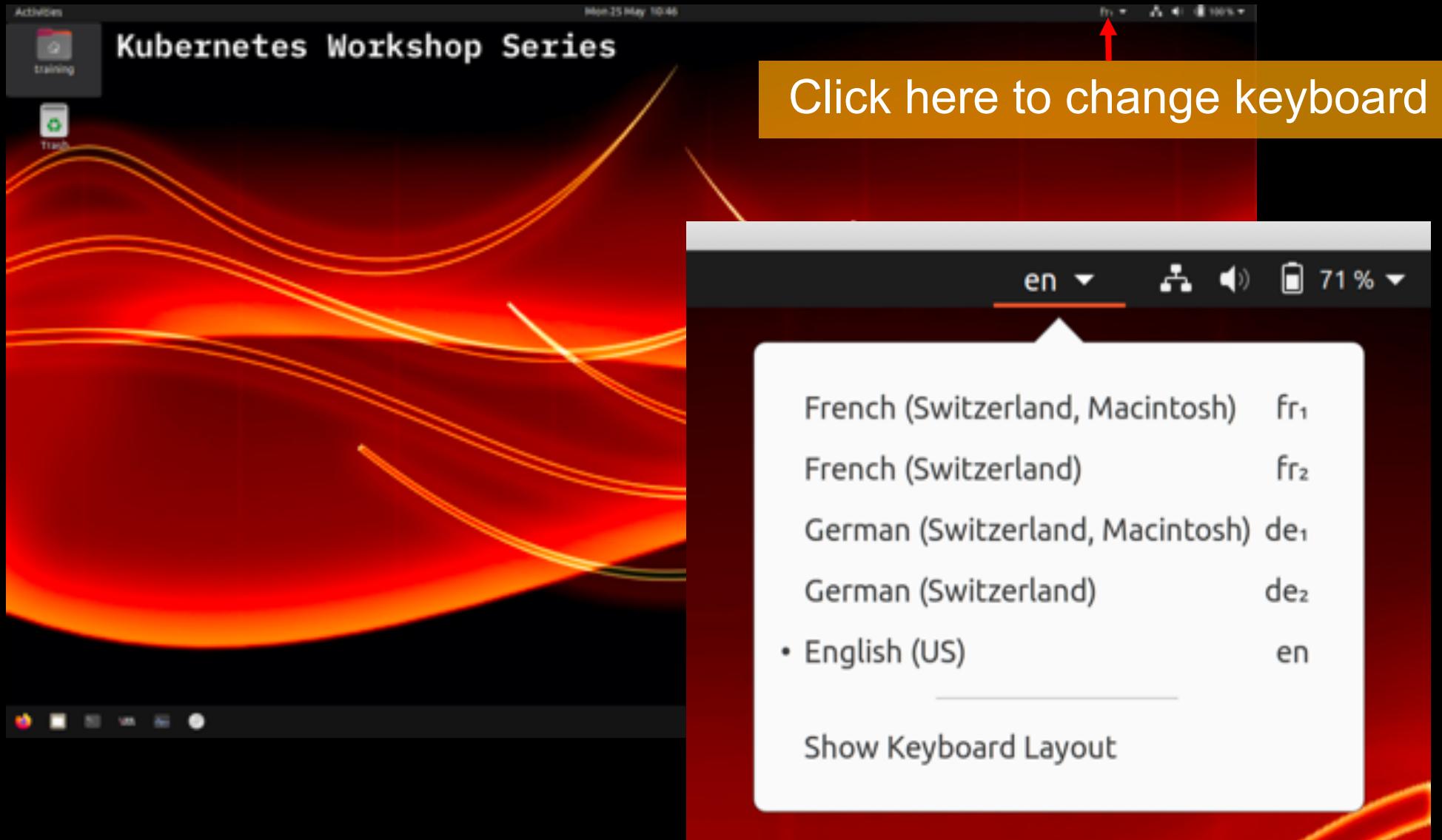


Accessing your Personal Training Environment





Accessing your Personal Training Environment





Accessing your Personal Training Environment



Start Terminal



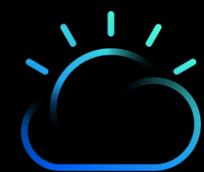
Accessing your Personal Training Environment

A screenshot of a terminal window titled "training@ubuntu: ~". The command "training@ubuntu:~\$./welcome.sh" is visible, with the "../welcome.sh" part highlighted by a red rectangle. A red arrow points from the text "Run ./welcome.sh" below the terminal to the highlighted command. The terminal has a dark background with light-colored text and icons.

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

Run ./welcome.sh

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



Accessing your Personal Training Environment

```
training@ubuntu:~  
nntent.com/cilium/cilium/v1.6/install/kubernetes/quick-install.yaml": deployments  
.apps "cilium-operator" already exists  
*****  
*****  
Startup done....  
*****  
*****  
Setting up your Personal Training Environment (PTE)  
-----  
The following steps will create your web-based Personal Training Environment  
t  
You will have to enter a name that will be used to show your progress in th  
e Instructor Dashboard  
in order to better assist you.  
*****  
*****  
Please enter your name  
Name:Niklaus Hirt
```

Enter your name



Name will be used to show your progress in the Instructor Dashboard in order to better assist you



Accessing 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 PTE Webpage just run `minikube service student-ui`
- Windows 10 problems can mostly be fixed by turning off Hyper-V by running (as admin)
`bcdedit /set hypervisorlaunchtype off`
and rebooting.
This disables Hyper-V and allows Virtualbox to support nested virtualisation.
- You can turn it back on again with
`bcdedit /set hypervisorlaunchtype auto`



Accessing your Personal Training Environment

Troubleshooting

I have added a standalone version to the Git repository for participants wishing to run the Labs directly on their PC.

This is **untested** and I cannot guarantee that all the Labs will be working 100%.

You must have the following setup on your PC:

- Minikube
- Docker
- Git

1. Clone the repository to your home directory

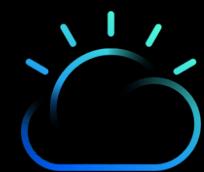
```
git clone https://github.com/niklaushirt/training.git
```

2. Go to the installation directory

```
cd ~/training/standalone
```

3. Run the preparation script

```
./welcome.sh
```



Accessing your Personal Training Environment

Troubleshooting

- Run **k9s** in the Terminal – wait for all the pods to be Running (blue – 1/1)

```
training@ubuntu: ~/training/standalone
Context: minikube
Cluster: minikube
User: minikube
K9s Rev: 0.19.4 [6601]
K8s Rev: v1.17.0
```

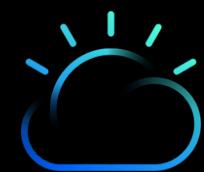
```
training@ubuntu: ~/training/standalone
Q - X
<0> all <0> Attach <shift-l> Logs P_
<1> kube-system <ctrl-d> Delete <shift-f> Port-F_
<2> default <d> Describe <s> Shell
<e> Edit <y> YAML
<ctrl-k> Kill
<l> Logs
```

Pods(all)[15]							
NAMESPACE	NAME	READY	RESTARTS	STATUS	IP	NODE	AGE
default	student-vt-945c5c77f-xp4rd	0/1	0	ContainerCreating	n/a	minikube	23m
kube-system	cilium-4jcob	1/1	6	Running	192.168.39.52	minikube	32d
kube-system	cilium-operator-78fcc89568-n9jbc	1/1	7	Running	192.168.39.52	minikube	32d
kube-system	coredns-6955765f44-75n9l	1/1	1	Running	10.88.0.54	minikube	27d
kube-system	coredns-6955765f44-q8rjs	1/1	1	Running	10.88.0.56	minikube	27d
kube-system	etcd-minikube	1/1	7	Running	192.168.39.52	minikube	32d
kube-system	kube-apiserver-minikube	1/1	7	Running	192.168.39.52	minikube	32d
kube-system	kube-controller-manager-minikube	1/1	9	Running	192.168.39.52	minikube	32d
kube-system	kube-proxy-lbxtz	1/1	7	Running	192.168.39.52	minikube	32d
kube-system	kube-registry-proxy-49v8d	1/1	6	Running	10.88.0.52	minikube	32d
kube-system	kube-registry-v0-ccsd5	1/1	6	Running	10.88.0.53	minikube	32d
kube-system	kube-scheduler-minikube	1/1	9	Running	192.168.39.52	minikube	32d
kube-system	storage-provisioner	0/1	7	Error	192.168.39.52	minikube	32d
kubernetes-dashboard	dashboard-metrics-scraper-7b64584c5c-95577	1/1	6	Running	10.88.0.57	minikube	32d
kubernetes-dashboard	kubernetes-dashboard-5b48b67b68-j49lv	0/1	7	CrashLoopBackOff	10.88.0.55	minikube	32d

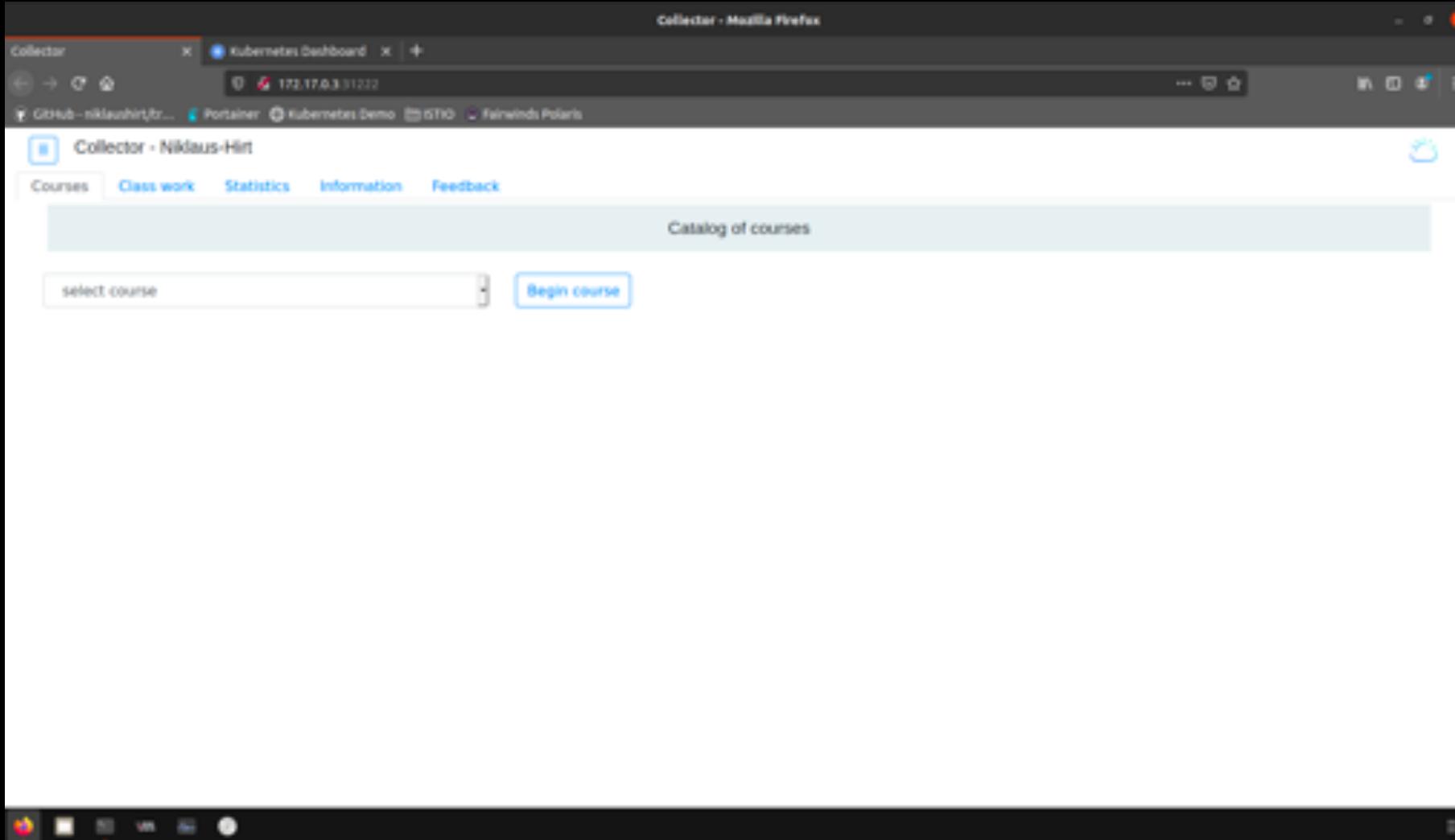
<pod>

Image pulling - wait

Dependencies - wait



Accessing your Personal Training Environment



When completed, your PTE and Kubernetes Dashboard will open automatically



JTC90 Lab Setup

EVERYBODY

Task 1: Download Training VM

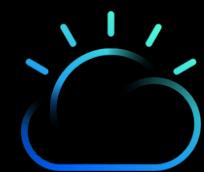
Task 2: Setup VMware / VirtualBox

OK

Task 3: Start Training VM

?

Task 4: Login / Check



Accessing your Personal Training Environment

Name will be shown



The screenshot shows a user interface for selecting a course. At the top, there is a header with a cloud icon and the text "Collector - Niklaus-Hirt". Below the header, there are five tabs: "Courses" (selected), "Class work", "Statistics", "Information", and "Feedback". To the right of the tabs, the text "Catalog of courses" is displayed. On the left, there is a dropdown menu with the placeholder "select course" and a list of course names. On the right, there is a button labeled "Begin course". A red box highlights the "Begin course" button, and a red arrow points from the text "Select course and press button to begin" to it.

- select course
- select course
- JTC01 Docker
- JTC02 Kubernetes Labs
- JTC10 Istio
- JTC14 Kubernetes Ansible Operators Labs
- JTC16 Kubernetes Security Labs
- JTC17 Kubernetes Advanced Security Labs
- JTC80 Kubernetes Introduction
- JTC90 Lab Setup

Current course catalog

Select course and
press button to begin



Class Work

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

The screenshot shows a course interface with a navigation bar at the top: 'Collector - test: K8s_101_01 Kubernetes Introduction', 'Courses' (selected), 'Class work' (highlighted in blue), 'Statistics', 'Information', and 'Feedback'. Below the navigation is a 'Task Intro' section. A green box highlights the 'Complete' button in the top right corner of the 'Task Intro' box. A red arrow points from the text 'Select class work and the blue portion of the screen is shown' to the 'Task Intro' header. Another red arrow points from the text 'Press the green Complete button to show the green portion.' to the 'Complete' button. A third red arrow points from the text 'Confirm completion by pressing the green "Press to mark completed" button.' to the 'Press to mark completed' button inside the green box.

Press the green Complete button to show the green portion.

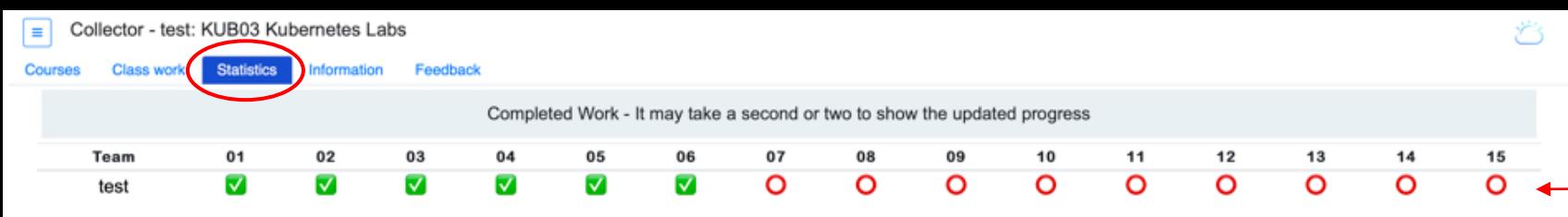
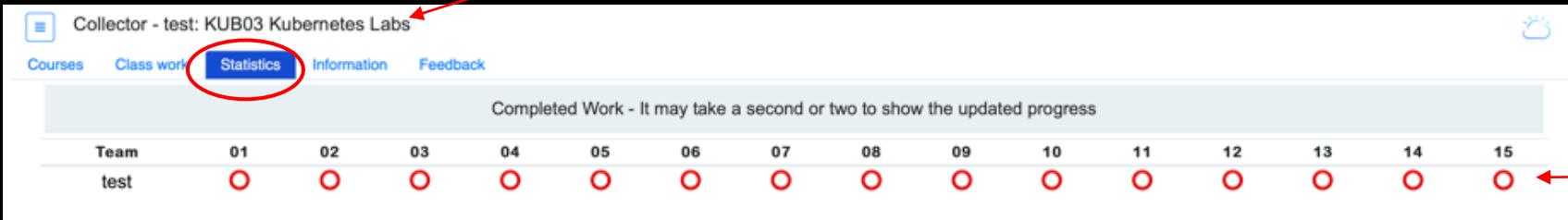
Confirm completion by pressing the green "Press to mark completed" button.

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



Following your progress

Course title



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

Green checkmark - item is completed

Red circle - item is waiting to be completed

Kubernetes Workshop Series

Kubernetes

01



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

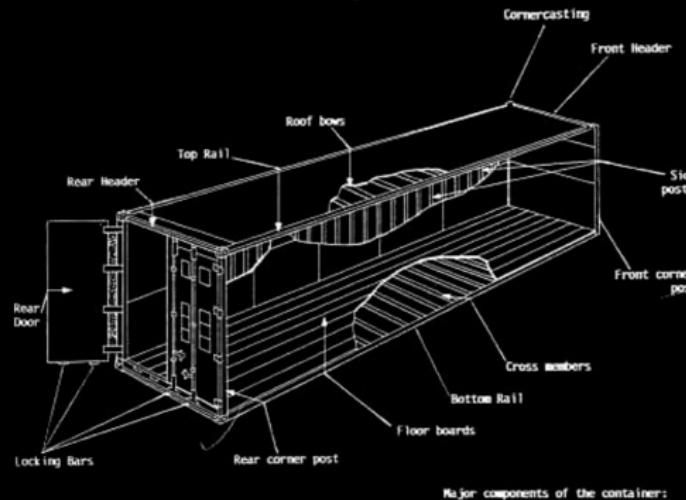
Containers work by **isolating the differences between applications** inside the container so that everything outside the container can be standardized

What happened so far... 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

Everybody Loves Containers

But when you go



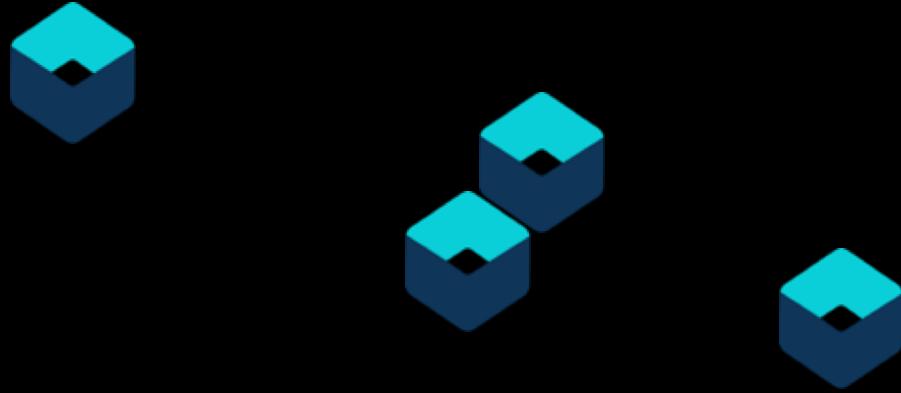
From this....



To this....



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



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



Pets vs Cattle

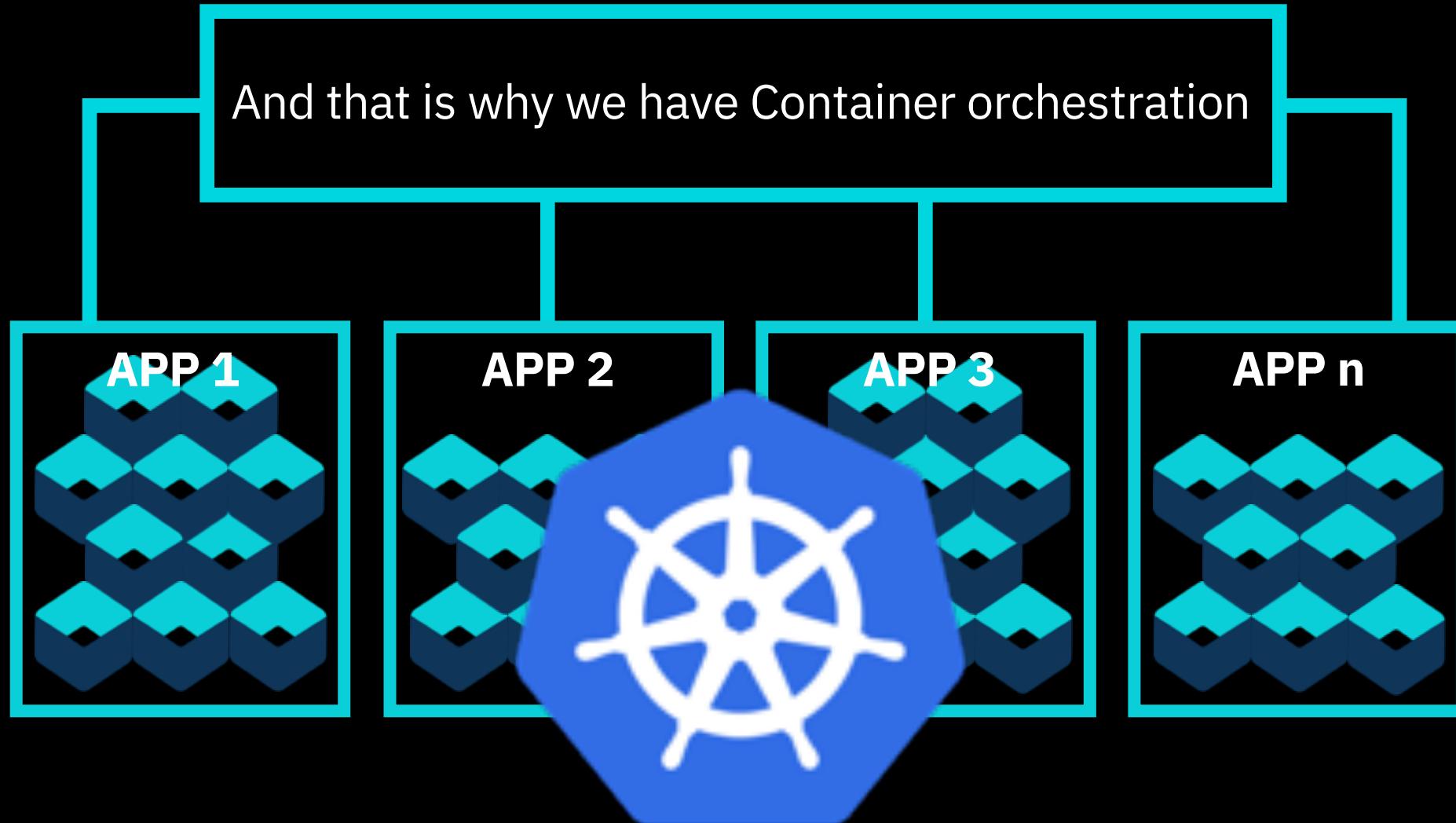
But soon you have many applications, many instances...

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

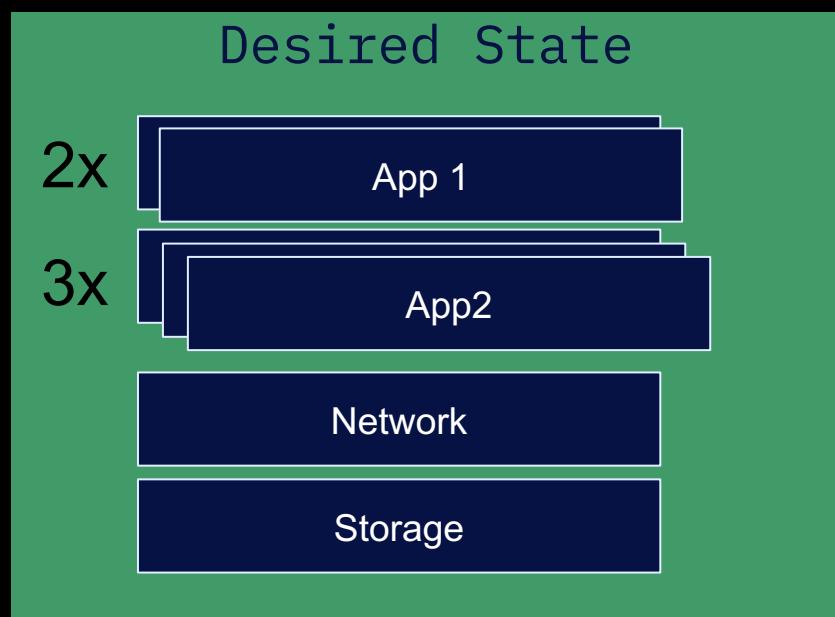
By contrast, in a declarative system, **the user** knows the desired state, supplies a representation of the desired state to the system, then **the system** reads the current state and determines the sequence of commands to transition the system to the desired state.

Kubernetes – Declarative System

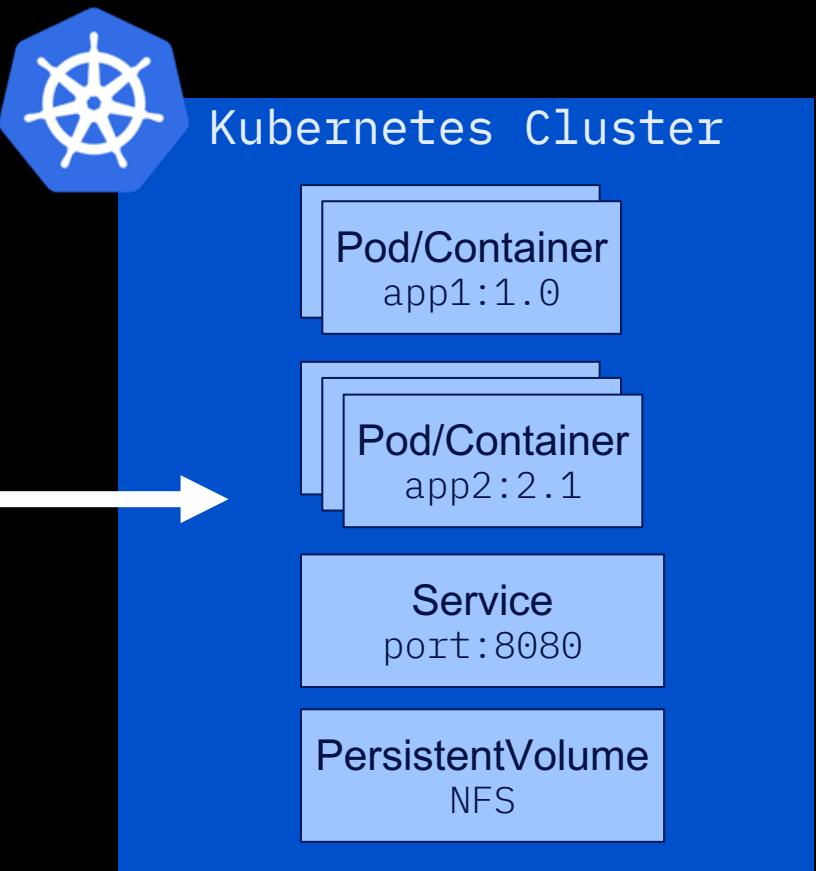


The Desired State

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:
```



So what is Kubernetes?



Container orchestrator

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

Manage applications, not machines

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

Designed for extensibility

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

Open source project managed by the Linux Foundation

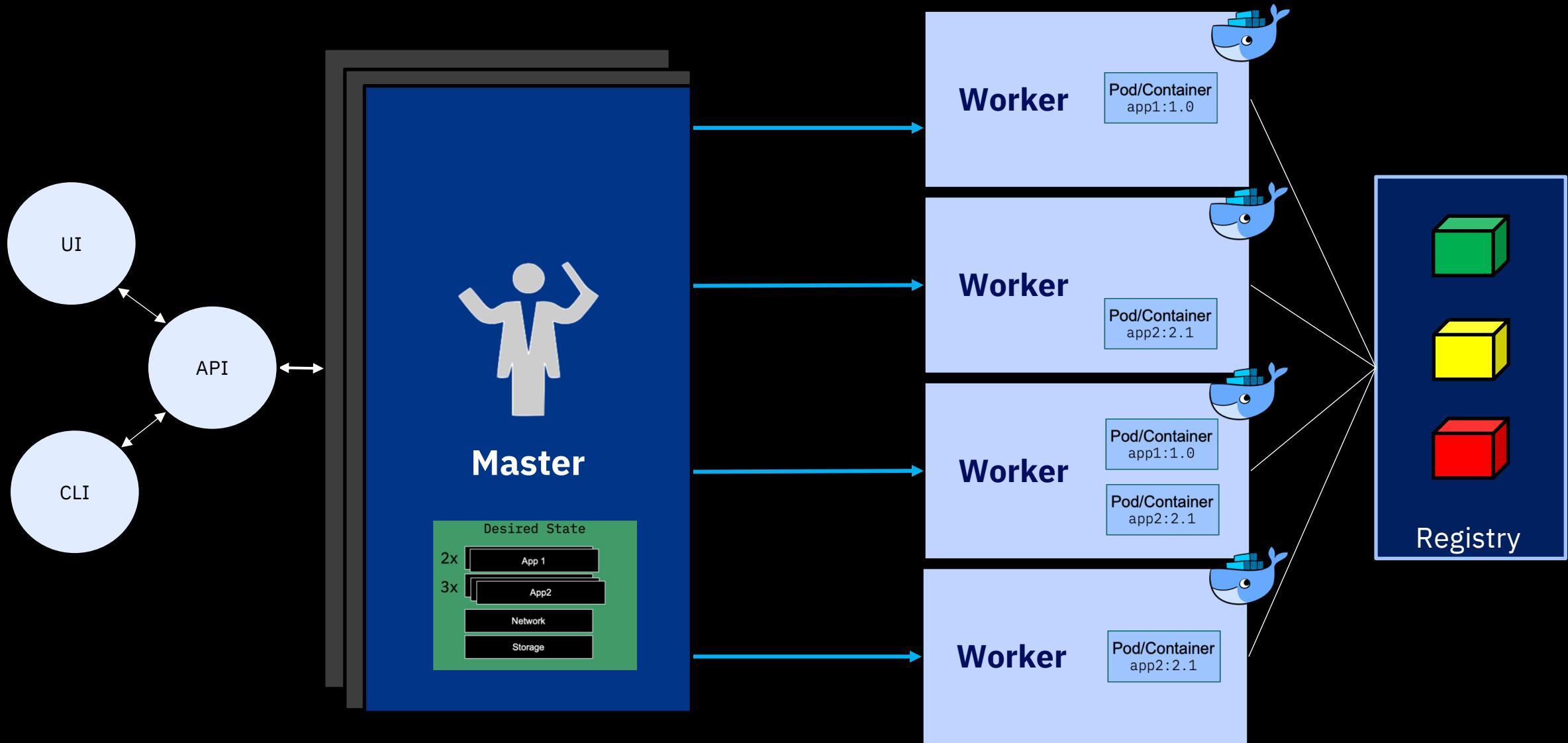
- Inspired and informed by Google's experiences and internal systems
- 100% open source, written in Go

Kubernetes Strengths



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

Kubernetes Management Architecture



Kubernetes Management Architecture



Nodes – hosts that run Kubernetes applications

API

Master nodes

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

- Controller Manager Server
- Scheduler

Worker nodes

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



Kubernetes Management Architecture

Pods

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

Deployment

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



ReplicationSet

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

Service

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



Kubernetes Management Architecture

ConfigMap

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

Labels

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

APT

Secrets

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

- Etcd
- API Server
- Controller Manager Server
- Scheduler

Worker Node 3

Worker Node n

Registry

What does Kubernetes really offer ?

Intelligent Scheduling



Automatically places containers based on their resource requirements and other constraints, while not sacrificing availability. Mix critical and best-effort workloads in order to drive up utilization and save even more resources.

Self Healing



Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

Horizontal Scaling



Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

Service Discovery and Load Balancing



No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives containers their own IP addresses and a single DNS name for a set of containers, and can load-balance across them.

Automated rollout and rollback



Kubernetes progressively rolls out changes to your application, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.

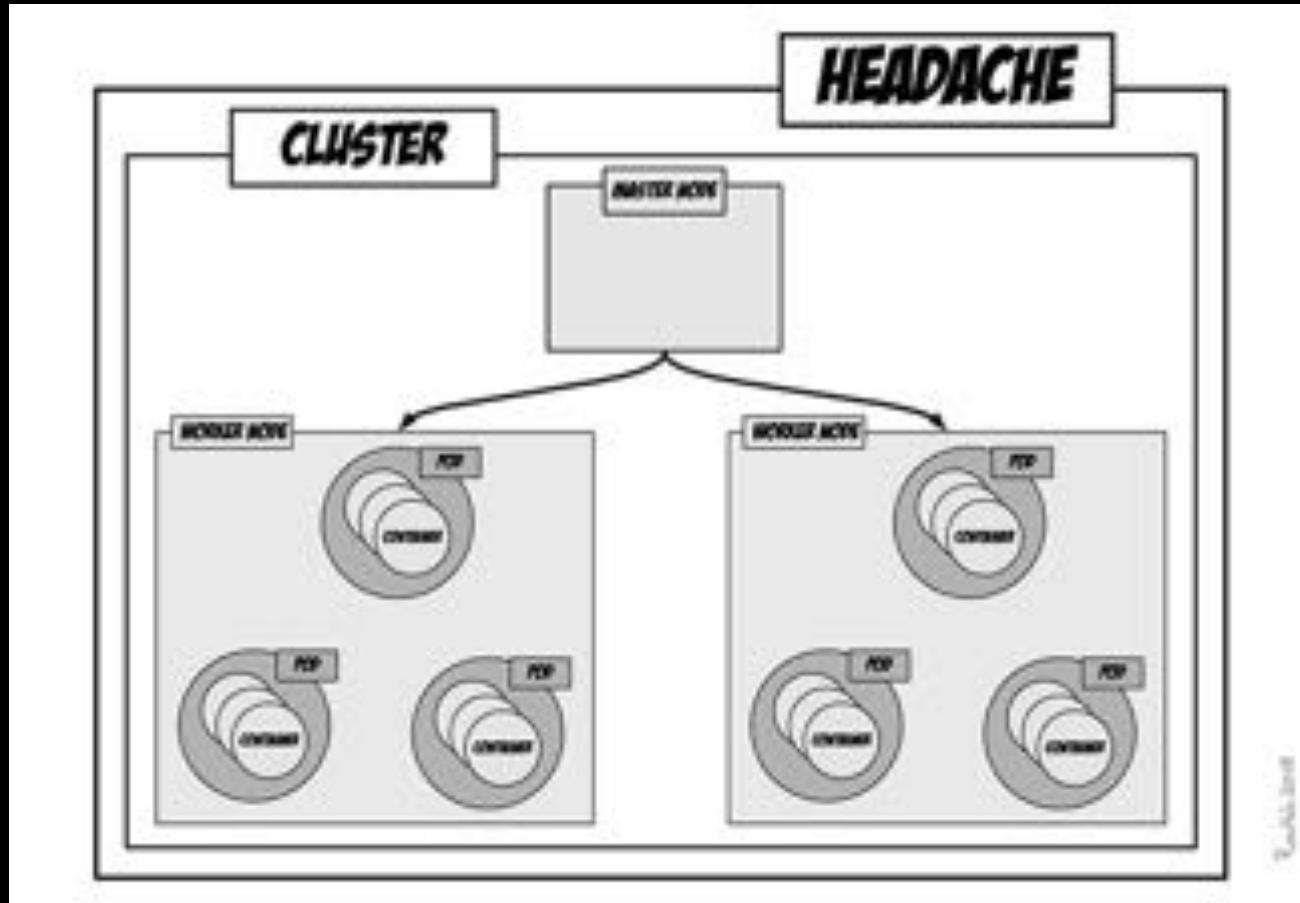
Secret and configuration management



Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.

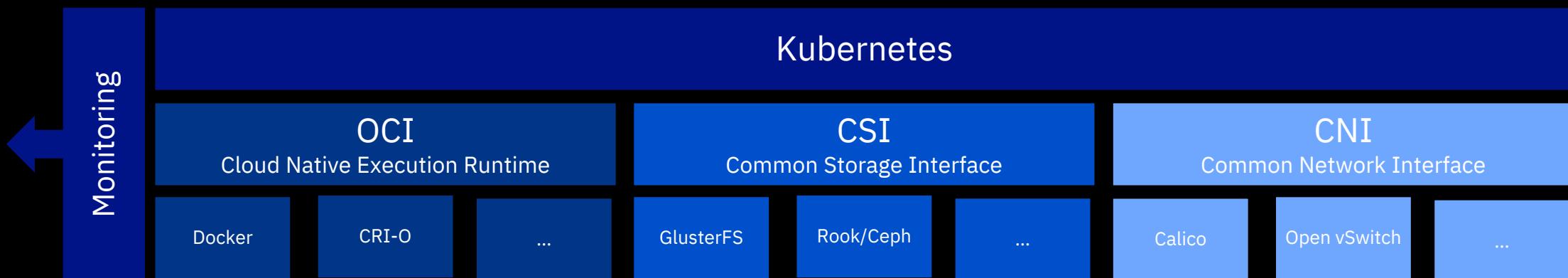
Kubernetes

Assemble vs. Operate



Source: ROELBOB - <https://devops.com/kubernetes-in-the-real-world/>

Kubernetes Cluster Layers



Kubernetes Cluster Architecture

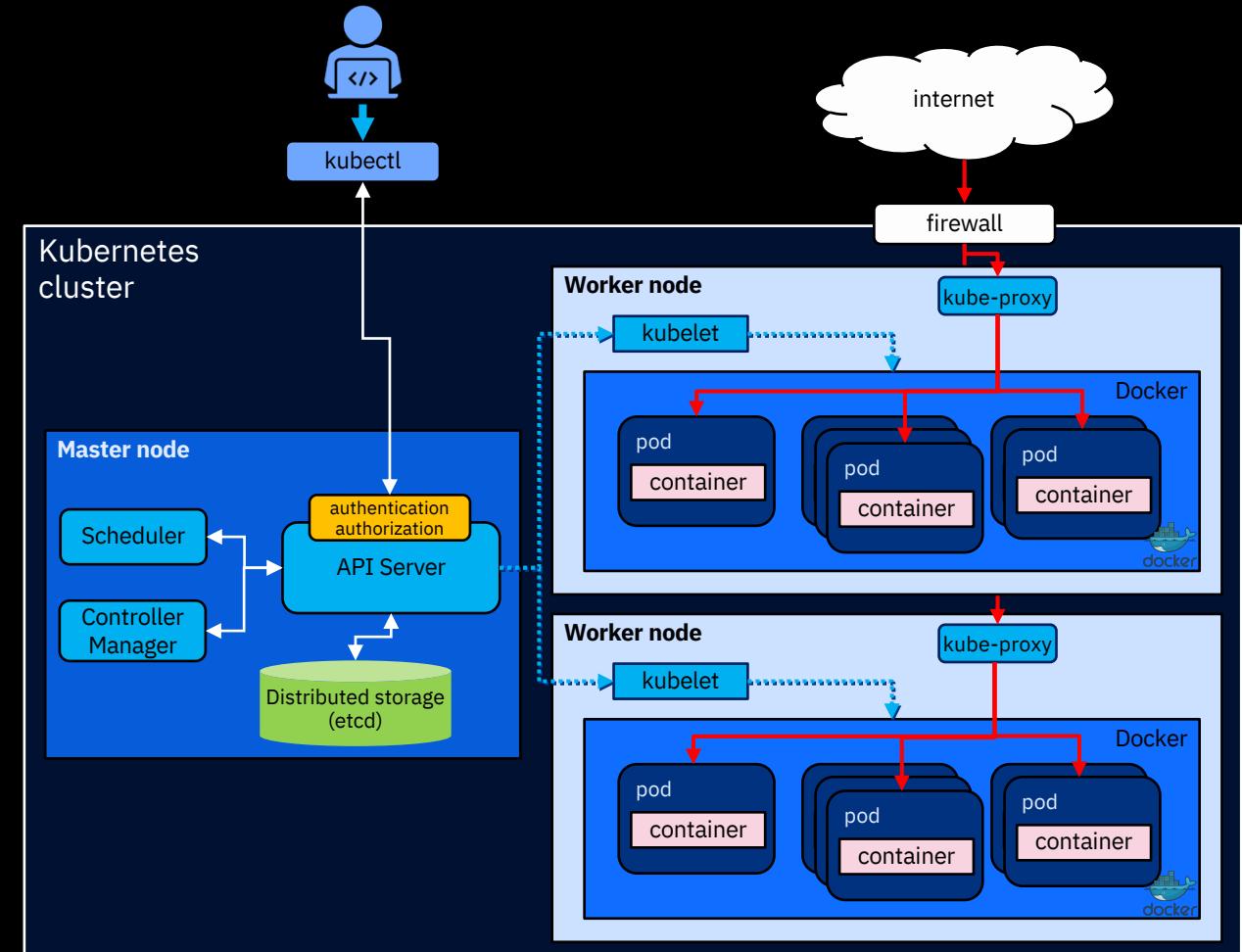


Master node

- 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





Master Node Components

Etcd

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

API Server

- Exposes API for managing Kubernetes
- Used by kubectl CLI

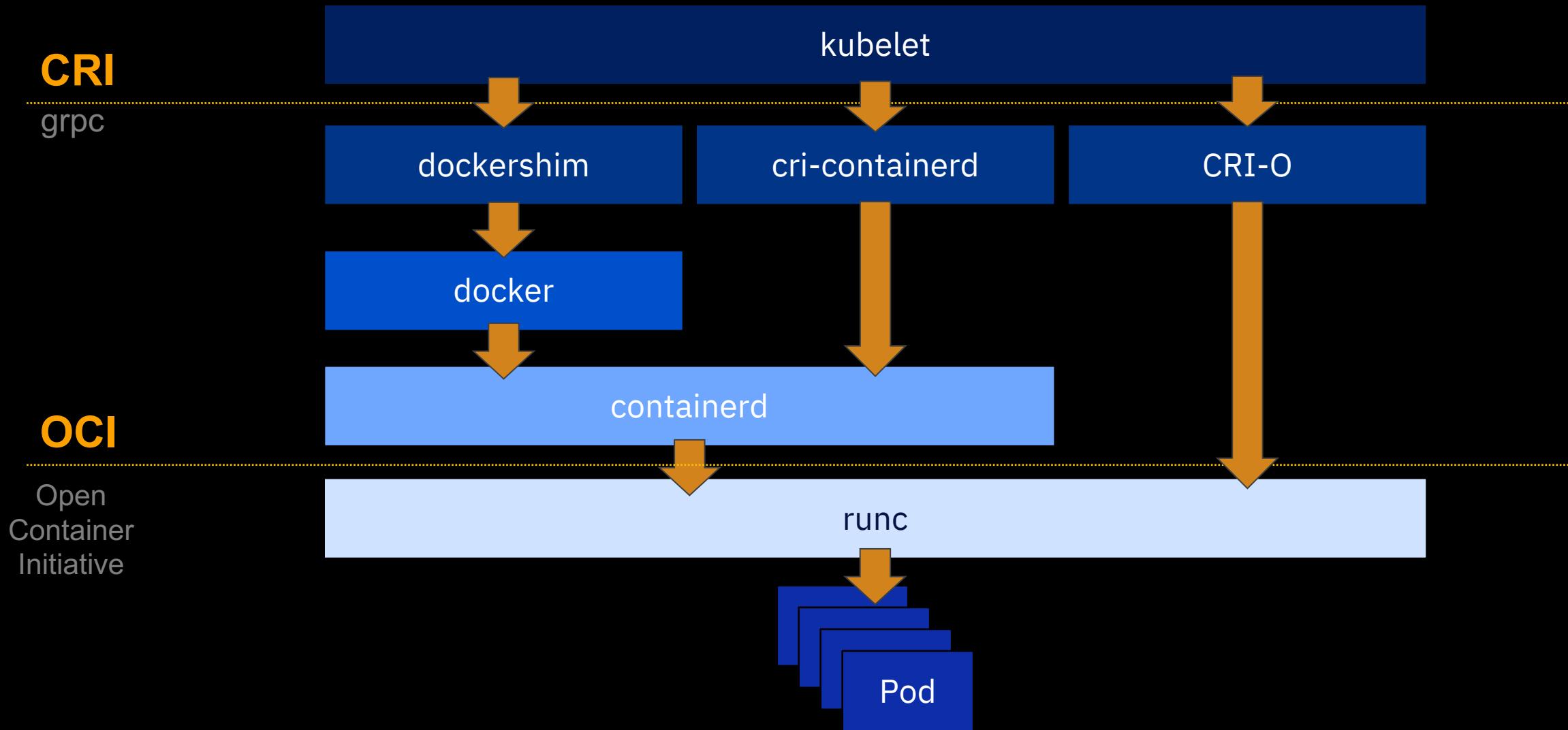
Controller manager

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

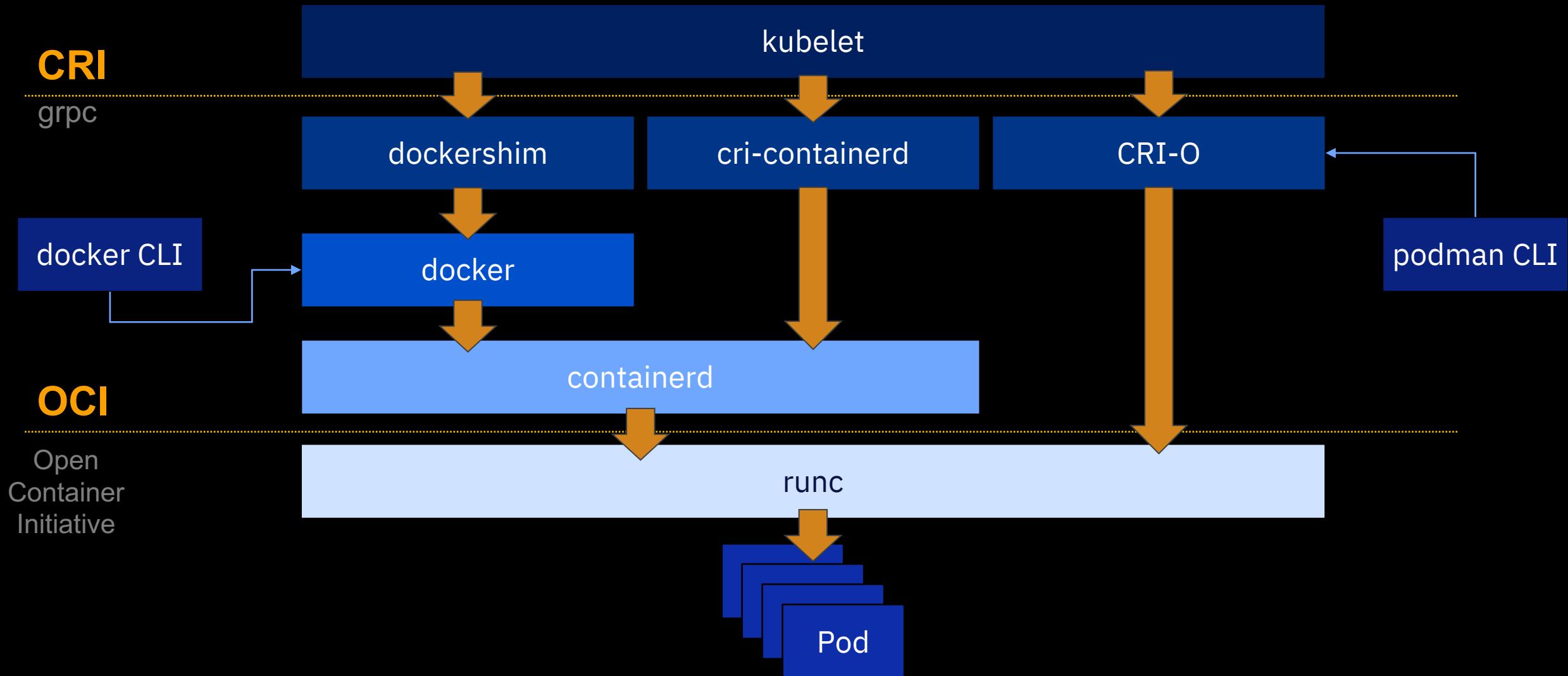
Scheduler

- Selects the worker node each pods runs in

Kubernetes – Common Runtime Interface



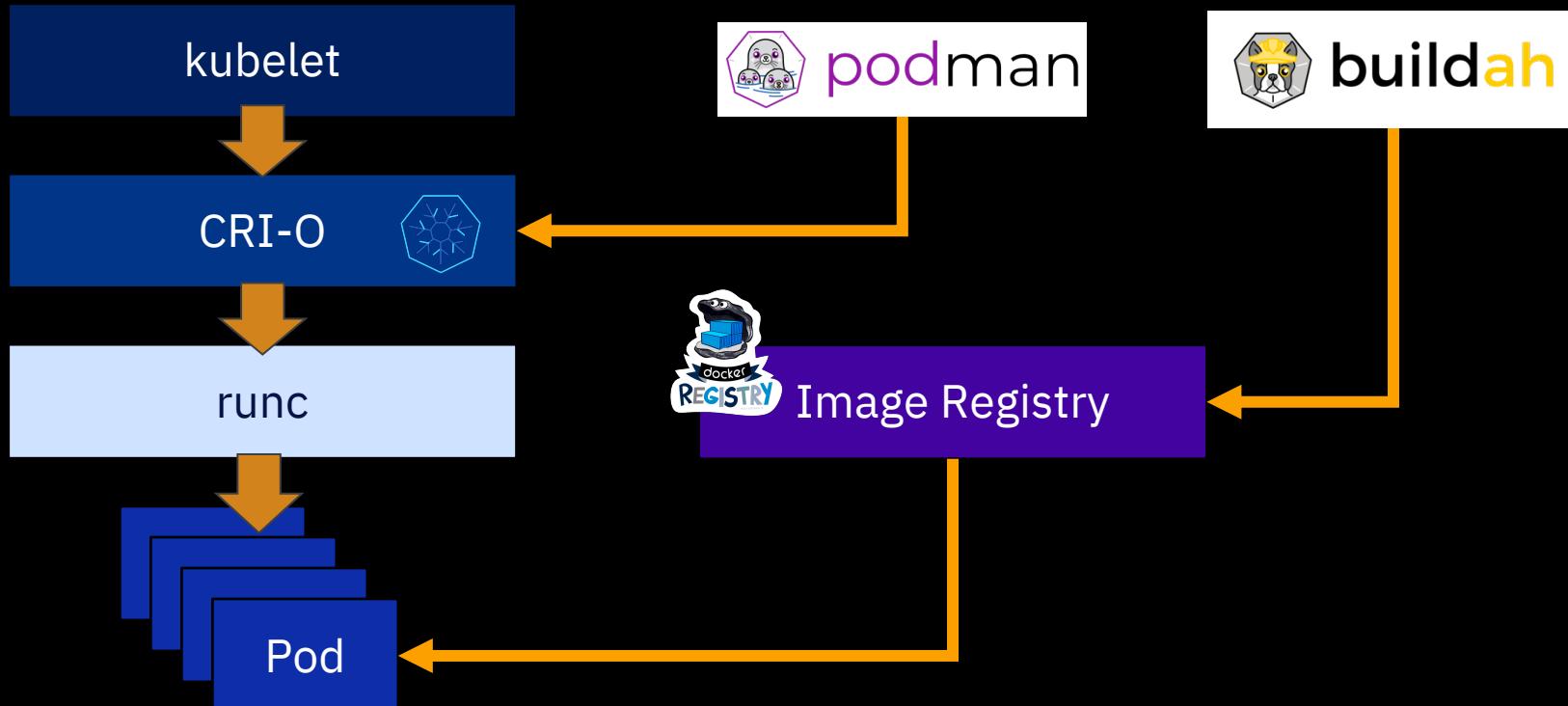
Kubernetes – Common Runtime Interface



Kubernetes – CRI-O



cri-O



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 1.16 APIs

As the Kubernetes API evolves, APIs are periodically reorganized or upgraded. When APIs evolve, the old API is deprecated and eventually removed..



Deprecated APIs in 1.16

- **NetworkPolicy**
Migrate from extensions/v1beta1 to networking.k8s.io/v1 API
- **PodSecurityPolicy**
Migrate from extensions/v1beta1 to policy/v1beta1 API
- **DaemonSet, Deployment, StatefulSet, and ReplicaSet**
Migrate from extensions/v1beta1 or apps/v1beta2 to apps/v1 API

Remediation

- Rewrite or
- `kubectl convert -f ./my-deployment.yaml --output-version apps/v1`

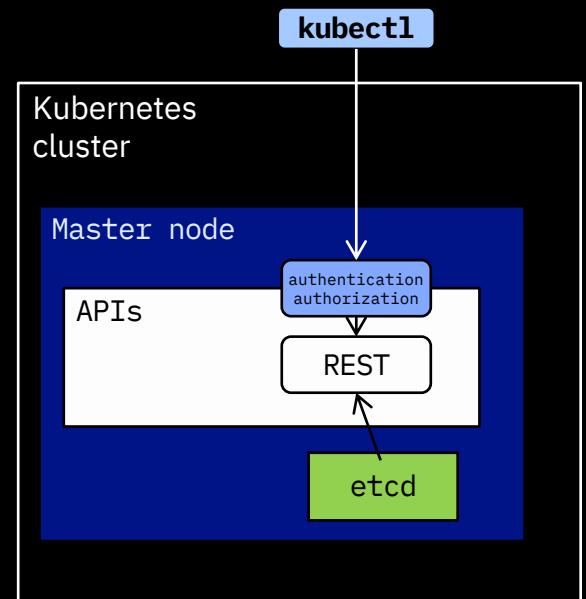
kubectl – talking to the Cluster



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

`kubectl` looks for a file named config in the `$HOME/.kube` directory. It contains all the information needed to communicate with your cluster.

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



kubectl – talking to the Cluster



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

`kubectl [command] [TYPE] [NAME]`

`kubectl create -f example.yaml`

Create objects in yaml file

`kubectl apply -f example.yaml`

Modify objects in yaml file

`kubectl delete -f example.yaml`

Delete objects in yaml file

`kubectl get pods`

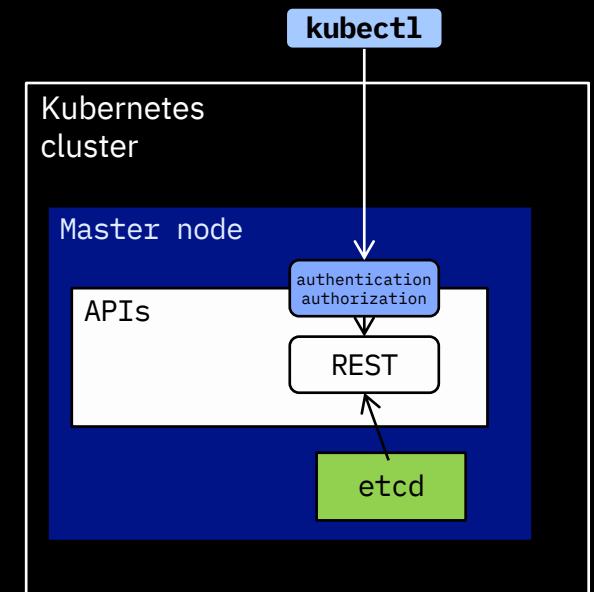
List Pods in Namespace

`kubectl describe nodes <node-name>`

Details about K8s object

`kubectl logs <pod-name>`

Get the logs for a Pod

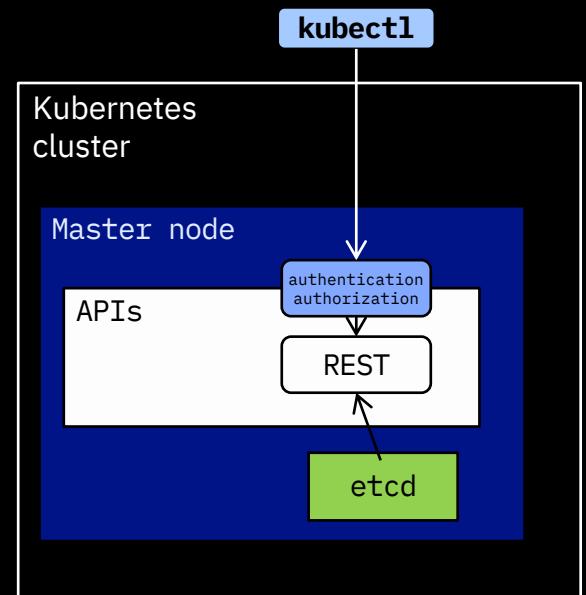


kubectl – talking to the Cluster



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

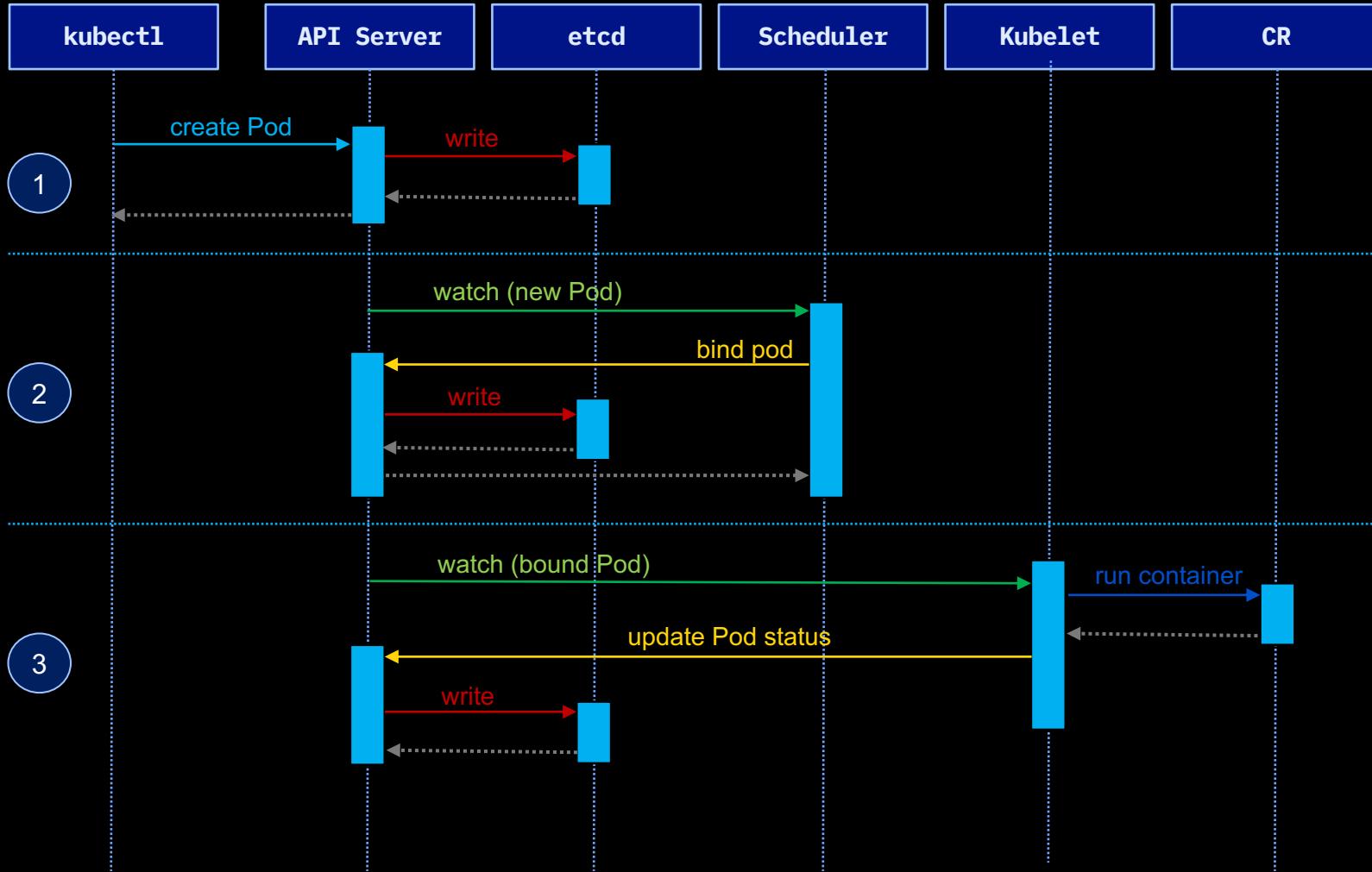
- Kubernetes uses `etcd` as a key-value database store.
- It stores the `configuration` of the Kubernetes cluster in etcd.
- It also stores the `actual state` of the system and the `desired state` of the system in etcd.
- Anything you might read from a `kubectl get xyz` command is coming from etcd.
- Any change you make via `kubectl create` will cause an entry in etcd to be updated.



kubectl – talking to the Cluster



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



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

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



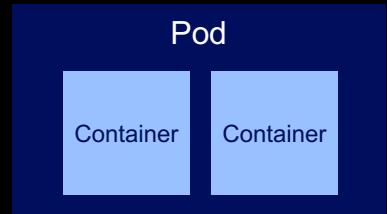
Pod

- 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
```

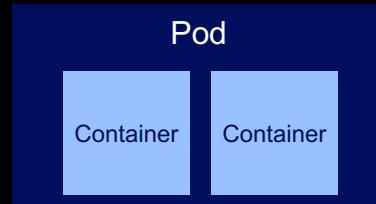
```
kubectl get pods  
NAME          READY  STATUS   RESTARTS  AGE  
nginx-5bd87f76c-vxc79  1/1    Running  0          29s
```

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

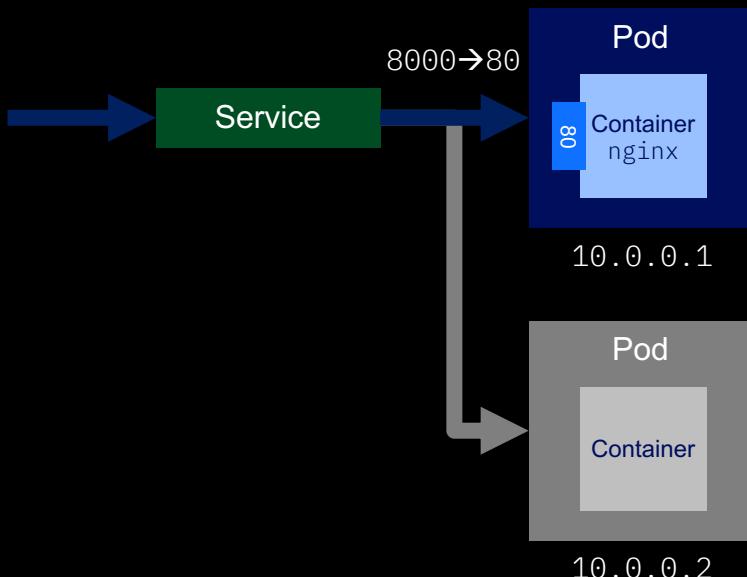


10.0.0.2

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

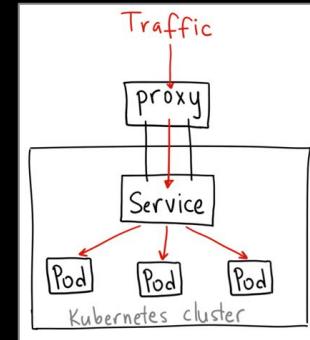


```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  ports:
    - port: 8000
      targetPort: 80
      protocol: TCP
  selector:
    app: nginx
```

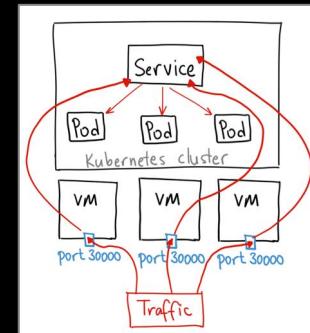
Service



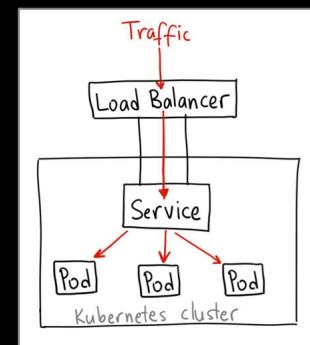
ClusterIP: This type exposes the service on the cluster internal IP. This means that the service is only reachable from within the cluster.



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.



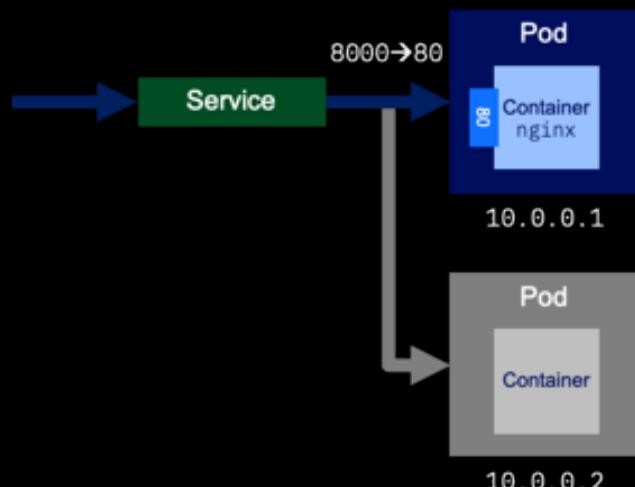


Creating a Service

```
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

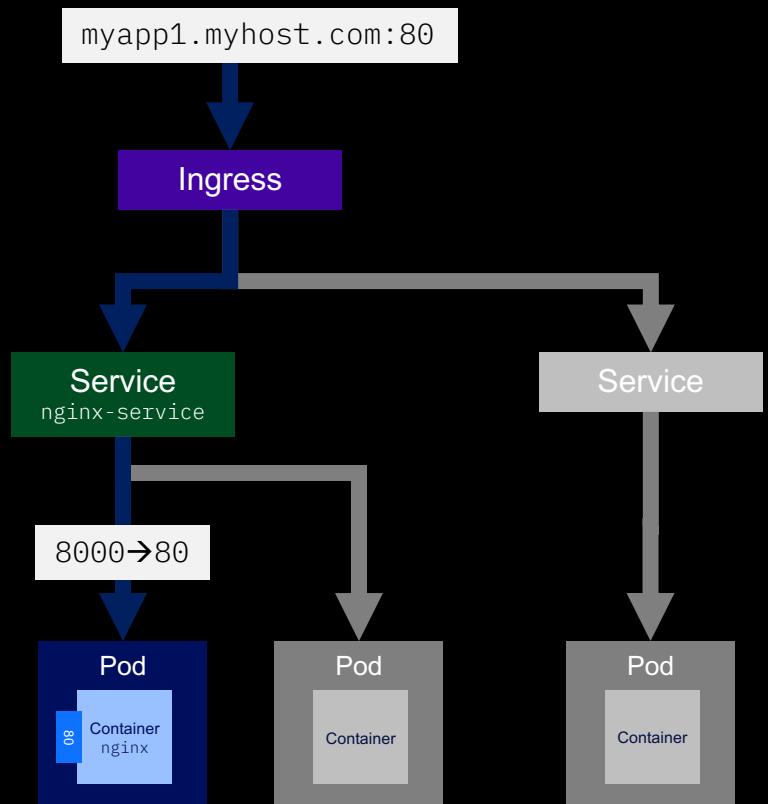
```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  ports:
    - port: 8000
      targetPort: 80
      protocol: TCP
  selector:
    app: nginx
```

Ingress



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

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

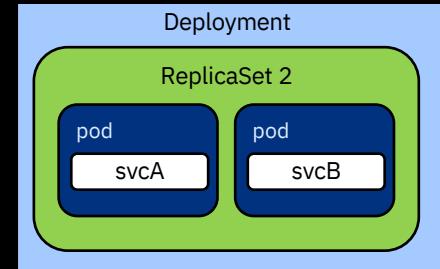


Deployments & ReplicaSets



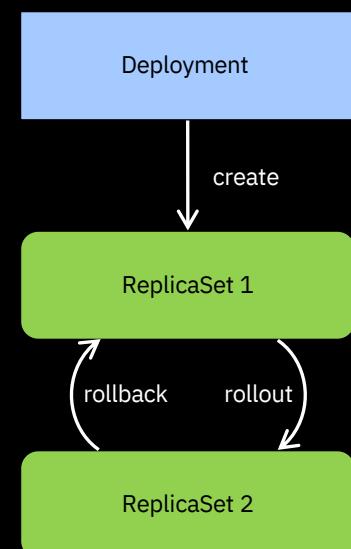
Deployment

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



ReplicaSet

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



Deployment



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

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

example.yaml

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



StatefulSet, DaemonSet, Job...

StatefulSet

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

DaemonSet

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

Job

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

Cron Job

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



Naming

Name

- Each resource object by type has a unique name

Namespace

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

Resource Quota

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

Naming



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

When referred from the same namespace

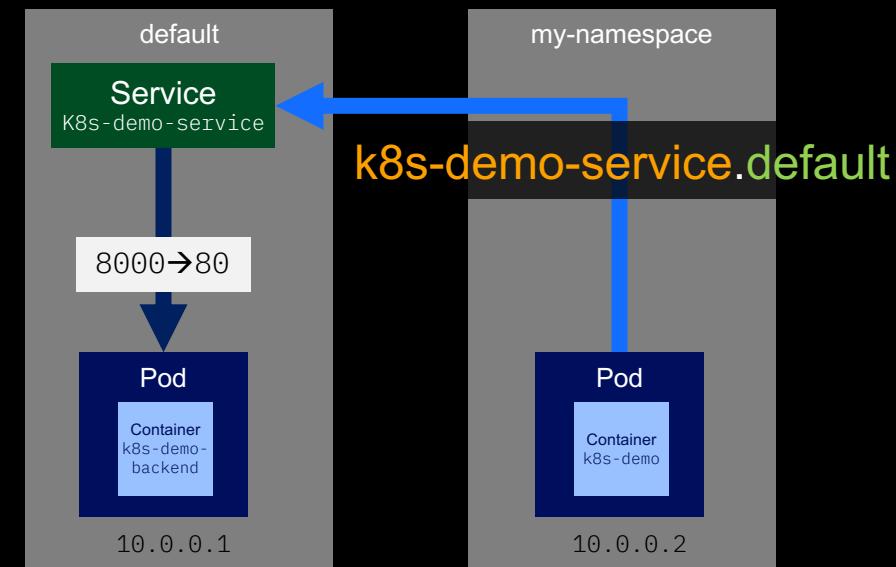
k8s-demo-service

When referred from another namespace

k8s-demo-service.default

Fully qualified name

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



Kubernetes User Security



- **Authentication**
 - OIDC Tokens
 - ServiceAccount Tokens
- **Authorization**
 - RBAC
 - Role
 - RoleBinding
 - ClusterRole
 - ClusterRoleBinding
- **Namespaces**
 - Help to scope access control to a specific namespace

```
---  
apiVersion: rbac.authorization.k8s.io/v1beta1  
kind: ClusterRole  
metadata:  
  name: viewer  
rules:  
  - apiGroups:  
    - apps  
    - extensions  
    resources:  
    - deployments  
    - replicaset  
    verbs:  
    - get  
    - list  
    - watch  
---  
apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRoleBinding  
metadata:  
  name: cicd-viewer  
roleRef:  
  apiGroup: rbac.authorization.k8s.io  
  kind: ClusterRole  
  name: viewer
```

More about this in two weeks

JTC17 – Kubernetes Advanced Security

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

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**)

Kubernetes Autoscaling



Horizontal Pod Autoscaling (HPA)

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

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

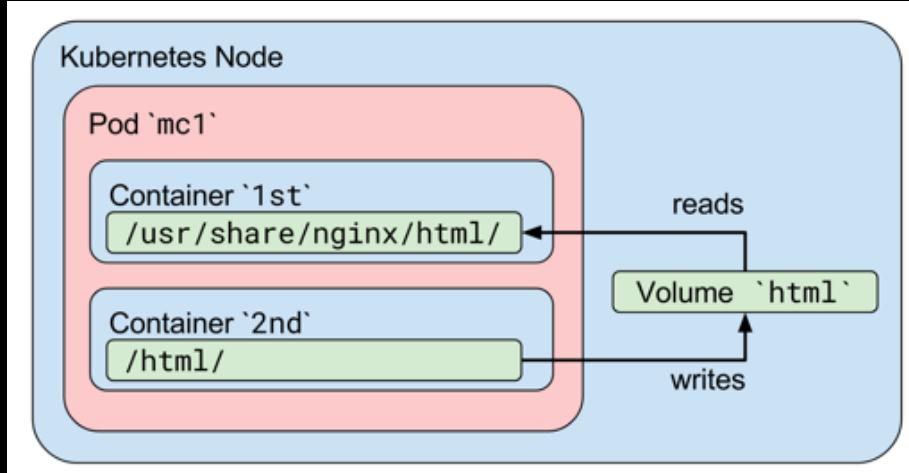
Creates a horizontal pod autoscaler

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

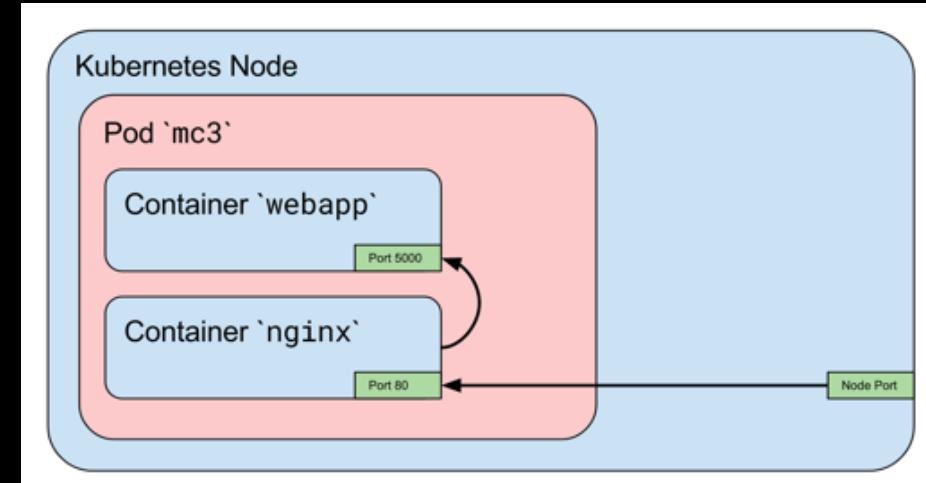
Multi-Container Pod Design Patterns



Shared Volumes

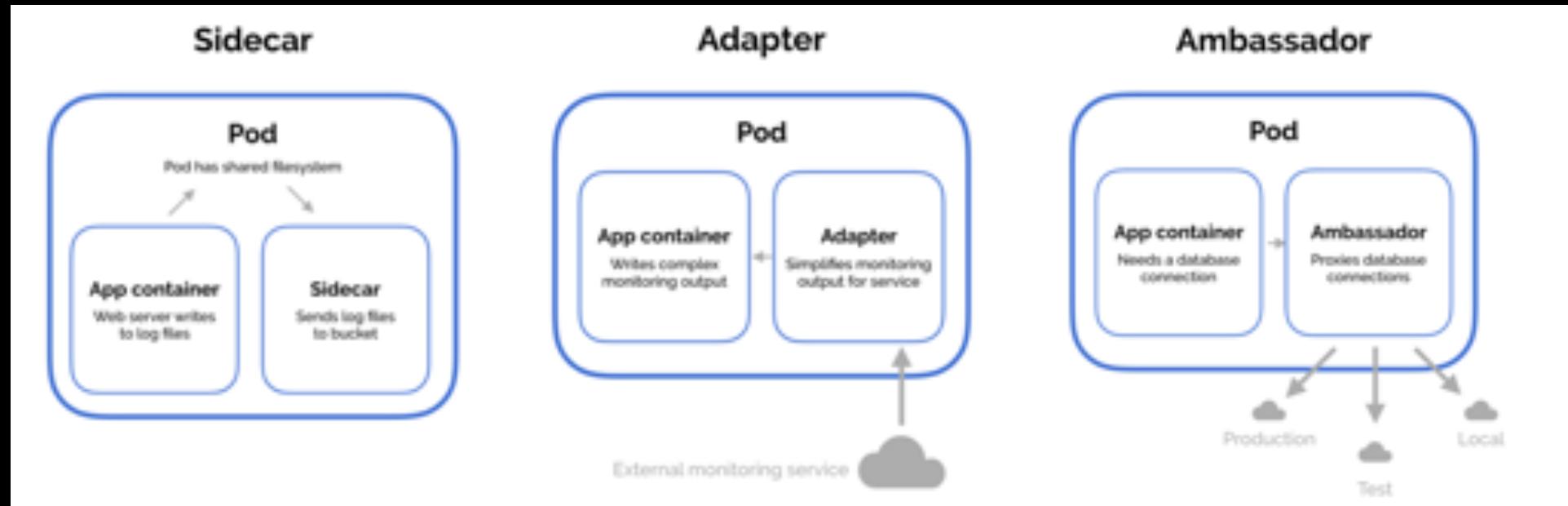


No Network isolation



In what order containers are being started in a Pod?

Multi-Container Pod Design Patterns



Sidecar pattern

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

Adapter pattern

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

Ambassador pattern

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

HELM Charts



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



Catalog

All Categories > Search items Filter

Category	Chart Name	Description	Type
Blockchain	acs-engine-autoscaler	DEPRECATED Scales worker nodes within agent pools	KUBE [Deprecated]
Business Automation	aerospike	A helm chart for Aerospike in Kubernetes	KUBE
Data	airflow	Airflow is a platform to programmatically author, schedule and monitor workflows	KUBE
Data Science & Analytics	ark	A Helm chart for ark	KUBE
DevOps	artifactory	DEPRECATED Universal Repository Manager supporting all major packaging formats, build tools	KUBE [Deprecated]
Integration	artifactory-ha	DEPRECATED Universal Repository Manager supporting all major packaging formats, build tools	KUBE [Deprecated]
IoT	artifactory-ha	Universal Repository Manager supporting all major packaging formats, build tools and CI serv...	KUBE [Community Charts]
Network	audit-logging	Audit logging storage and search management solution	Ingress Charts
Operations	auditbeat	A lightweight shipper to audit the activities of users and processes on your systems	KUBE
Runtimes & Frameworks	auth-apis	SCP IAM Token Service	Ingress Charts
Security	auth-idp	SCP Security Authentication Provider	Ingress Charts
Storage	auth-pap	SCP IAM Policy Administration	Ingress Charts
Tools	auth-pdp	SCP IAM Policy Decision	Ingress Charts
Other	aws-cluster-autoscaler	Scales worker nodes within auto-scaling groups	KUBE [Deprecated]
	boltbond	BoltBond is an innovative payment network and a new kind of money	KUBE
	bookstack	BookStack is a simple, self-hosted, easy-to-use platform for organizing and storing informa...	KUBE

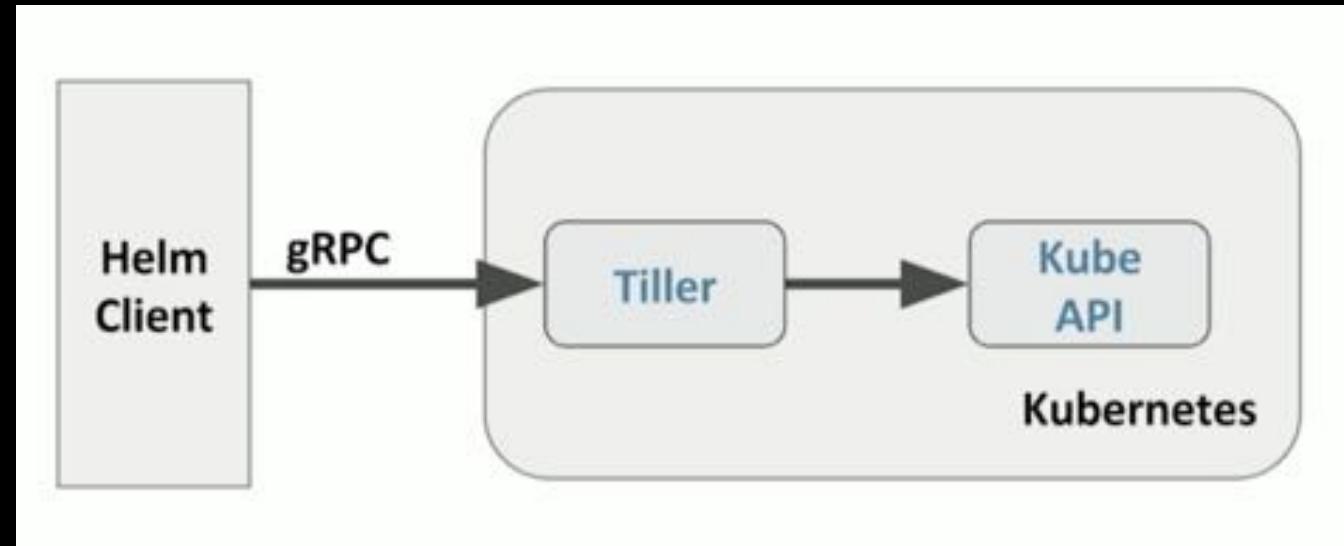
HELM Example



Helm 2 has 2 Components :

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

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



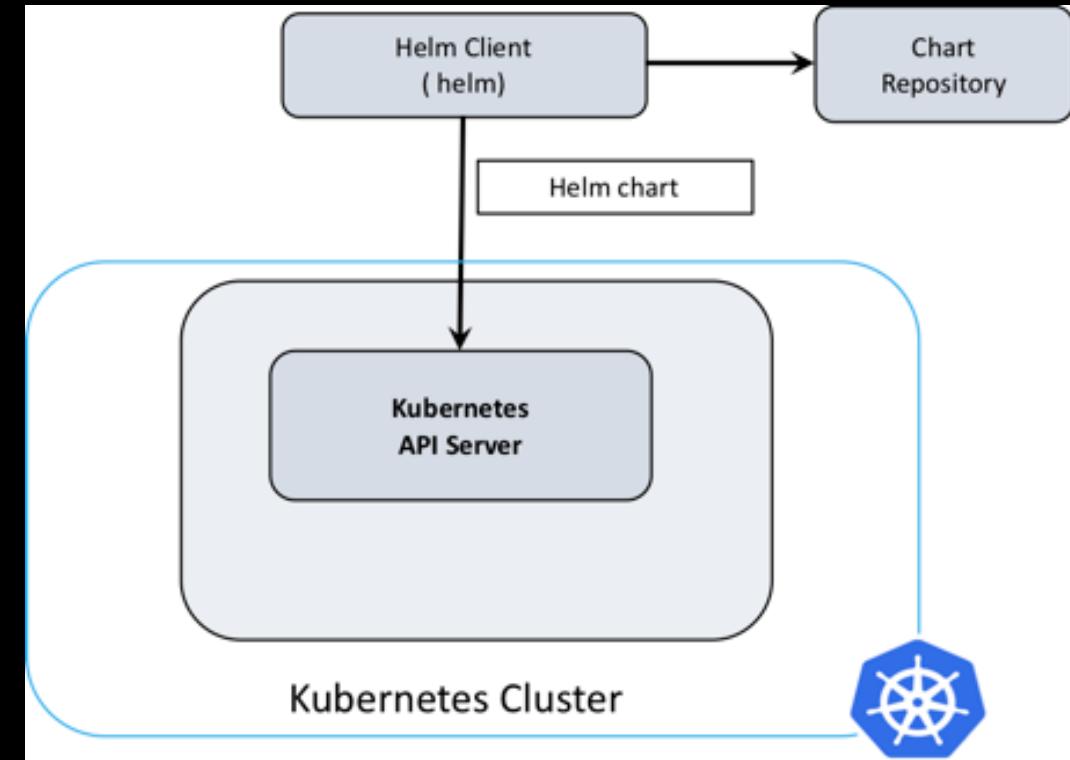
HELM Example



Helm 3 uses Custom Resource Definitions (CRD):

Tillerless : No tiller needed anymore

Client : The Helm client interacts with the cluster and stores information in in-Cluster CRDs.





HELM Example

- **Chart :**

A Helm package that contains information sufficient for installing a set of Kubernetes resources into a Kubernetes cluster.

- **Chart.yaml**

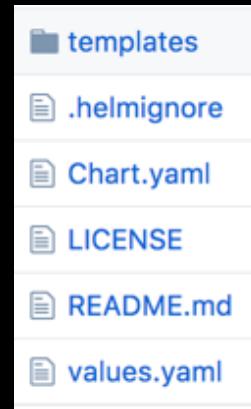
Basic information about the chart (name, versions, links, icon, ...)

- **Templates (yaml)**

Kubernetes deployment manifests (yaml) with placeholders for external parameters.

- **values.yaml**

A set of variables that will show up in the UI (or customized at the helm command line) and be passed on at deploy time to the deployment manifests.



Deployment Chart

```
1 apiVersion: v1
2 kind: PersistentVolumeClaim
3 metadata:
4   name: {{ template "fullname" . }}-pv-claim
5 spec:
6   accessModes:
7     - ReadWriteMany
8   resources:
9     requests:
10       storage: {{ .Values.storage.size }}
```

Values.yaml file

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

Resources



Linux Foundation – Introduction to Kubernetes

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

Kubernetes tutorial

- <https://kubernetes.io/docs/tutorials/kubernetes-basics/>

Introduction to container orchestration

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

TNS Research: The Present State of Container Orchestration

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

Large-scale cluster management at Google with Borg

- <https://research.google.com/pubs/pub43438.html>

[Cognitiveclass.ai](#) – Free basic to advanced classes on AI, Cloud... etc

 Learning Paths Courses Mobile Apps Badges Business Competitions Explore new learning opportunities Login



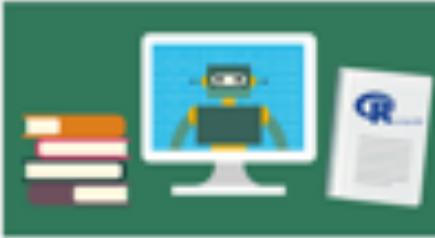
[How to Build Watson AI & Swift APIs and Make Money](#)
Cognitive Class: SW0201EN
 Intermediate



[Beyond the Basics: Istio and IBM Cloud Kubernetes Service](#)
Cognitive Class: CO0401EN
 Advanced



[Deep Learning with TensorFlow](#)
Cognitive Class: MLD120ENv2
 Advanced



[Machine Learning with R](#)
Cognitive Class: ML0151EN
 Intermediate

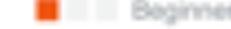


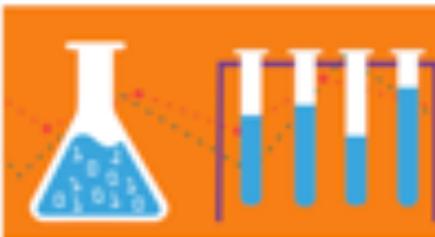
[Data Analysis with Python](#)
Cognitive Class: DA0101EN
 Beginner



[Data Visualization with R](#)
Cognitive Class: DV0151EN
 Beginner



[Spark Fundamentals I](#)
Cognitive Class: BD0211EN
 Beginner



[Data Science Methodology](#)
Cognitive Class: DS0103EN
 Beginner

Benefits of using a Kubernetes Based Platform

Speed

- Fast boot-time
- Easy scalability
- Rapid deployment

Portability

- Between different environments
- Between private and public cloud

Efficiency

- Better usage of computer resources

Automation

- Next level standardization and automation

13x

More software releases

Eliminate

“works on my machine” syndrome

~47%

Reduction of VMs,
OS licensing and
server cost

Reduce

operation cost

QUESTIONS?





Kubernetes Workshop Series
Kubernetes – *Use Kubernetes in
the cloud*

02



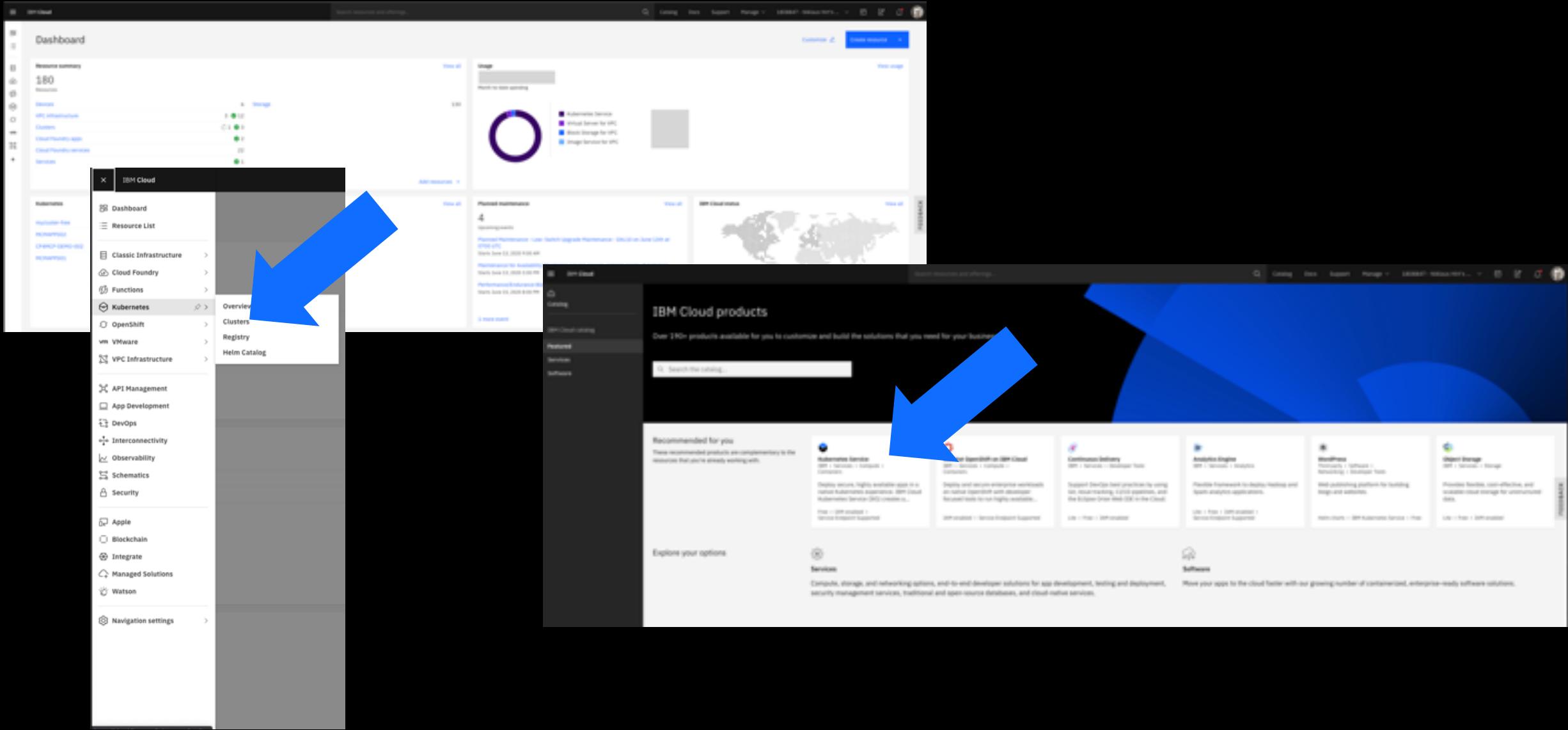
Kubernetes in the Cloud

Go to cloud.ibm.com

The screenshot shows the "Create an account" process on the IBM Cloud website. At the top left is the "IBM Cloud" logo. Below it is a sun icon. A link "Already have an IBM Cloud account? [Log in](#)" is visible. The main heading is "Create an account". The first step, "1. Account information", contains fields for "Email" (with "hkh@ch.ibm.com" entered) and "Password". A "next" button with a downward arrow is located below these fields. Subsequent steps are "2. Verify email" and "3. Personal information". At the bottom of the form is a "Create account" button.

[Log in with SoftLayer ID](#)

Kubernetes in the Cloud



Kubernetes in the Cloud

The screenshot shows the IBM Cloud interface for creating a Kubernetes cluster. The top navigation bar includes links for Home, Catalog, Marketplace, Support, Manage, and Logout. The main page title is "Kubernetes Cluster". Below it, there are two tabs: "Create" (highlighted in blue) and "About". On the right side, a sidebar titled "Summary" displays the cluster configuration: "Kubernetes cluster" and "Worker node" (Free - 2 vCPUs + 4GB RAM). The sidebar also indicates the plan is "Free".

Select a plan
Choose between a free or standard cluster. For more differences, see the [docs](#).

Free cluster
New to Kubernetes? Create a 30-day cluster with 2 worker nodes and limited capabilities for quick tests and learning how Kubernetes works.

Standard cluster
Ready to get to work! Create a cluster with fully-customizable compute, storage, and networking options for all your development and production environments.

Starting from \$0.122 monthly

Cluster type and version: Kubernetes 1.21.0 (Custom, Default)

Cluster name: mycluster

Resource group: default

Total monthly cost: Free (estimated)

Additional charges for bandwidth might apply
[Learn more](#)
*Actual monthly total will vary with usage pricing

Create

Add to estimate

Kubernetes in the Cloud

The screenshot shows the IBM Cloud interface for creating a new Kubernetes cluster named "mycluster". The status is "Requesting creation..." with a progress bar showing "8 days" remaining. The "Access" tab is selected, displaying instructions for setting up CLIs and connecting to the cluster via SSH or Kubectl. It includes step-by-step guides for logging in, setting the cluster context, and verifying connection. A note says "Tip: Plan to use multiple clusters? Repeat these steps for each cluster." Below this, there's a section about enabling continuous delivery. At the bottom, there are "Enable machine" and "Learn more" buttons, along with a "Want to learn more? Check out the docs." link.

mycluster Requesting creation... 8 days Add tags

Access Overview Worker Nodes Worker Pools Add-ons Environs

Before your cluster provisions, set up your CLI tools.
Download and install a few CLI tools and the Kubernetes Service plugin.

root@ 45.192.1.134:8080~1/kubelet

After your cluster provisions, gain access.

1. Log in to your IBM Cloud account, include the --region option if you have a federated ID.

```
ibmcloud login -a ibmcloud -r us-south -g default
```

2. Set the Kubernetes context to your cluster for this terminal session. For more information about this command, [see the docs](#).

```
ibmcloud ks cluster config --cluster myclusterusouth
```

3. Verify that you can connect to your cluster.

```
kubectl config get-contexts
```

Now, you can run Kubectl commands to manage your cluster workloads in IBM Cloud! For a full list of commands, see the [Kubernetes docs](#).

Tip: Plan to use multiple clusters? Repeat these steps for each cluster. Then you can use the kubectl config user-context command to switch your context to a different cluster.

It can take a few minutes for your cluster to be ready. While you wait, try [creating a registry](#). When it's ready, you can use the Kubernetes dashboard button to access your cluster information.

Next step: Enable continuous delivery

Enable continuous delivery to automate builds, test, and deployments through the Delivery Pipeline, git Hooks, Issue tracking, and more.

Enable machine Learn more

Want to learn more? [Check out the docs](#).

Kubernetes in the Cloud

Let me show you...

QUESTIONS?





Kubernetes Workshop Series

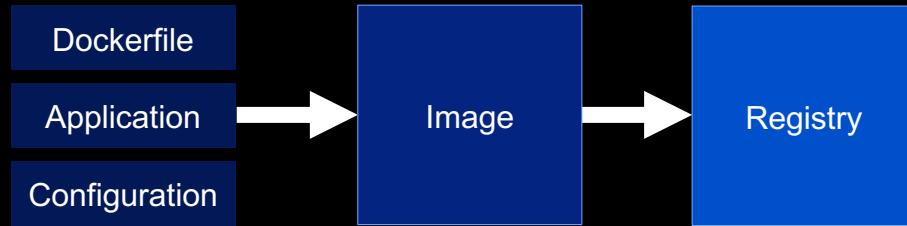
Kubernetes - Applied

03



..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:

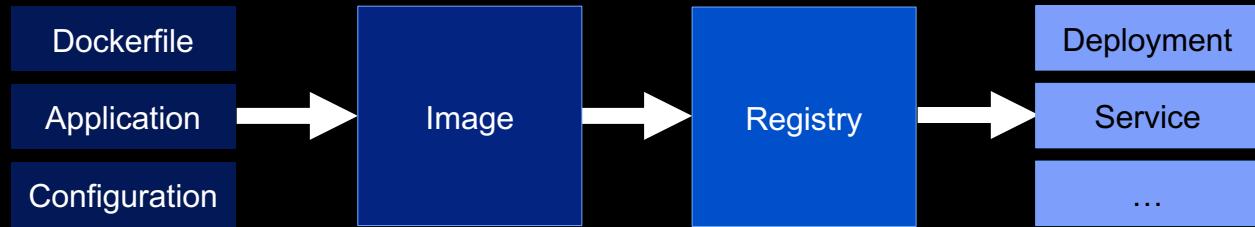
- Built Application or source
(NodeJS, Java, Golang, ...)
- Dockerfile
→ Use the right base image for your app

```

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



Minimal Application – Kubernetes Manifests



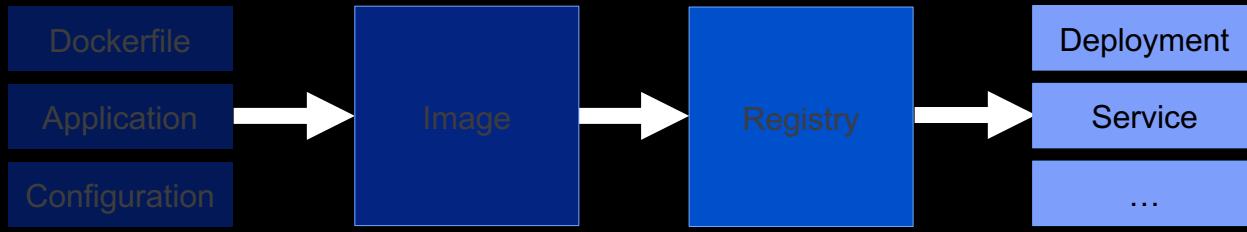
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

Minimum components you need:

- Deployment
- Service



Minimal Application – Kubernetes Manifests

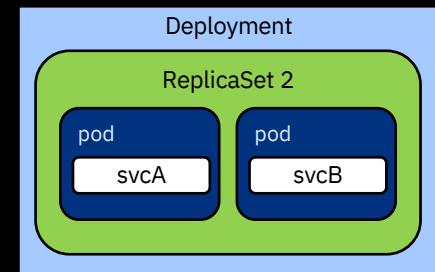


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

Deployment

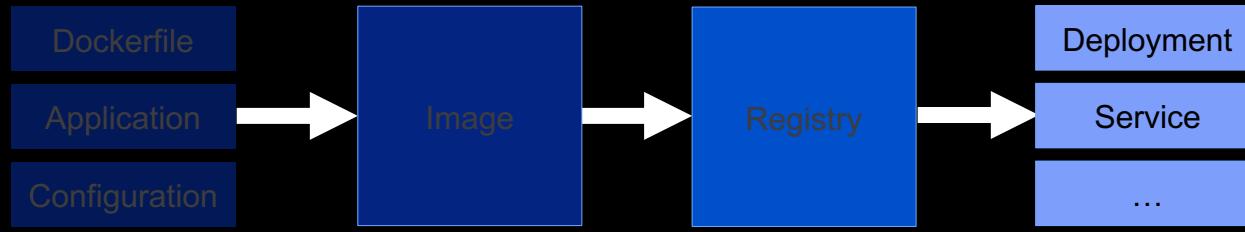
Minimum components you need:

- Deployment
- Service





Minimal Application – Kubernetes Manifests

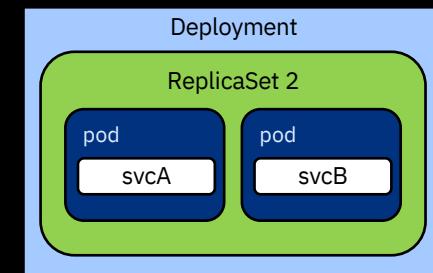


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

A vertical bracket on the left side of the code highlights the **Deployment** and **ReplicaSet** sections. The **Deployment** section is highlighted in blue, and the **ReplicaSet** section is highlighted in green.

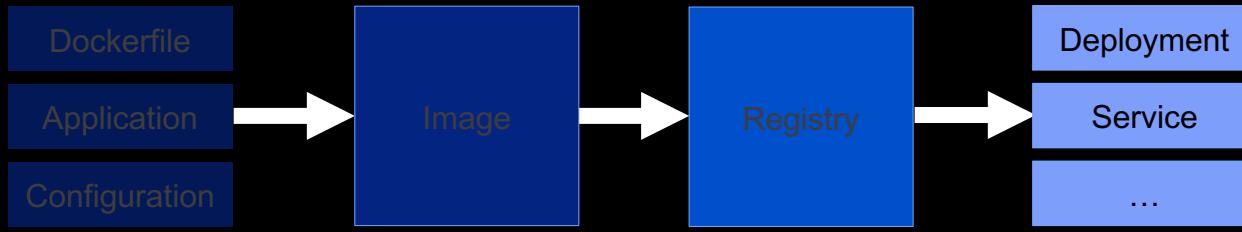
Minimum components you need:

- 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
```

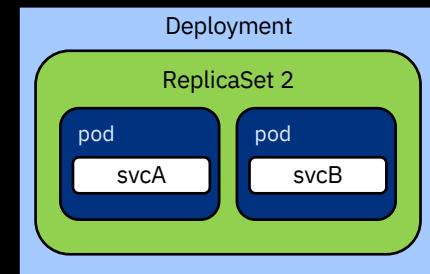
Deployment

ReplicaSet

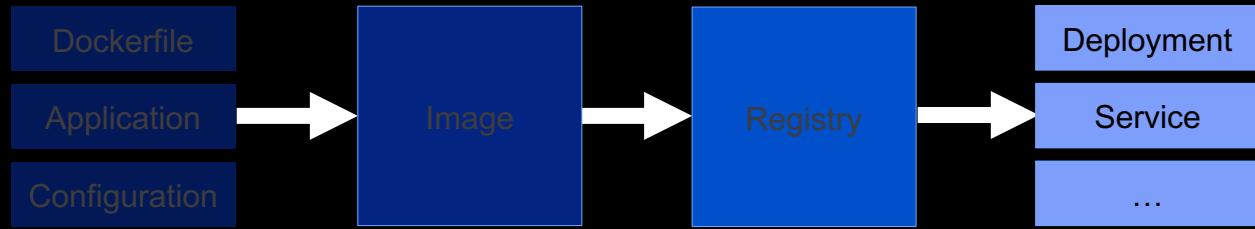
Pod

Minimum components you need:

- Deployment
- Service



Minimal Application – Kubernetes Manifests



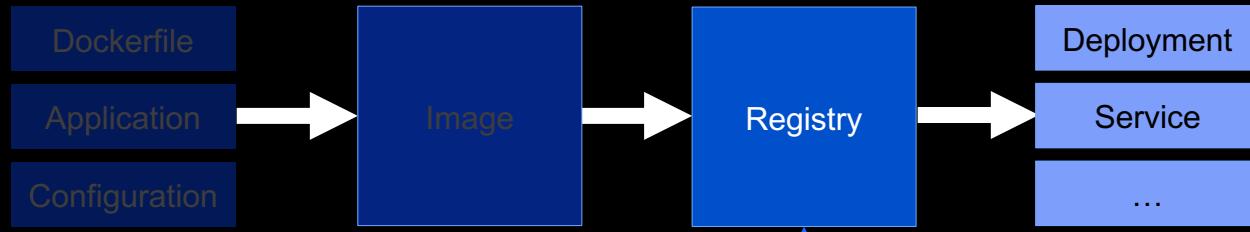
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
template:
  metadata:
    labels:
      app: k8s-demo
spec:
  containers:
    - name: k8s-demo
      imagePullPolicy: Always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

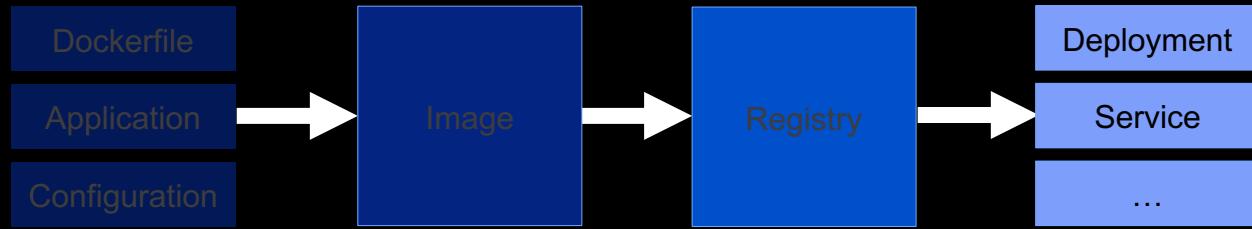
- Deployment
- Service

imagePullPolicy:

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



Minimal Application – Kubernetes Manifests



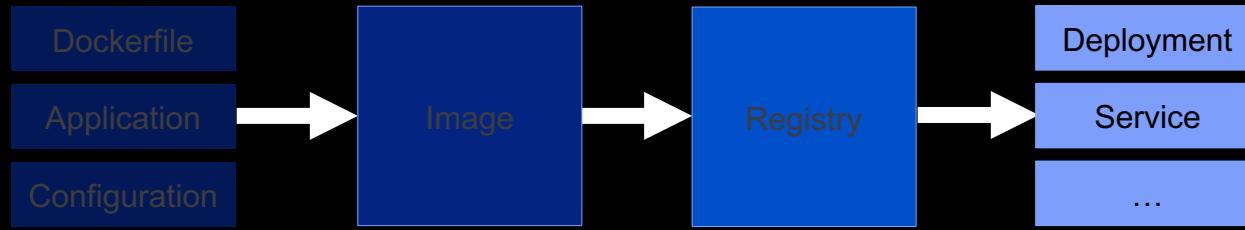
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

Minimum components you need:

- Deployment
- Service



Minimal Application – Kubernetes Manifests

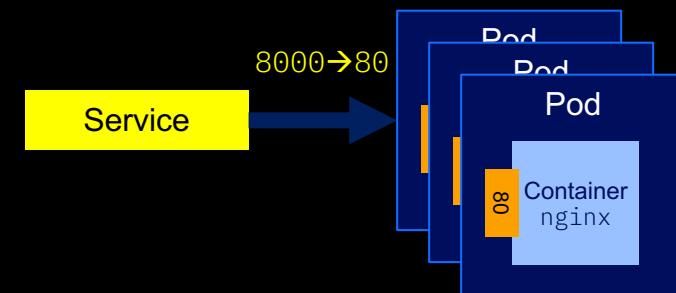


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
         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

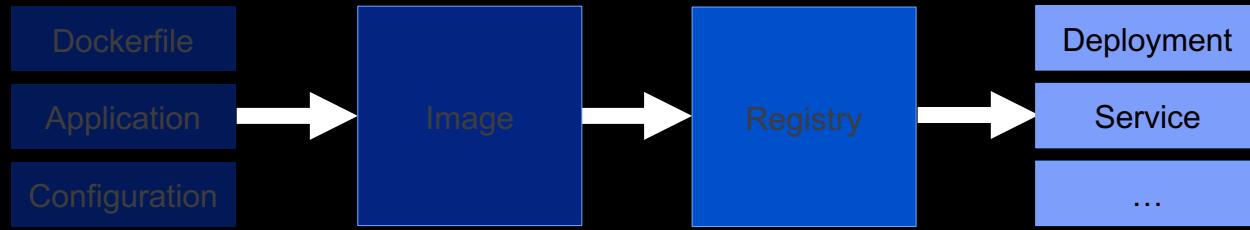
```
apiVersion: v1
kind: Service
metadata:
  name: k8s-demo-service
spec:
  ports:
    - port: 8000
      targetPort: 80
      protocol: TCP
    selector:
      app: k8s-demo
```

Minimum components you need:

- Deployment
- Service



Minimal Application – Kubernetes Manifests

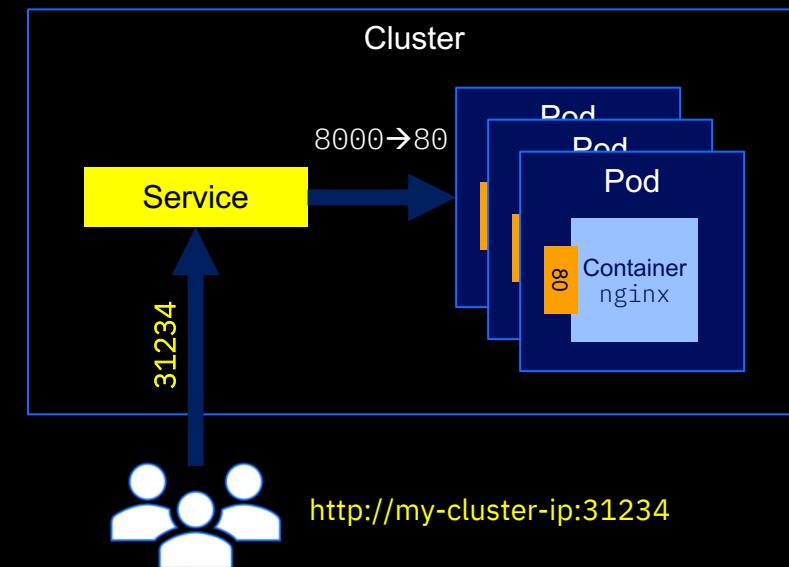


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-demo-deployment
  labels:
    app: k8s-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: k8s-demo
  template:
    metadata:
      labels:
        app: k8s-demo
    spec:
      containers:
        - name: k8s-demo
          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
```

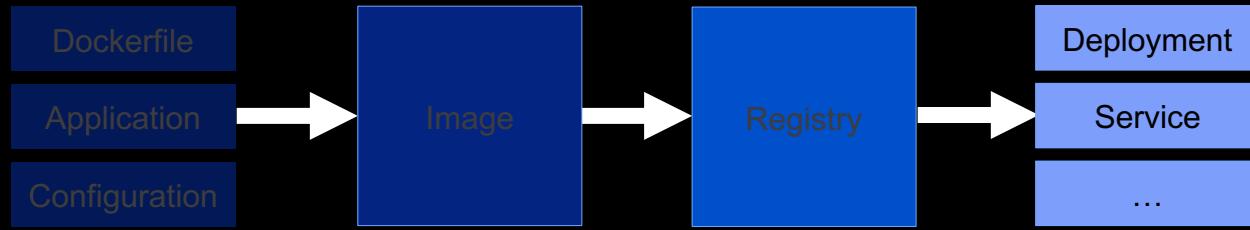
Minimum components you need:

- Deployment
- Service





Minimal Application – Kubernetes Manifests



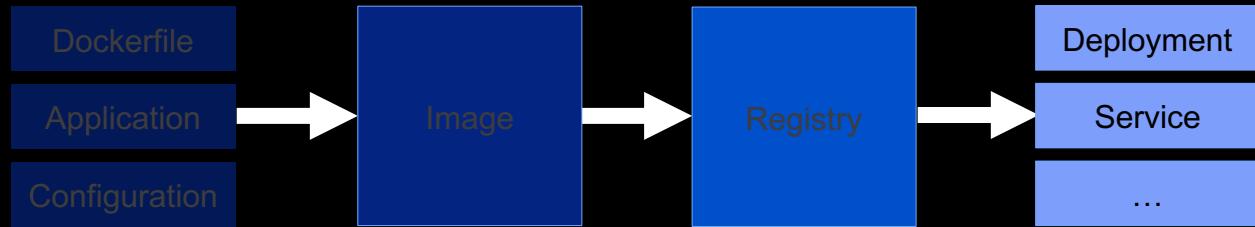
```
apiVersion: apps/v1
kind: Deployment
metadata:
...
spec:
...
template:
...
resources:
  requests:
    memory: 60Mi
    cpu: 10m
  limits:
    memory: 500Mi
    cpu: 100m
```

Minimum components you need:

- Deployment
- Service



Minimal Application – Kubernetes Manifests

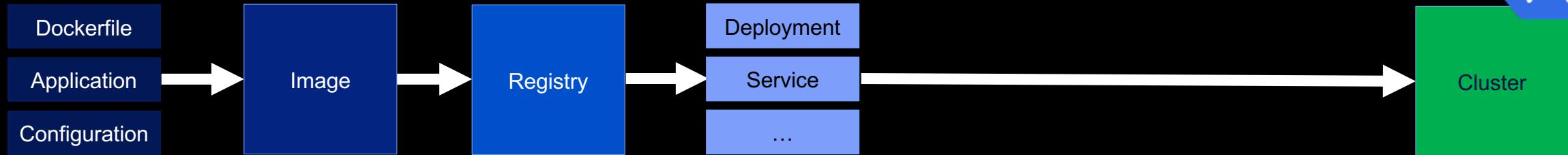


```
apiVersion: apps/v1
kind: Deployment
metadata:
...
spec:
...
template:
...
env:
- name: PORT
  value : "3000"
- name: APPLICATION_NAME
  value: k8sdemo
- name: BACKEND_URL
  value: http
```

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
```

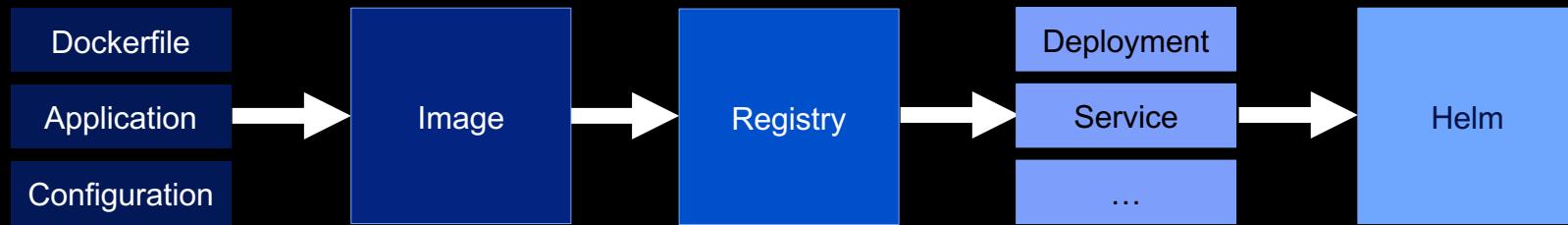
Minimum components you need:

- 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

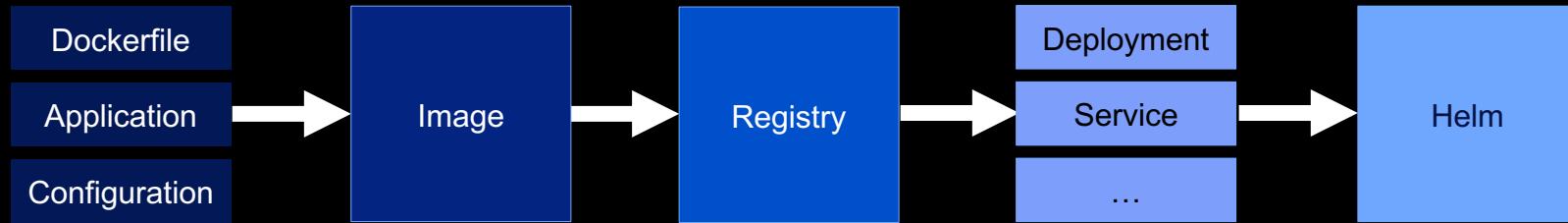
Templates (yaml)

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

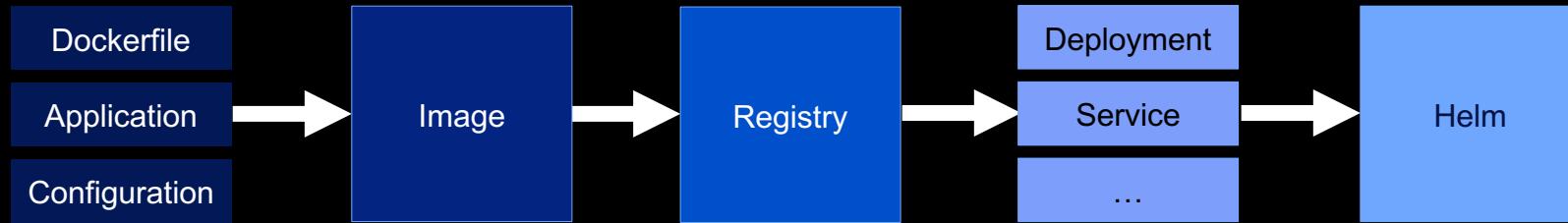
Templates (yaml)

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

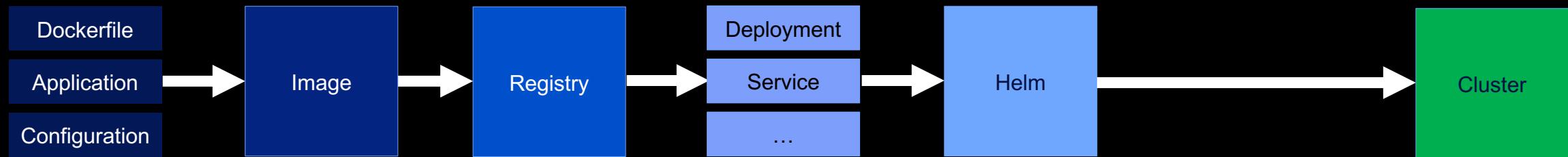
Templates (yaml)

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
  
```

```

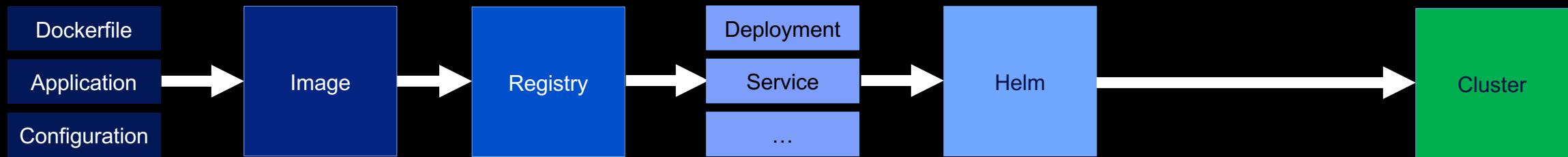
apiVersion: v1
image:
  name:
    - niklaushirt/k8s-demo
  tag: 1.0
  pullPolicy: IfNotPresent
replicas: 1
  
```

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
```

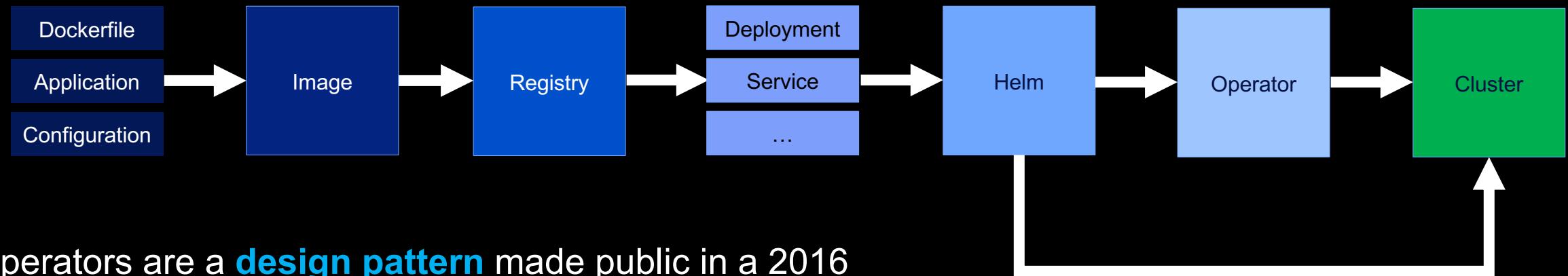
```
apiVersion: v1
image:
  name:
    - niklaushirt/k8s-demo
  tag: 1.0
  pullPolicy: IfNotPresent
replicas: 1
```

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



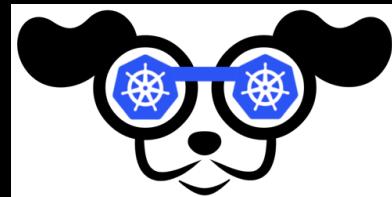
Operators are a **design pattern** made public in a 2016 CoreOS [blog](#) post.

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



A screenshot of a terminal window titled "K9s: Norton Commander for Kubernetes". The window shows a list of pods in the "default" namespace. The columns include NAME, READY, STATUS, RESTARTS, CPU, MEM, NODE, and AGE. A context menu is open over the first pod, listing options like "ctrl-d", "Delete", "Describe", "Edit", "ctrl-k", "Kill", "Logs", "Logs Previous", and "Shell". The background of the terminal has a faint, abstract blue and yellow circular pattern.

NAME	READY	STATUS	RESTARTS	CPU	MEM	NODE	AGE
dashboard-78d6cfd49-mpjv2	1/1	Terminated	0	0	21	10.94.80.44	6m18s
dashboard-78d6cfd49-wrwez	1/1	Running	0	n/a	n/a	10.94.80.49	7s
kubetoy-deployment-866fc8c587-gv9v3	1/1	Running	23	0	16	10.94.80.49	39d
nginx-policy-deployment-94d74c6bf-qqnfr	1/1	Running	0	0	1	10.94.80.49	5d14h
nginx-policy-deployment-94d74c6bf-t6mrw	1/1	Running	0	0	1	10.94.80.27	8d14h
openldap-94d6cf788-fw7lm	1/1	Running	0	0	19	10.94.80.44	43d
openldap-admin-78cfca0d5-bt6dp	1/1	Running	0	0	83	10.94.80.44	43d
student-vi-48c59a4c64-vcsd5	1/1	Running	0	0	21	10.94.80.49	27d

Preinstalled in the training VM

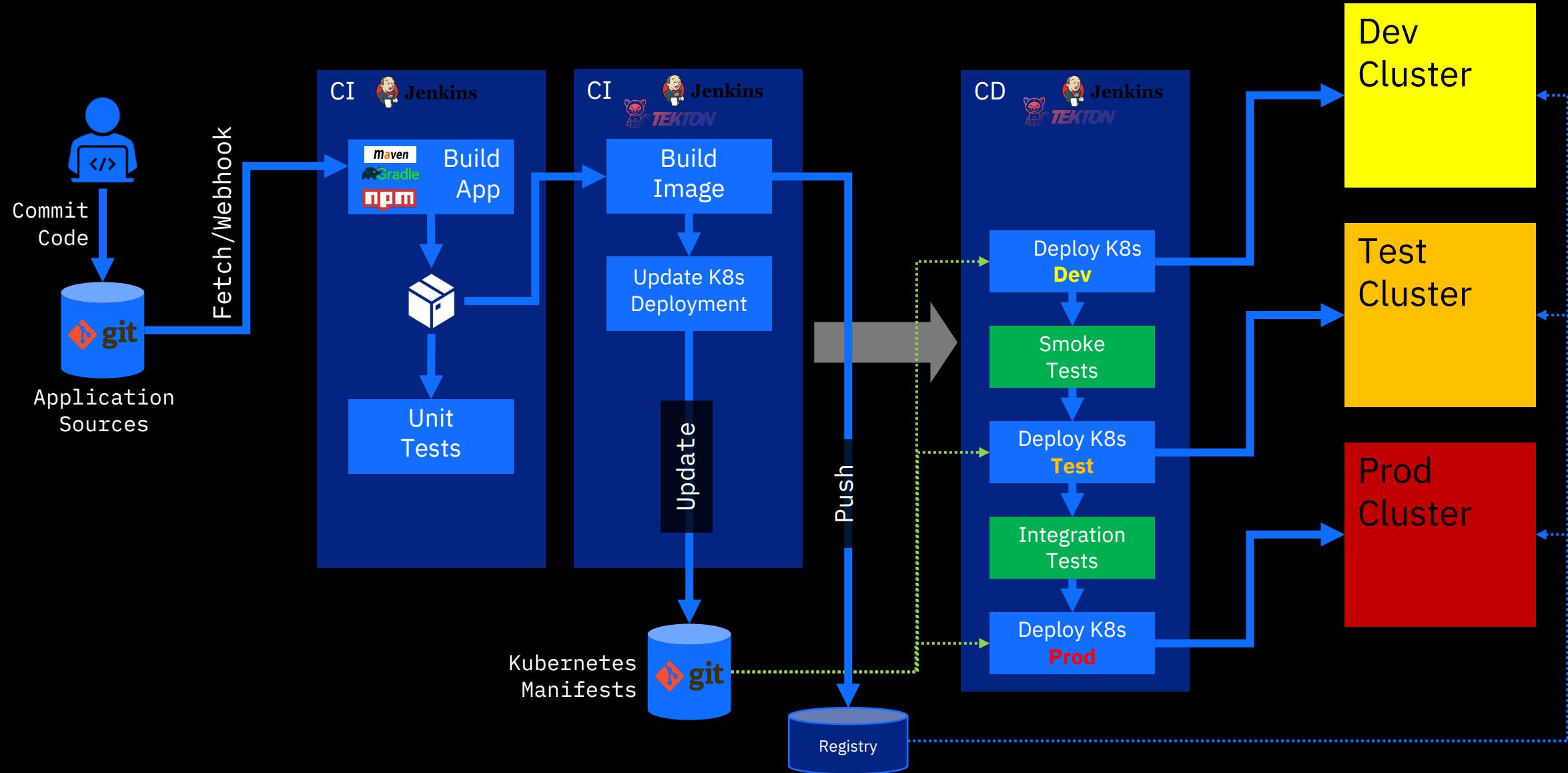
For dumb people like me
(who just don't get vi)

`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



QUESTIONS?



QUIZZ!!!

<https://kahoot.it/>





How do we secure this?

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



More on
this next week.....

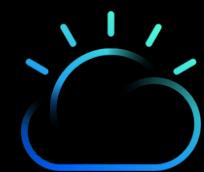




Kubernetes Workshop Series
Kubernetes - Hands-On

04





Starting Course JTC02 Kubernetes

Name will be shown



The screenshot shows a user interface for selecting a course. At the top, there is a header with a cloud icon and the text "Collector - Niklaus-Hirt". Below the header, there are tabs for "Courses", "Class work", "Statistics", "Information", and "Feedback". A large button labeled "Catalog of courses" is visible. On the left, there is a dropdown menu with the placeholder "select course" and a list of course names. On the right, there is a button labeled "Begin course". A red box highlights the "Begin course" button, and a red arrow points from the text "Select course and press button to begin" to it.

- select course
- select course
- JTC01 Docker
- JTC02 Kubernetes Labs
- JTC10 Istio
- JTC14 Kubernetes Ansible Operators Labs
- JTC16 Kubernetes Security Labs
- JTC17 Kubernetes Advanced Security Labs
- JTC80 Kubernetes Introduction
- JTC90 Lab Setup

Current course catalog

Select course and
press button to begin



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



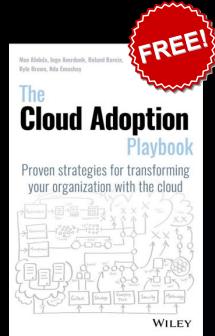
READY
SET
GO!!!!

Duration: 60 mins

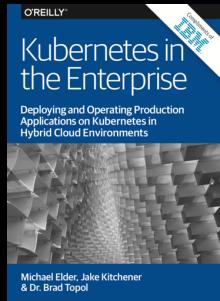
QUESTIONS?



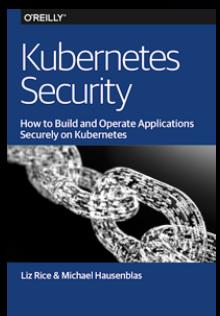
Kubernetes – Some Reading Tips



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/>



Sources and documentation will be available here:

https://github.com/niklaushirt/k8s_training_public

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

See you next week!

- **Same place**
- **Same time**

Kubernetes Workshop
Series

**Kubernetes
Security Basics**



Niklaus Hirt



nikh@ch.ibm.com



@nhirt



THANK YOU!!!!