loodse

Kubernetes for Developers

From Beginner to Expert

loodse

Get the slides!

http://bit.ly/loodse-CHANGEME

loodse

Agenda

- Containers
- Kubernetes
- A Pod
- Labels
- Namespaces
- Replication Controller
- Deployments
- Jobs
- kubectl
- Graceful Termination
- Scheduling
- Ressourcen
- Authenticating
- Admission Controller
- Managing State
- Configmaps
- Secrets
- Ingress
- Network Policies
- Node Drain
- Network Plugins
- More…
- Cluster Addons
- CNCF Landscape
- ...

loodse

Our Agenda

Day I

Introduction
- Introduction Round
  - Participants
  - Backgrounds
  - Expectations
- Lab Environments / Tools

Container Fundamentals
- Why Containers?
- Architecture of Containers: Linux, Namespaces, cgroups
- What is Docker?
- Dev/Ops Impact

Container Hands-On
- Installation
- Images
- Container: Handling and Interaction
- Dockerfiles
- Image Builds
- Advanced Dockerfiles
- Networking
- Volumes
- Tips & Tricks
- Labs

Day II

Kubernetes Introduction
- Kubernetes Introduction
  - FOSS and Community
  - K8s architecture and principles
- Preparing for Kubernetes
  - CNCF landscape
  - Tools / Environment
- kubectl: The Kubernetes CLI

Kubernetes Architecture
- Kubernetes Architecture
- API Objects
- API Access
- Cluster Setups

The Kubernetes Resources
- Pods, RC, Deployments
- DaemonSets, StatefulSets
- Jobs / CronJobs
- Limits / Namespaces

Deploy $things
- Dashboard (user + proxy)
- prometheus
- cAdvisor daemonset
- metrics server

Services
- Accessing Services
- DNS
- Labels & Selectors

Day III

Scheduling
- Behind the scenes
- Draining
- Affinities / Taints / Tolerations
- Advanced scheduling

Managing State
- Persistent Volumes
- ConfigMaps
- Secrets

Ingress
- Concept & API
- Ingress Controller and Rules
- Labs

Security
- Authentication and Authorization
- Admission Controller
- Pod & Network Policies

Cluster Addons
- Overlay Networks
- Monitoring & Logging
- DNS
- Helm

Scaling
- Metrics and HPA
- Eviction
- Service Discovery
- CRD / extending

(Backup)

Setting up Kubernetes
- Getting Started With Kubernetes
- Minikube
- kubeadm
- ClusterAPI
- More Installation Tools

Tooling - Tips & Tricks
- kubectl cheat sheet
- fubectl
- kind
- …

Training Environment

- Katacoda
- Google Kubernetes Engine (GKE)
  - Cloud Shell
  - kubeconfig
- Docker
- kubectl
- Training Notes editable Google Doc

loodse

Containers

A quick recap of containers

- Lightweight
- Hermetically sealed
- Isolated
- Easily deployable
- Introspectable
- Runnable

Linux processes

- Improves overall developer experience
- Fosters code and component reuse
- Simplifies operations for cloud native applications

loodse

Containers are awesome!

Let’s run lots of them!

Everything at Google runs in containers….

> 2 billion containers per week

- Gmail, Web Search, Maps, ...
- MapReduce, batch, ...
- GFS, Colossus, …
- Even **Google's Cloud Platform:** VMs run in containers!

loodse

Kubernetes Introduction

loodse

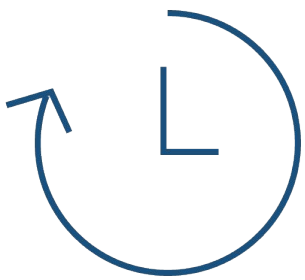
Why should we use Cloud Native technologies?

loodse

Modern Business Challenge

Business wants more with less

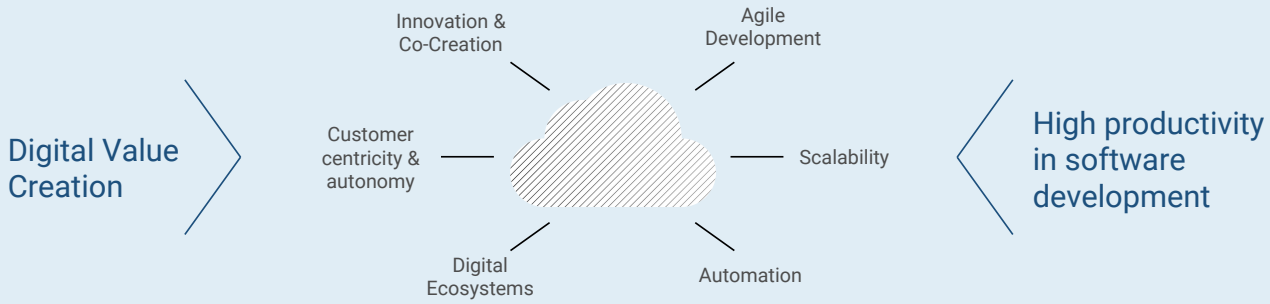
- Increased expectations from business to deliver new functionality **faster**
- Pressure from the business to **reduce cost**



How can we do more with less?

loodse

Cloud native technologies drive the digital transformation



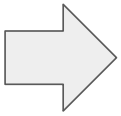
Data Security, IT-Security & Architecture fulfill legal requirements

loodse

Why Change?

Business expectations become IT challenges

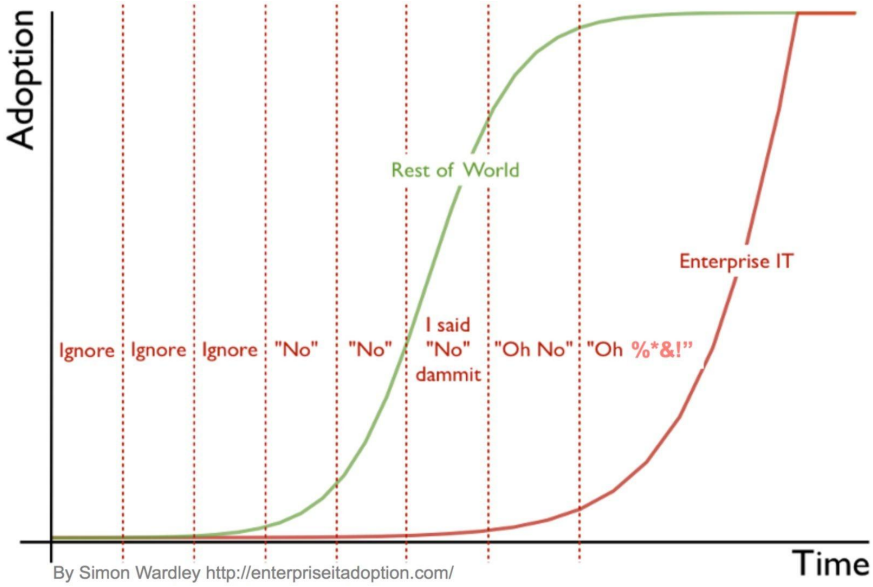
100%
of resources



70%
to maintain portfolio

30%
to invest in innovation, differentiation

A Look At Cloud Adoption



loodse

Why Does It Happen?

- Enterprise IT exists to support business mission
  - IT is a cost center, and enabler.
- For emerging tech, ROI is usually unknown in the beginning
  - Most enterprises adopt at significant maturity
- Organizational inertia
  - Skill Sets
  - Processes
  - Existing systems

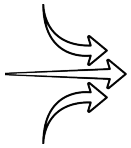
A business view

Enable enterprise to add value at internet scale

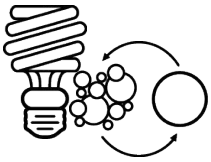
Re-balance maintenance and innovation

Reduce / vendor lock-in, license models

Become more productive with lightweight tech



Remove technical debt & risk

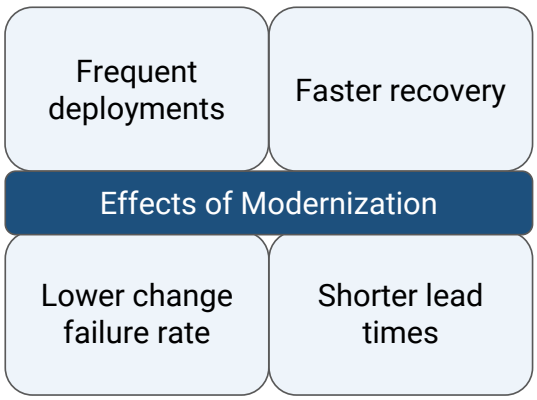


Adopt agile methodologies, DevOps, or cloud

loodse

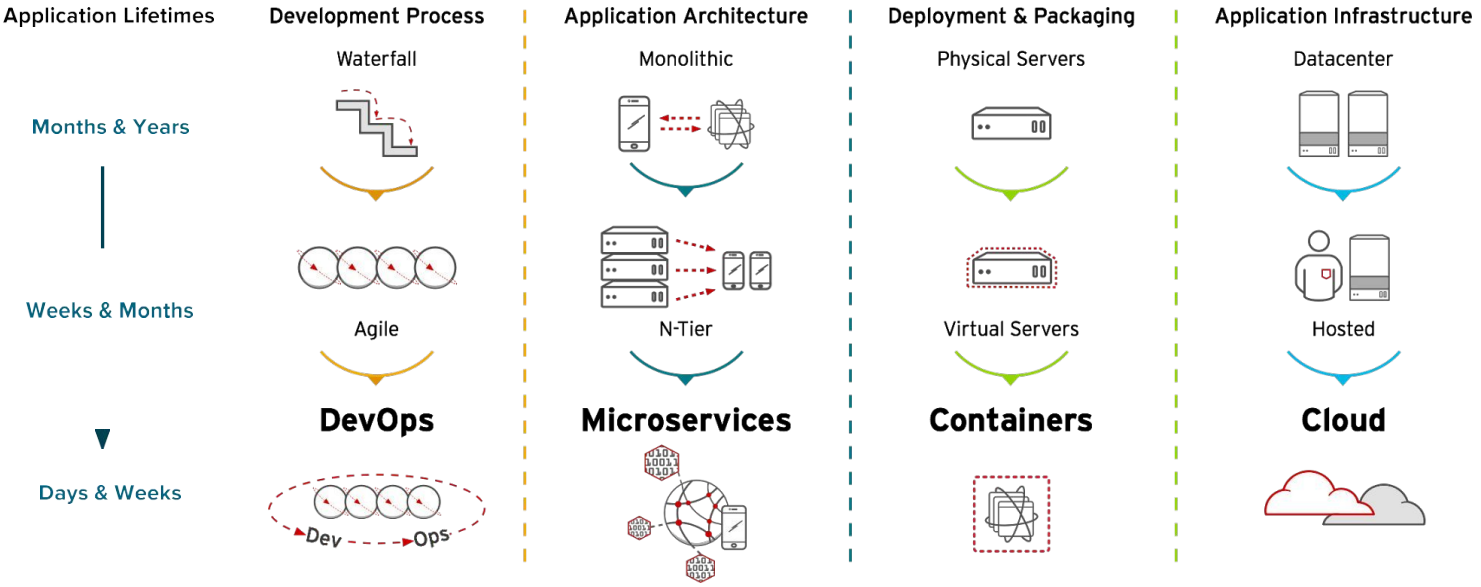
Why Modernization?

- Enable experimental approach to product development
- Create high performance teams
  - Improve quality of work
  - Drive loyalty
- Overall improvement in quality



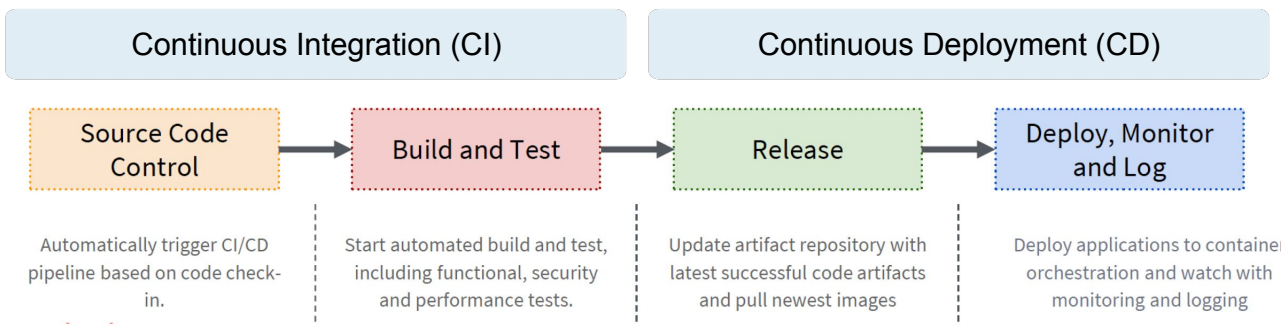
loodse

IT Must Evolve to Stay Ahead of Demands



loodse

How we bring the App to production on Kubernetes?

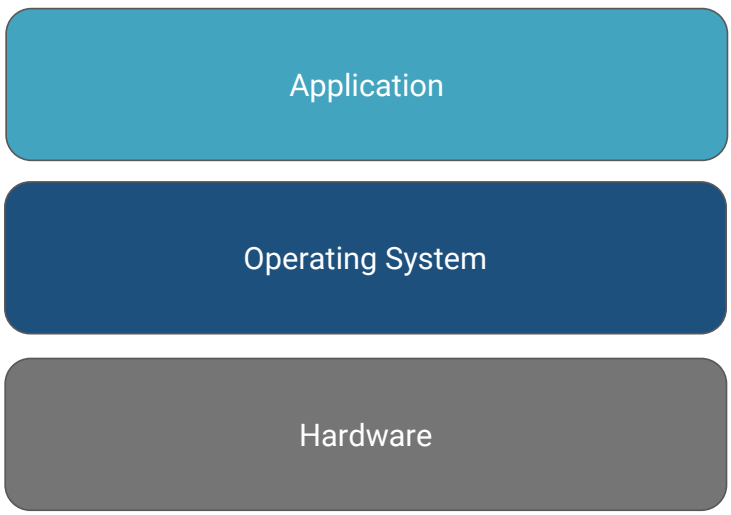


loodse

From Monoliths to Containers and Kubernetes

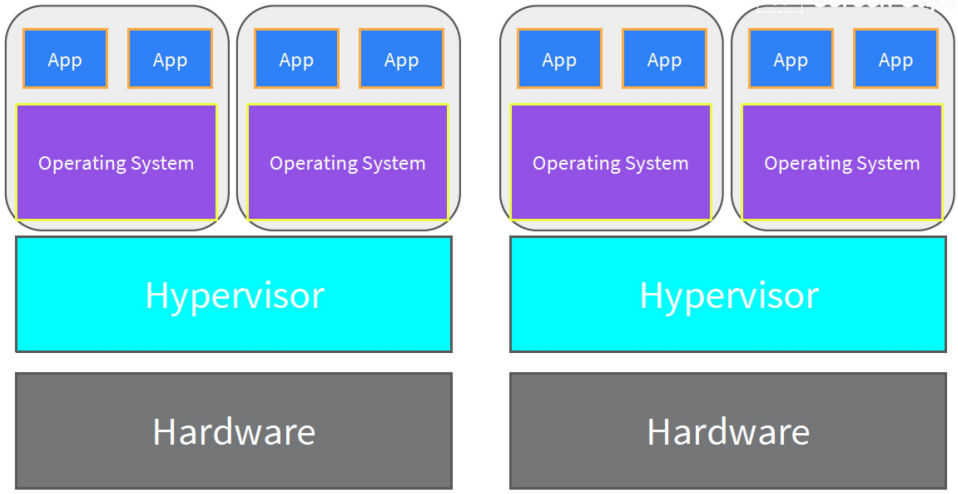
loodse

Monoliths



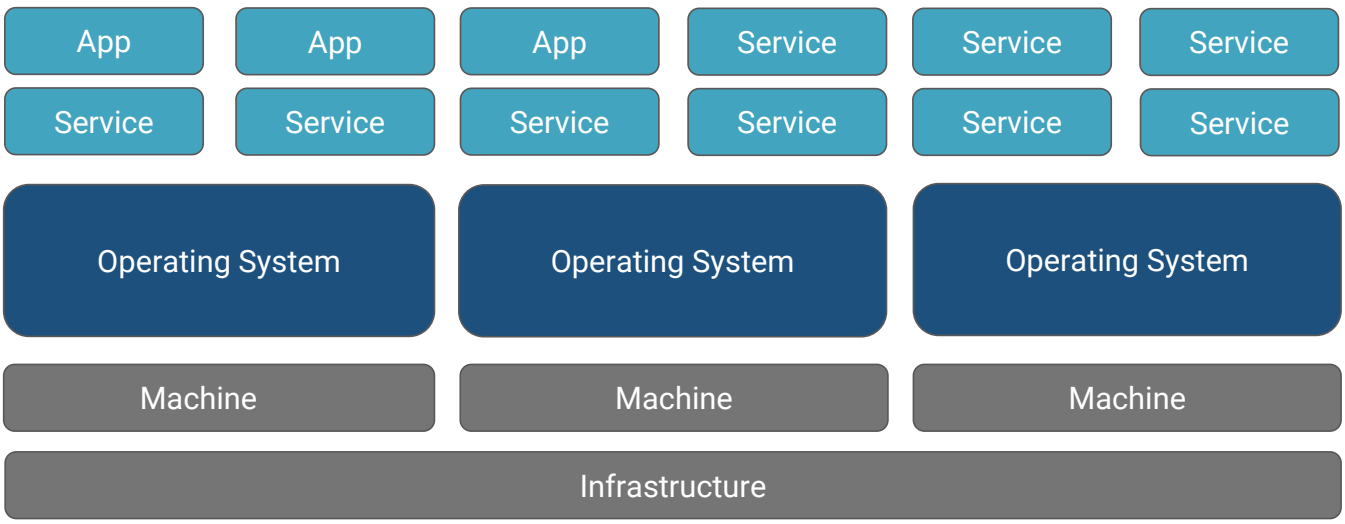
loodse

Virtualisation



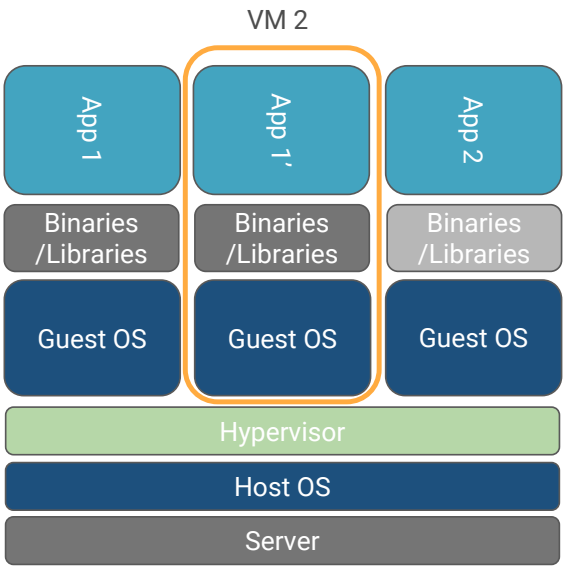
loodse

Microservices

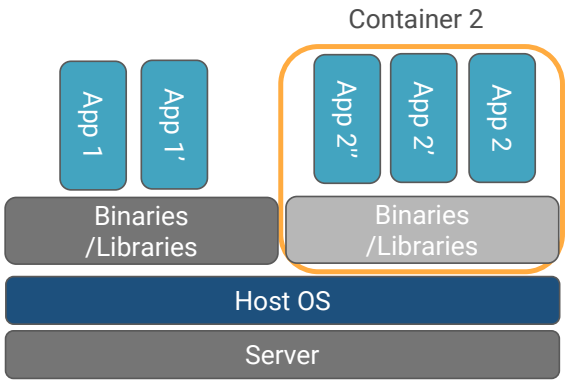


loodse

VMs

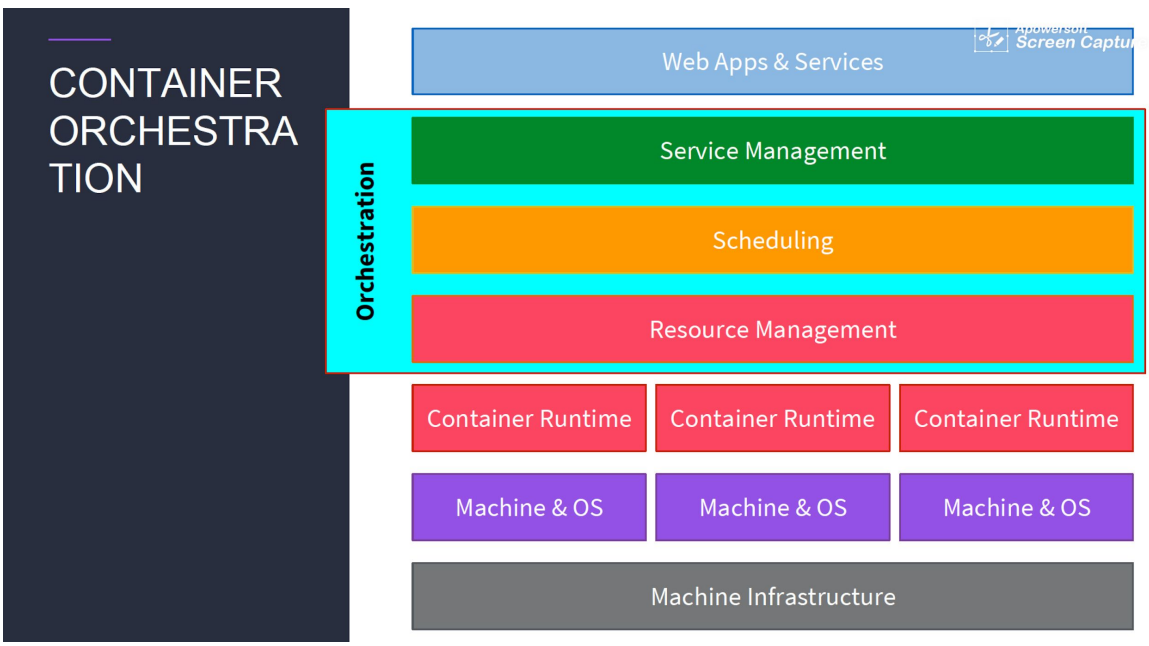


Container



loodse

How to manage all the Containers?



loodse

Kubernetes to the rescue!

loodse

Kubernetes

Greek for “Helmsman”

also the root of the words “governor” and “cybernetic”
the German “Lotse” (which in old German/Platt is “Loodse”.)

- Infrastructure for containers
- Schedules, runs, and manages containers on virtual and physical machines
- Platform for automating deployment, scaling, and operations
- Inspired and informed by Google’s experiences and internal systems
- 100% Open source, written in Go

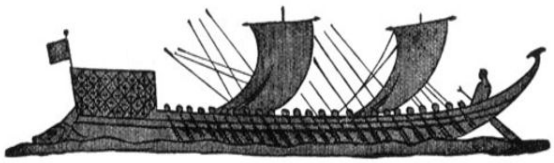


Image Source

loodse

What is the problem to solve?

- We are going to talk about a solution…
- Can we agree on a problem first?

A problem can be defined as any situation in which a gap is perceived to exist between what is and what should be.
- Arthur B. Van Gundy Jr.

loodse

What is the problem to solve?

- We are going to talk about a solution…
- Can we agree on a problem first?

*A problem can be defined as any situation in which a gap is perceived to exist between **what is** and **what should be**.*
Arthur B. Van Gundy Jr.

This! Exactly!

"Actual State" → Reconciliation → "Desired State"

What Does Kubernetes do?

- Known as the linux kernel (or POSIX) of distributed systems.
- Abstracts away the underlying hardware of the nodes and provides a uniform interface for workloads to be both deployed and consume the shared pool of resources.
- Works as an engine for resolving state by converging actual and the desired state of the system.

loodse

Decouples Infrastructure and Scaling

- All services within Kubernetes are natively Load Balanced.
- Can scale up and down dynamically.
- Used both to enable self-healing and seamless upgrading or rollback of applications.

Self Healing

Kubernetes will **ALWAYS** try and steer the cluster to its desired state.

Me: “I want 3 healthy instances of redis to always be running.”

Kubernetes: “Okay, I’ll ensure there are always 3 instances up and running.”

Kubernetes: “Oh look, one has died. I’m going to attempt to spin up a new one.”

loodse

Most Importantly...

Use the **SAME** API
across bare metal and **EVERY** cloud provider!!!

Want to automate orchestration for velocity & scale

Diverse workloads and use cases demand still more functionality

- Rolling updates and blue/green deployments
- Application secret and configuration distribution
- Continuous integration and deployment
- Workflows
- Batch processing
- Scheduled execution
- Application-specific orchestration

…

A composable, extensible **Platform** is needed

loodse

Cutting the cord: container-centric infrastructure

Once specific containers are no longer bound to specific machines,

host-centric infrastructure no longer works

- **Scheduling**: Decide where my containers should run
- **Lifecycle and health**: Keep my containers running despite failures
- **Scaling**: Make sets of containers bigger or smaller
- **Naming and discovery**: Find where my containers are now
- **Load balancing:** Distribute traffic across a set of containers
- **Storage volumes**: Provide data to containers
- **Logging and monitoring**: Track what's happening with my containers
- **Debugging and introspection**: Enter or attach to containers
- **Identity and authorization**: Control who can do things to my containers

loodse

kubectl: The Kubernetes CLI

loodse

Main tool to interact with Kubernetes: `kubectl`

How to install?
⇒ visit: https://kubernetes.io/docs/tasks/tools/install-kubectl

```
sudo apt-get update && sudo apt-get install -y apt-transport-https

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a
/etc/apt/sources.list.d/kubernetes.list

sudo apt-get update
sudo apt-get install -y kubectl
```

Deployment

```
$ kubectl run my-nginx --image=nginx
replicationcontroller "my-nginx" created

$ kubectl get po
NAME              READY     STATUS    RESTARTS   AGE
my-nginx-wepbv    1/1       Running   0          1m
```

loodse

Scaling

```
$ kubectl scale deployment my-nginx --replicas=2
deployment "my-nginx" scaled

$ kubectl get pod
NAME              READY     STATUS    RESTARTS   AGE
my-nginx-wepbv    1/1       Running   0          1m
my-nginx-yrf3u    1/1       Running   0          20s
```

Shutdown

```
$ kubectl delete deployment my-nginx
deployment "my-nginx" deleted

$ kubectl get pod
NAME              READY     STATUS        RESTARTS   AGE
my-nginx-wepbv    0/1       Terminating   0          4m
my-nginx-yrf3u    0/1       Terminating   0          3m

$ kubectl get pod
```

loodse

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/courses/kubernetes/kubernetes-01-playground

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/training/siemens-k8s/kubernetes-01-playground

loodse

FOSS and Community

Open Source Model

Linux Foundation
www.linuxfoundation.org

- Nonprofit consortium dedicated to fostering the growth of Linux
- Promotes Linux and provides neutral collaboration and education
- Protects and supports Linux development
- Improves Linux as a technical platform
- Sponsors the work of Linux creator Linus Torvalds
- Supported by leading technology companies and developers

Cloud Native Computing Foundation
www.cncf.io

- Sub Organisation of the Linux Foundation
- CNCF is not just about Kubernetes
- Foundation serves as the governing body for open source software
- Solves specific issues faced by Cloud native applications (i.e applications that are written specifically for a Cloud environment).

loodse

Open Source Model

- Kubernetes is open source software using an Apache license v2
  ⇒ https://github.com/kubernetes/kubernetes/blob/master/LICENSE
- Google donated Kubernetes to a newly formed collaborative Linux Foundation project (CNCF) within the in July 2015
  ⇒ Kubernetes reached the v1.0 release.
- CNCF now owns the Kubernetes copyright
- Contributors to the source need to sign a contributor license agreement (CLA)
  like any contributor to an Apache-licensed project to the Apache Software Foundation
  ⇒ https://github.com/kubernetes/community/tree/master/contributors/guide

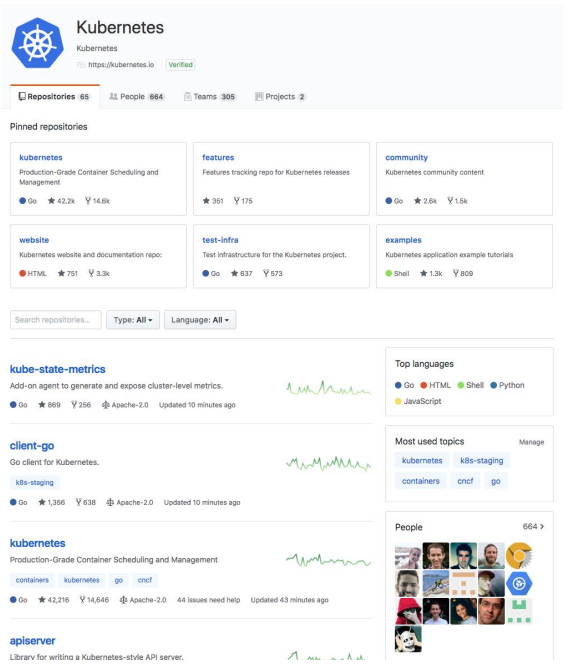
loodse

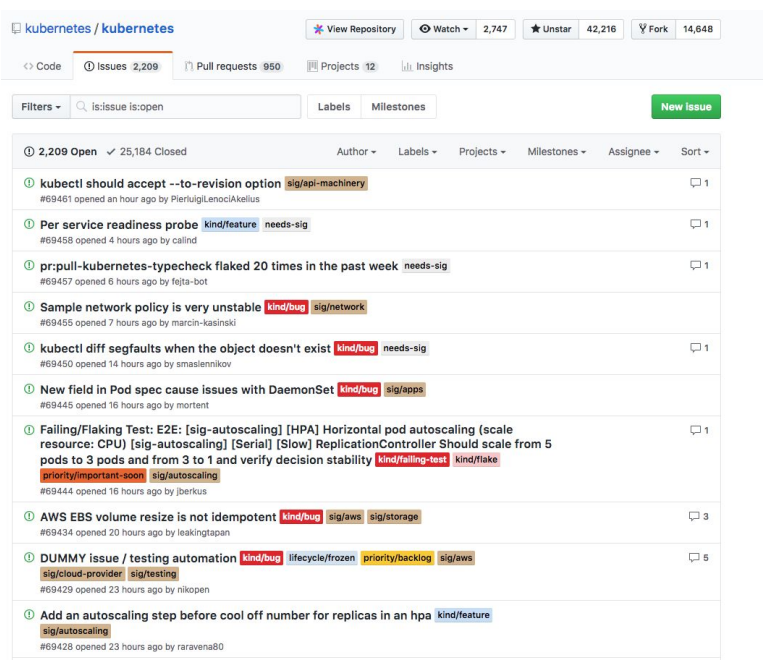
Kubernetes is Open

- open community
- open source
- open design
- open to ideas

loodse

GitHub

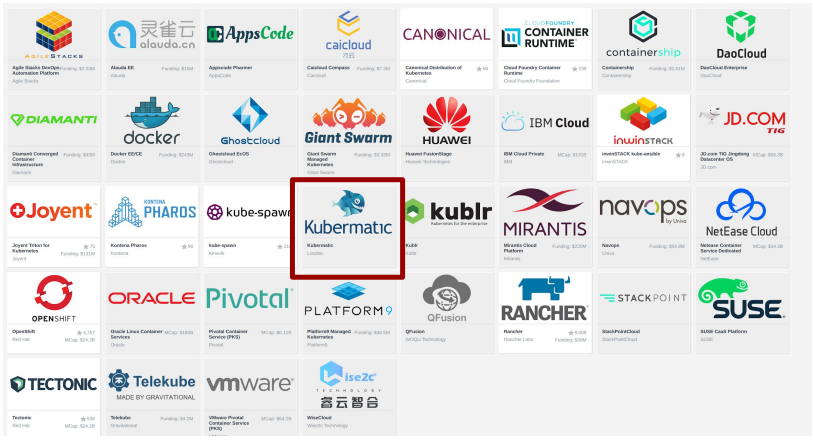




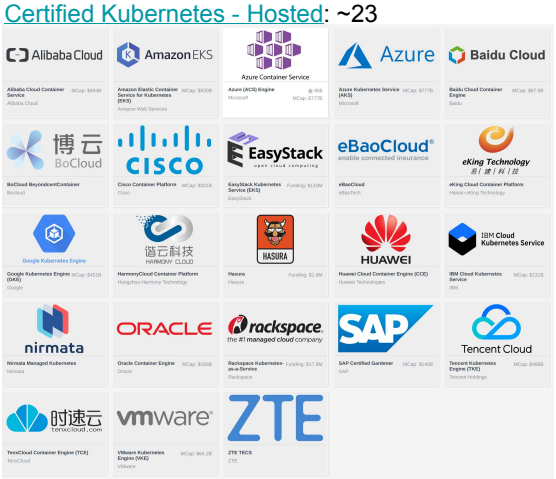
loodse

CNCF Landscape ⇒ The Kubernetes Universe

Certified Kubernetes - Distribution: ~36



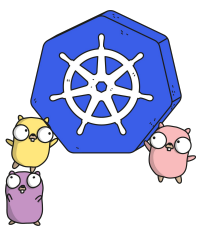
Certified Kubernetes - Hosted: ~23

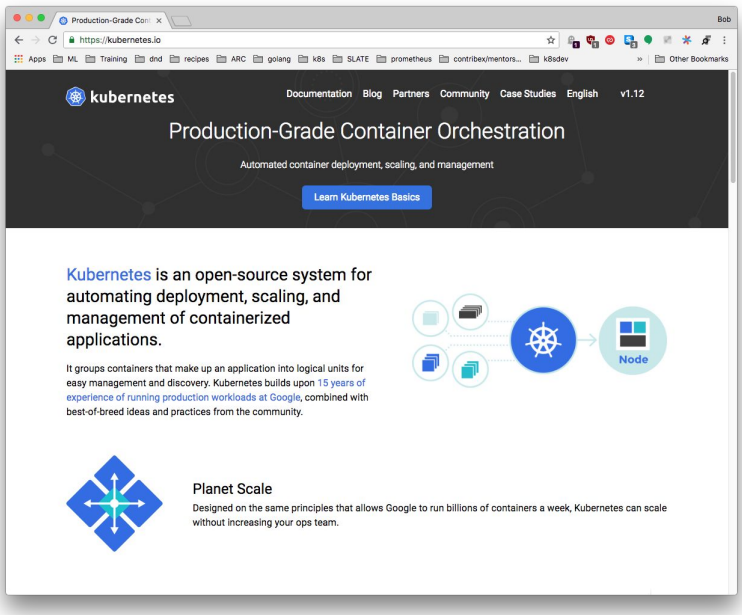


loodse

kubernetes.io

- Documentation
- Examples
- API Reference





loodse

Slack

Kubernetes

slack.k8s.io

50,000+ users

160+ public channels

CNCF

slack.cncf.io

4,200+ users

70+ public channels

10/2/2018

loodse

Other Communities

- **Official Forum**
  https://discuss.kubernetes.io
- **Subreddit**
  https://reddit.com/r/kubernetes

- **StackOverflow**
  https://stackoverflow.com/questions/tagged/kubernetes

loodse

Meetups

Kubernetes

meetup.com/topics/kubernetes/

620+ groups

250,000+ members

CNCF

meetups.cncf.io

155+ groups

80,000+ members

36+ countries

10/2/2018

loodse

Conventions

Europe:
June 24 - 26, 2019
Hamburg, Germany

KubeCon | CloudNativeCon

Europe:
May 21 – 23, 2019
Barcelona, Spain

North America:
November 18 - 21, 2019
San Diego, CA

Asia:
June 25 – 26, 2019
Shanghai, China

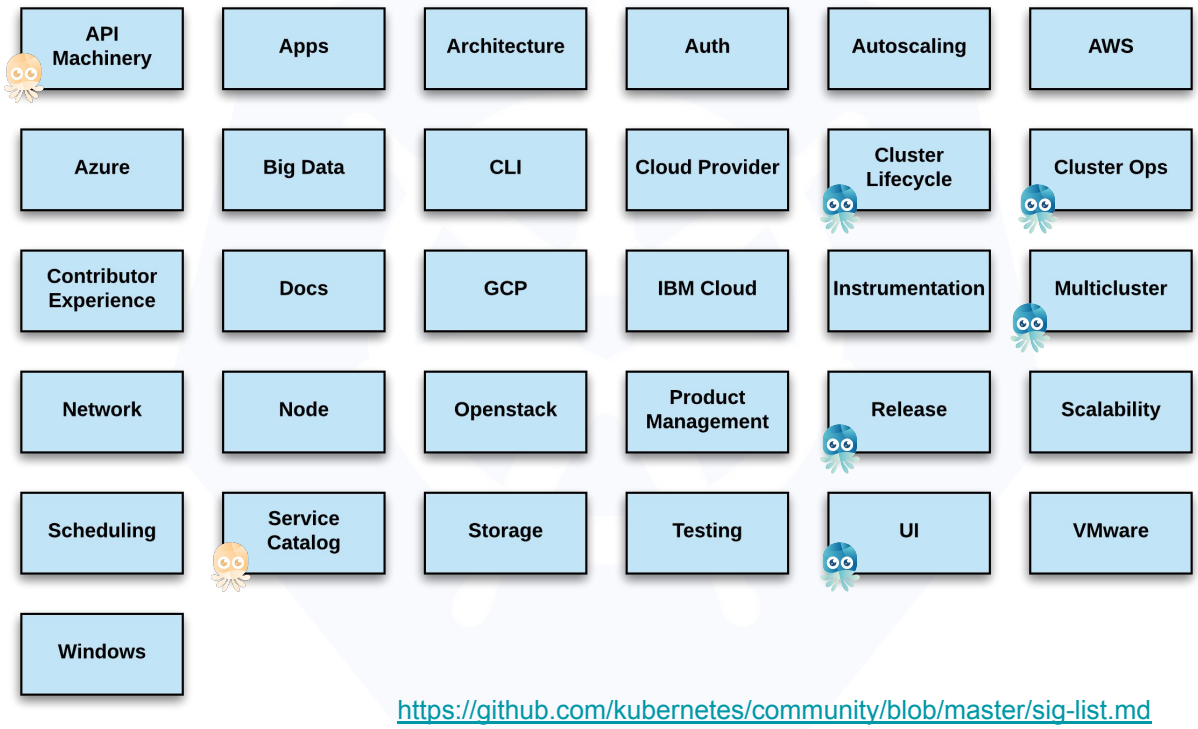
loodse

SIGs

- Kubernetes components and features are broken down into smaller self-managed communities known as **Special Interest Groups** (**SIG**).
- Hold weekly public recorded meetings and have their own mailing lists and slack channels.

loodse

SIG List



https://github.com/kubernetes/community/blob/master/sig-list.md

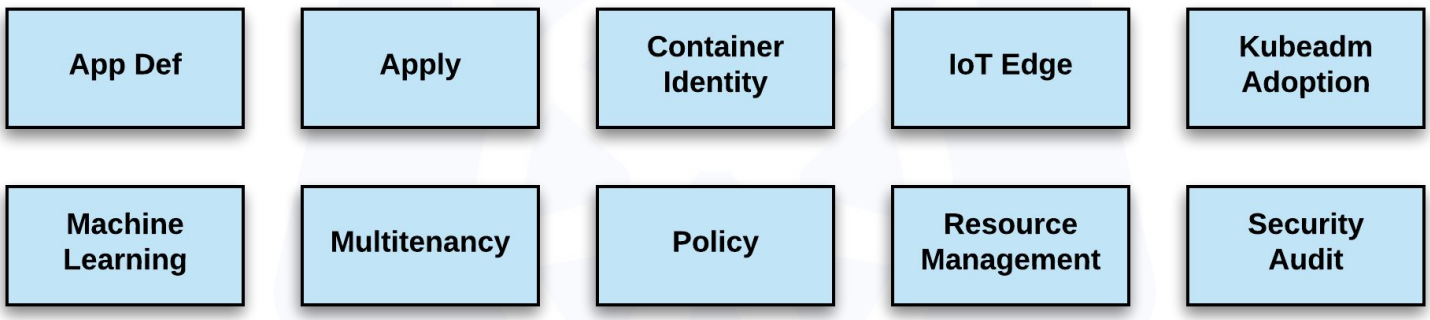
loodse

Working Groups

- Similar to SIGs, but are topic focused, time-bounded, or act as a focal point for cross-sig coordination.
- Hold scheduled publicly recorded meetings in addition to having their own mailing lists and slack channels.

loodse

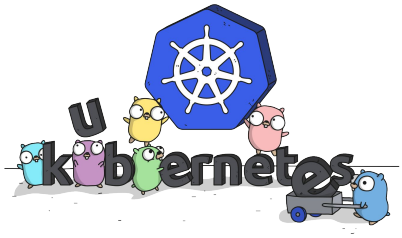
WG List



https://github.com/kubernetes/community/blob/master/sig-list.md

loodse

Links



- **Free Kubernetes Courses**
  https://www.edx.org/
- **Interactive Kubernetes Tutorials**
  https://www.katacoda.com/courses/kubernetes
- **Learn Kubernetes the Hard Way**
  https://github.com/kelseyhightower/kubernetes-the-hard-way
- **Official Kubernetes Youtube Channel**
  https://www.youtube.com/c/KubernetesCommunity
- **Official CNCF Youtube Channel**
  https://www.youtube.com/c/cloudnativefdn
- **Track to becoming a CKA/CKAD (Certified Kubernetes Administrator/Application Developer)**
  https://www.cncf.io/certification/expert/
- **Awesome Kubernetes**
  https://ramitsurana.gitbooks.io/awesome-kubernetes/content/

loodse

Our Agenda

Day I

Introduction
- Introduction Round
  - Participants
  - Backgrounds
  - Expectations
- Lab Environments / Tools

Container Fundamentals
- Why Containers?
- Architecture of Containers: Linux, Namespaces, cgroups
- What is Docker?
- Dev/Ops Impact

Container Hands-On
- Installation
- Images
- Container: Handling and Interaction
- Dockerfiles
- Image Builds
- Advanced Dockerfiles
- Networking
- Volumes
- Tips & Tricks
- Labs

Day II

Kubernetes Introduction
- Kubernetes Introduction
  - FOSS and Community
  - K8s architecture and principles
- Preparing for Kubernetes
  - CNCF landscape
  - Tools / Environment
- kubectl: The Kubernetes CLI

Kubernetes Architecture
- Kubernetes Architecture
- API Objects
- API Access
- Cluster Setups

The Kubernetes Resources
- Pods, RC, Deployments
- DaemonSets, StatefulSets
- Jobs / CronJobs
- Limits / Namespaces

Deploy $things
- Dashboard (user + proxy)
- prometheus
- cAdvisor daemonset
- metrics server

Services
- Accessing Services
- DNS
- Labels & Selectors

Day III

Scheduling
- Behind the scenes
- Draining
- Affinities / Taints / Tolerations
- Advanced scheduling

Managing State
- Persistent Volumes
- ConfigMaps
- Secrets

Ingress
- Concept & API
- Ingress Controller and Rules
- Labs

Security
- Authentication and Authorization
- Admission Controller
- Pod & Network Policies

Cluster Addons
- Overlay Networks
- Monitoring & Logging
- DNS
- Helm

Scaling
- Metrics and HPA
- Eviction
- Service Discovery
- CRD / extending

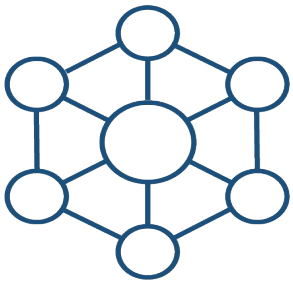
(Backup)

Setting up Kubernetes
- Getting Started With Kubernetes
- Minikube
- kubeadm
- ClusterAPI
- More Installation Tools

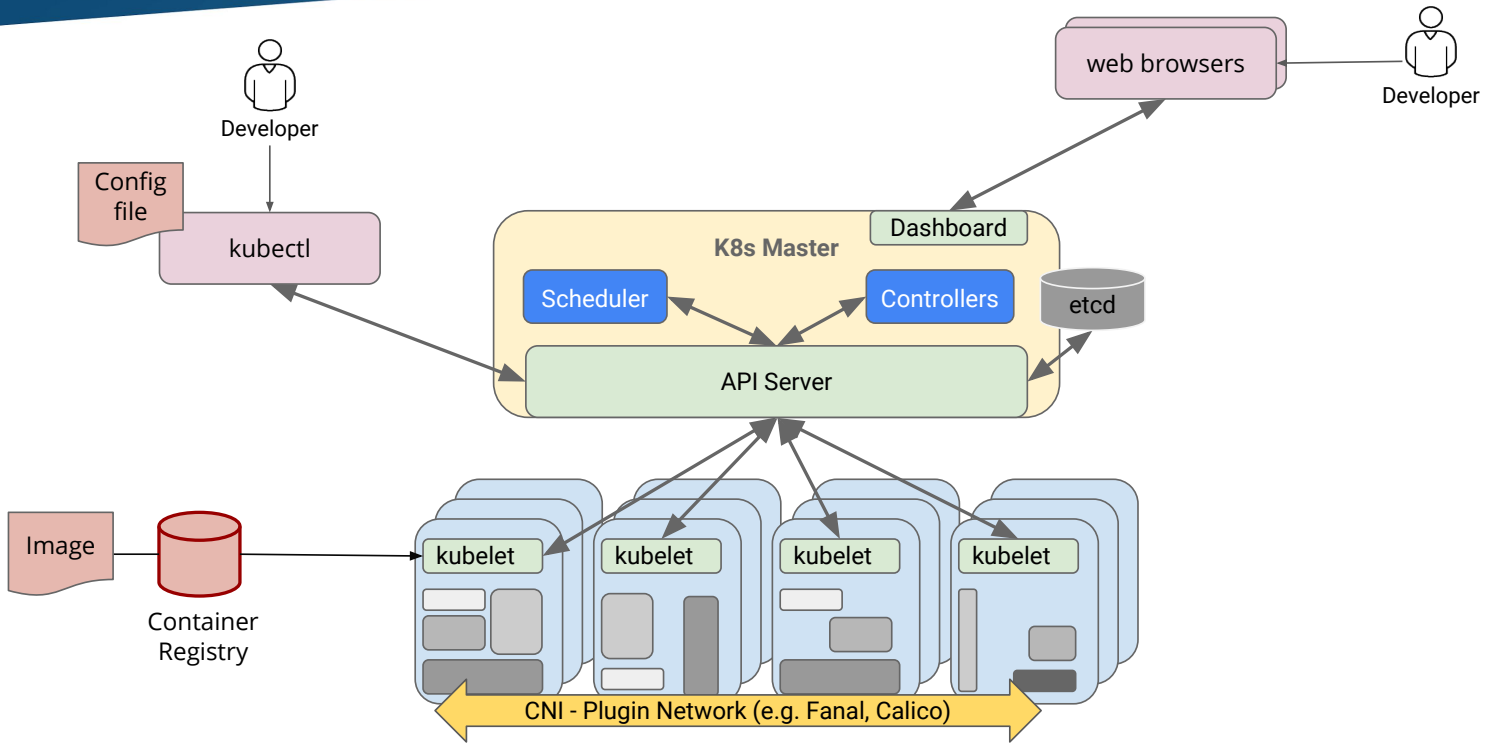
Tooling - Tips & Tricks
- kubectl cheat sheet
- fubectl
- kind
- …

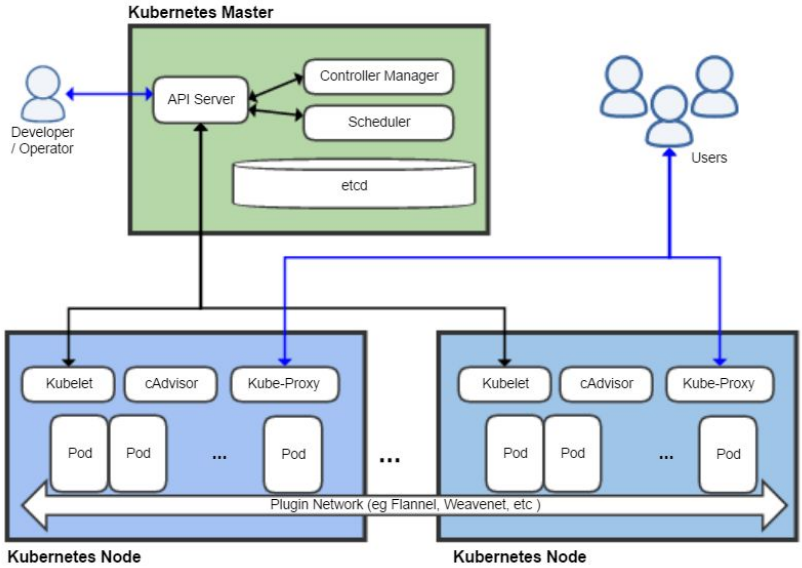
loodse

Kubernetes Architecture



loodse





loodse

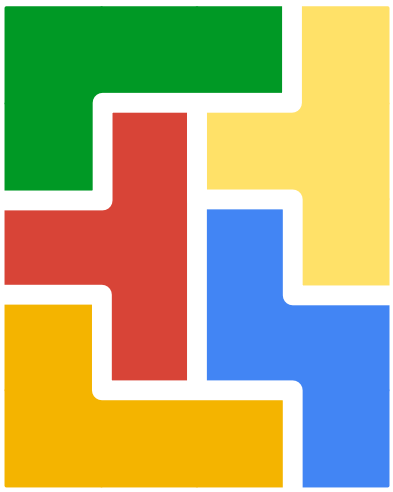
Modularity

Modularity facilitates

- composability
- extensibility
- separate release cycle

APIs - **no shortcuts** or back doors

- ensures extensions are on equal footing



Example:

- **Scheduler**
- **Controllers**
- **Cloud Manager** (cloud provider features)

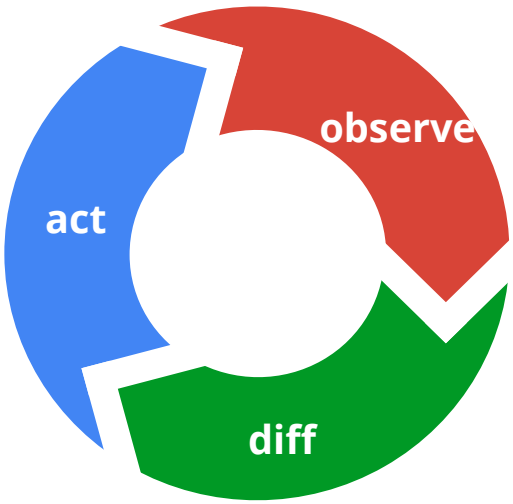
Control loops

Drive **current state** → **desired state**

Observed state is truth
- Open World -- anything can happen

Act independently
- *choreography* rather than centralized *orchestration*

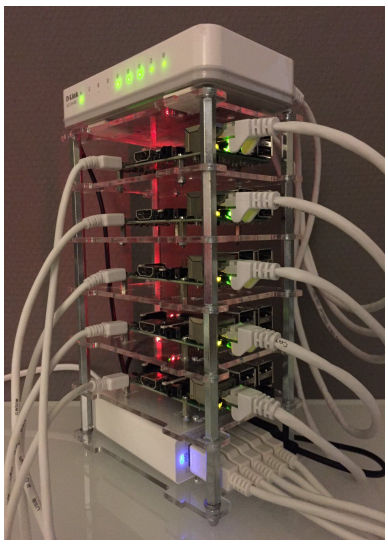
Recurring pattern in the system, e.g.:
- **Scheduler**
- **Controllers**



loodse

Start with a cluster

- Laptop to high-availability multi-node cluster
- Hosted or self managed
- On-Premise or Cloud
- Bare Metal or Virtual Machines
- Most OSes (inc. RedHat Atomic, Fedora, CentOS)
- Or just a bunch of Raspberry PIs



Many options, see Kubernetes Cluster Matrix for details:

https://kubernetes.io/docs/setup/pick-right-solution/

loodse

Setting up a cluster

- Choose a platform: GCE, AWS, Azure, Rackspace, Ubuntu, Juju …
  - Then run:

```
export KUBERNETES_PROVIDER=<your_provider>; curl -sS https://get.k8s.io | bash
```

- Or choose a distro such as RedHat Atomic, CoreOS Tectonic, Mirantis Murano (OpenStack), Mesos
- Or use recipes for bare metal configurations for Centos, Fedora, etc
- Use a hosted option such as Google Container Engine

loodse

Kubernetes in Docker (KinD)

Probably the quickest way to get something in your hands for testing:

```
docker run -d --privileged -p 8443:8443 -p 10080:10080 bsycorp/kind:latest-1.12

mkdir ~/.kube
wget -O ~/.kube/config 127.0.0.1:10080/config

apt-get install -y kubectl

kubectl get all
```

```
root@thz-instance-2:~# kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   19d
root@thz-instance-2:~#
```

loodse

Kubernetes Installation

- The hard way: https://github.com/kelseyhightower/kubernetes-the-hard-way
- Kubeadm: https://kubernetes.io/docs/setup/independent/install-kubeadm/

```
export pubip=192.0.2.1

sudo apt-get install -y "docker-ce=18.06*" kubelet kubeadm kubectl
# sudo apt-mark hold kubelet kubeadm kubectl docker-ce

sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-cert-extra-sa$pubip
mkdir -p $HOME/.kube && sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/rbac-kdd.yaml
kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/kubernetes-datastore/calico-networking/1.7/calico.yaml

kubectl taint nodes --all node-role.kubernetes.io/master-
kubectl get nodes
```

Get this snippet: **https://thz.loodse.training/kubeadm-go.sh**

loodse

Katacoda > Getting Started with kubeadm

https://www.katacoda.com/courses/kubernetes/getting-started-with-kubeadm

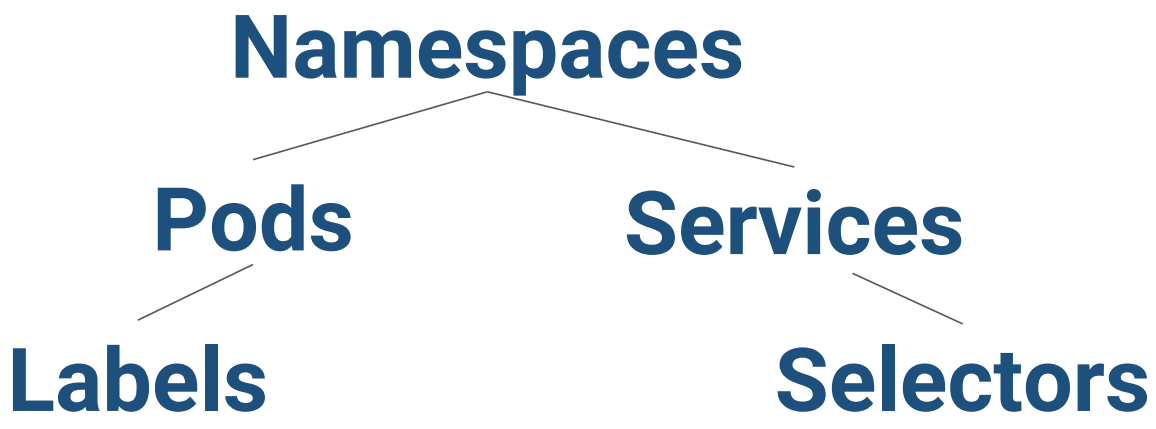
loodse

The Kubernetes Resources

loodse

Core Concepts

Kubernetes has several core building blocks that make up the foundation of their higher level components.



loodse

Namespaces

Namespaces provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces.

Namespaces are a way to divide cluster resources between multiple users.

Most Kubernetes resources (e.g. *pods*, *services*, *replication controllers*, and others) are in some namespaces. However namespace resources are not themselves in a namespace. And low-level resources, such as *nodes* and *persistentVolumes*, are not in any namespace.

```yaml
kind: Namespace
apiVersion: v1
metadata:
  name: test
  labels:
    name: test
```

loodse

Namespaces

- Namespaces are a way of grouping objects
- Some objects are non-namespaced, e.G. `nodes`
- By default, the "`default`" namespace is used
- "`kubectl -n my-namespace`" performs operation in a different namespace
- Try it out: "`kubectl get pod -n kube-system`"

Namespaces

Namespaces are a logical cluster or environment, and are the primary method of partitioning a cluster or scoping access.

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: prod
  labels:
    app: MyBigWebApp
```

```
$ kubectl get ns --show-labels
NAME          STATUS    AGE       LABELS
default       Active    11h       <none>
kube-public   Active    11h       <none>
kube-system   Active    11h       <none>
prod          Active    6s        app=MyBigWebApp
```

loodse

Default Namespaces

- **default**: The default namespace for any object without a namespace.

```
$ kubectl get ns --show-labels
NAME          STATUS    AGE       LABELS
default       Active    11h       <none>
kube-public   Active    11h       <none>
kube-system   Active    11h       <none>
```

- **kube-system**: Acts as the home for objects and resources created by Kubernetes itself.

- **kube-public**: A special namespace; readable by all users that is reserved for cluster bootstrapping and configuration.

loodse

“A pod is a social group of whales”

loodse

A pod of ~~whales~~ containers

The atom of scheduling for containers

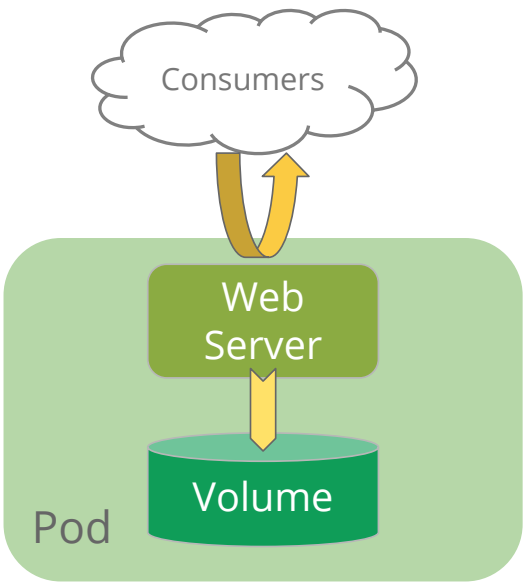
Represents an application specific **logical host**

Hosts **containers** and **volumes**

Each has its own routable (no NAT) IP address

Ephemeral

- Pods are functionally identical and therefore ephemeral and replaceable



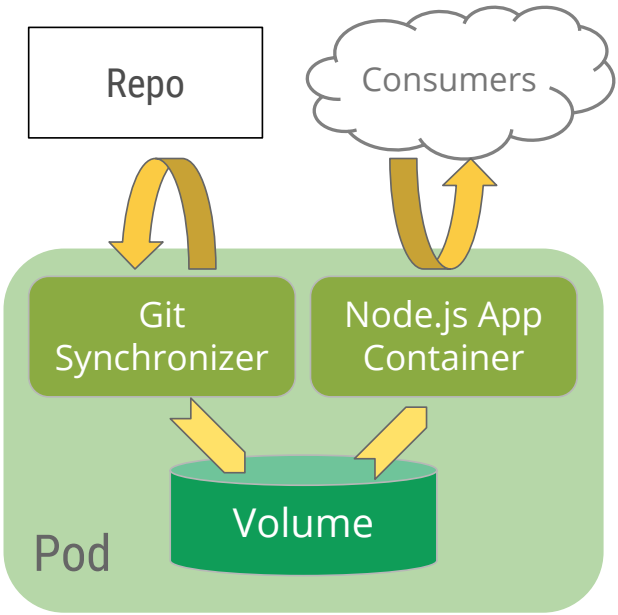
Pods

Can be used to group multiple containers & shared volumes

Containers within a pod are **tightly** coupled

Shared namespaces

- Containers in a pod share IP, port and IPC namespaces
- Containers in a pod talk to each other through localhost



loodse

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/courses/kubernetes/kubernetes-01-playground

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/training/siemens-k8s/kubernetes-01-playground

loodse

Task: Create a pod

- Create a file "*pod.yaml*" with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - image: my-image
    name: my-container-name
```

- Run "*kubectl apply -f pod.yaml*"
- Try to deploy a Nginx pod

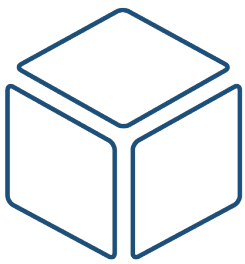
More Kubectl

- Interact with Kubernetes API trough “*kubectl*”
- Try to explorer the API with “pod” object

```
kubectl { get | describe | delete } OBJECT_TYPE NAME
kubectl apply -f myfile.yaml
kubectl { logs | exec } POD_NAME
kubectl explain objectType.field.subField
```

loodse

Kubernetes Resources in Detail



loodse

Pod networking (across nodes)

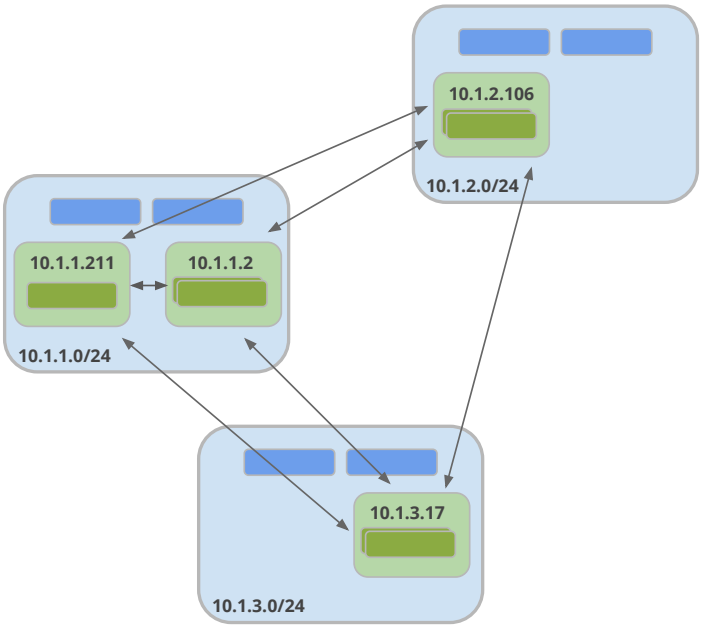
Pods have IPs which are routable
Pods can reach each other without NAT

- Even across nodes

No Brokering of **Port Numbers**
These are fundamental requirements

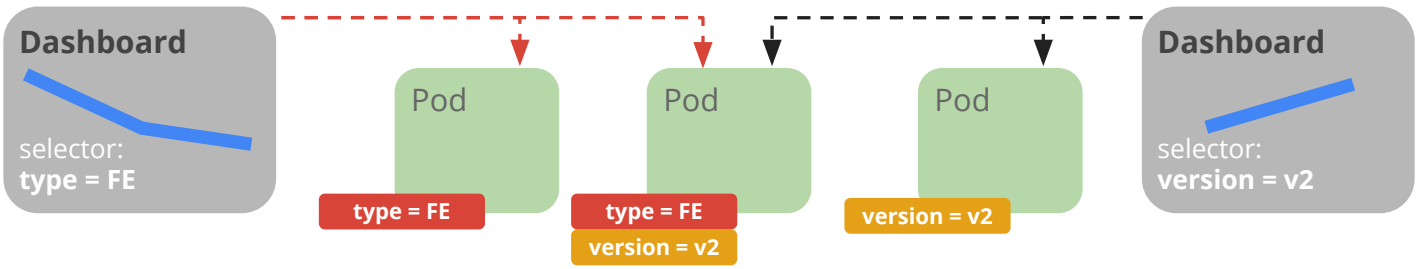
Many solutions

- GCE Advanced Routes, AWS Flannel, Weave, OpenVSwitch, Cloud Provider



loodse

Labels



Behavior

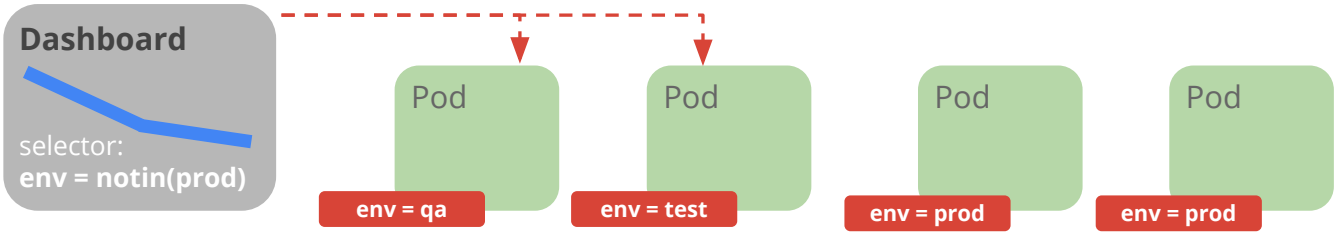
- Metadata with semantic meaning
- Membership identifier
- The only Grouping Mechanism

Benefits

- Allow for intent of many users (e.g. dashboards)
- Build higher level systems …
- Queryable by Selectors

loodse

Label expressions

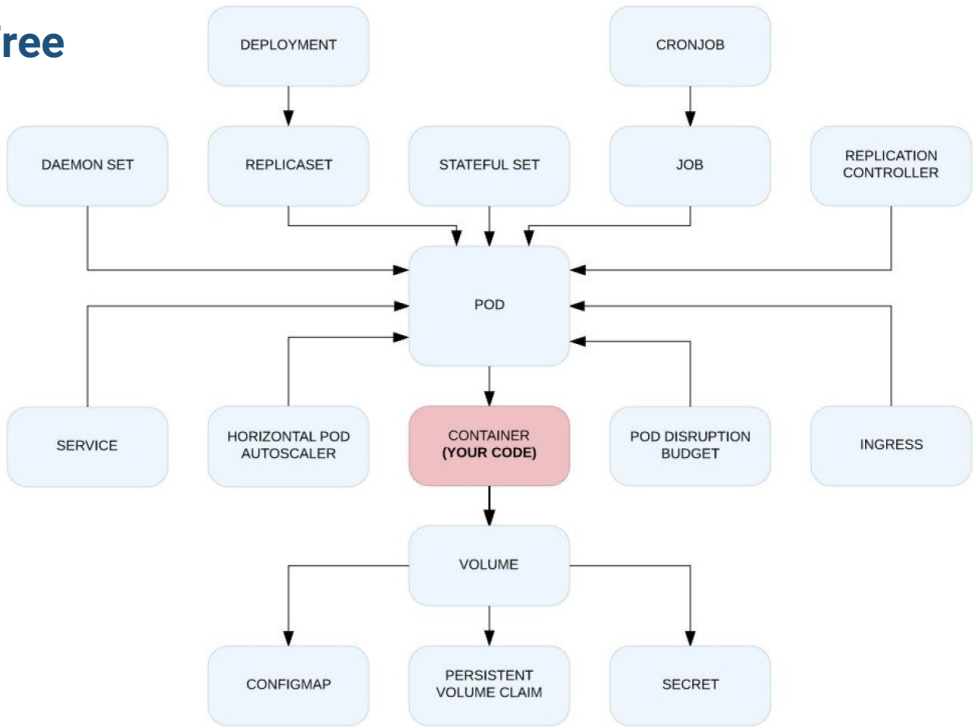


Expressions

- env = prod
- tier != backend
- env = prod, tier !=backend
- env in (test,qa)
- release notin (stable,beta)
- tier
- !tier

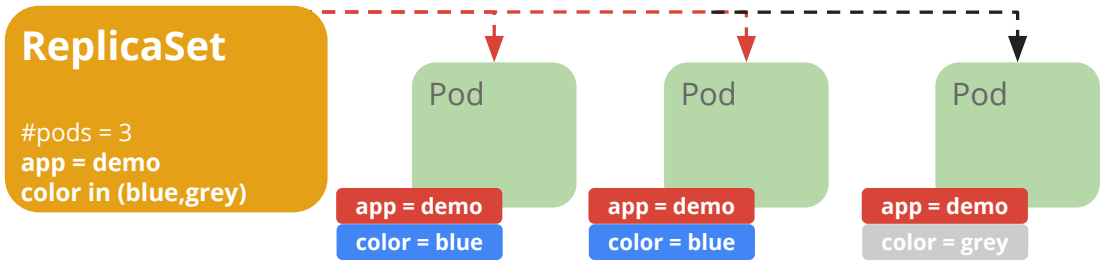
loodse

Kubernetes Object Tree



loodse

ReplicaSet



Behavior

- Keeps Pods running
- Gives direct control of Pod #s
- Grouped by Label Selector

Benefits

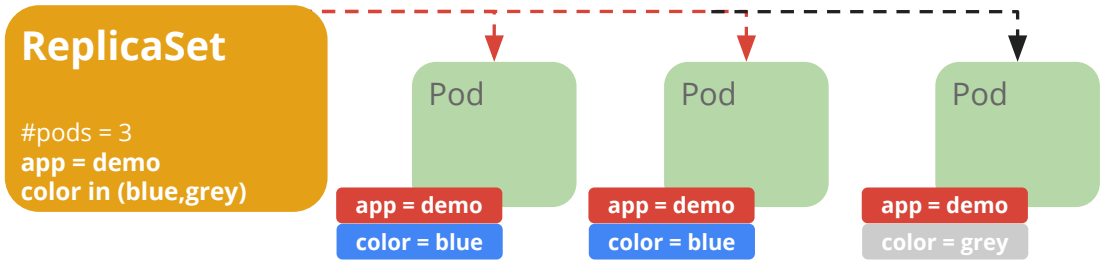
➔ Recreates Pods, maintains desired state

➔ Fine-grained control for scaling

➔ Standard grouping semantics

loodse

ReplicaSet



Supports generalized Selectors

```
selector:
   matchLabels:
     app: demo
   matchExpressions:
     - {key: color, operator: In, values: [blue,grey]}
```

ReplicaSet

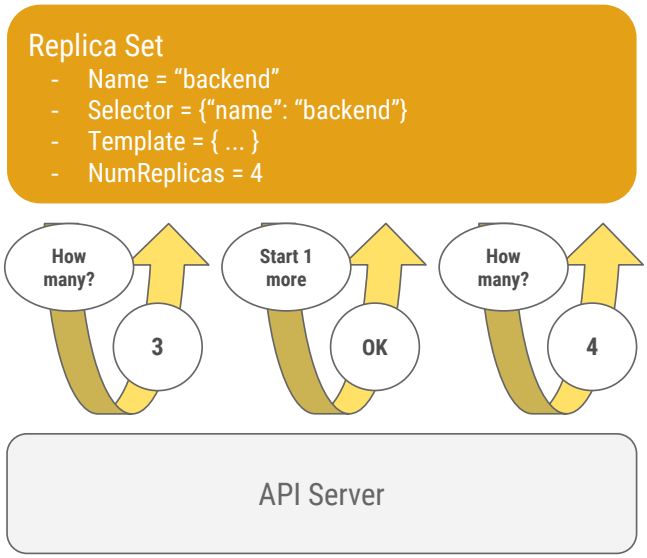
Canonical example of control loops

Have one job: ensure N copies of a pod

- if too few, start new ones
- if too many, kill some
- group == selector

Replicated pods are fungible

- No implied order or identity



loodse

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/courses/kubernetes/kubernetes-01-playground

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/training/siemens-k8s/kubernetes-01-playground

loodse

Task: Create a Replicaset

```yaml
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: another-app
    spec:
      containers:
      - image: my-image
        name: my-container
```

loodse

Services

A logical grouping of pods that perform the same function (the Service's endpoints)

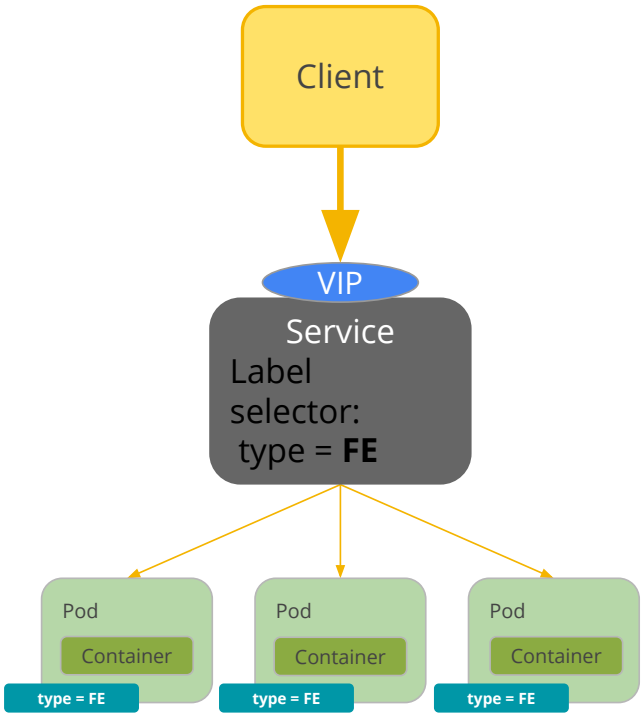
- grouped by label selector

Load balances incoming requests across constituent pods

Choice of pod is random but supports session affinity (ClientIP)

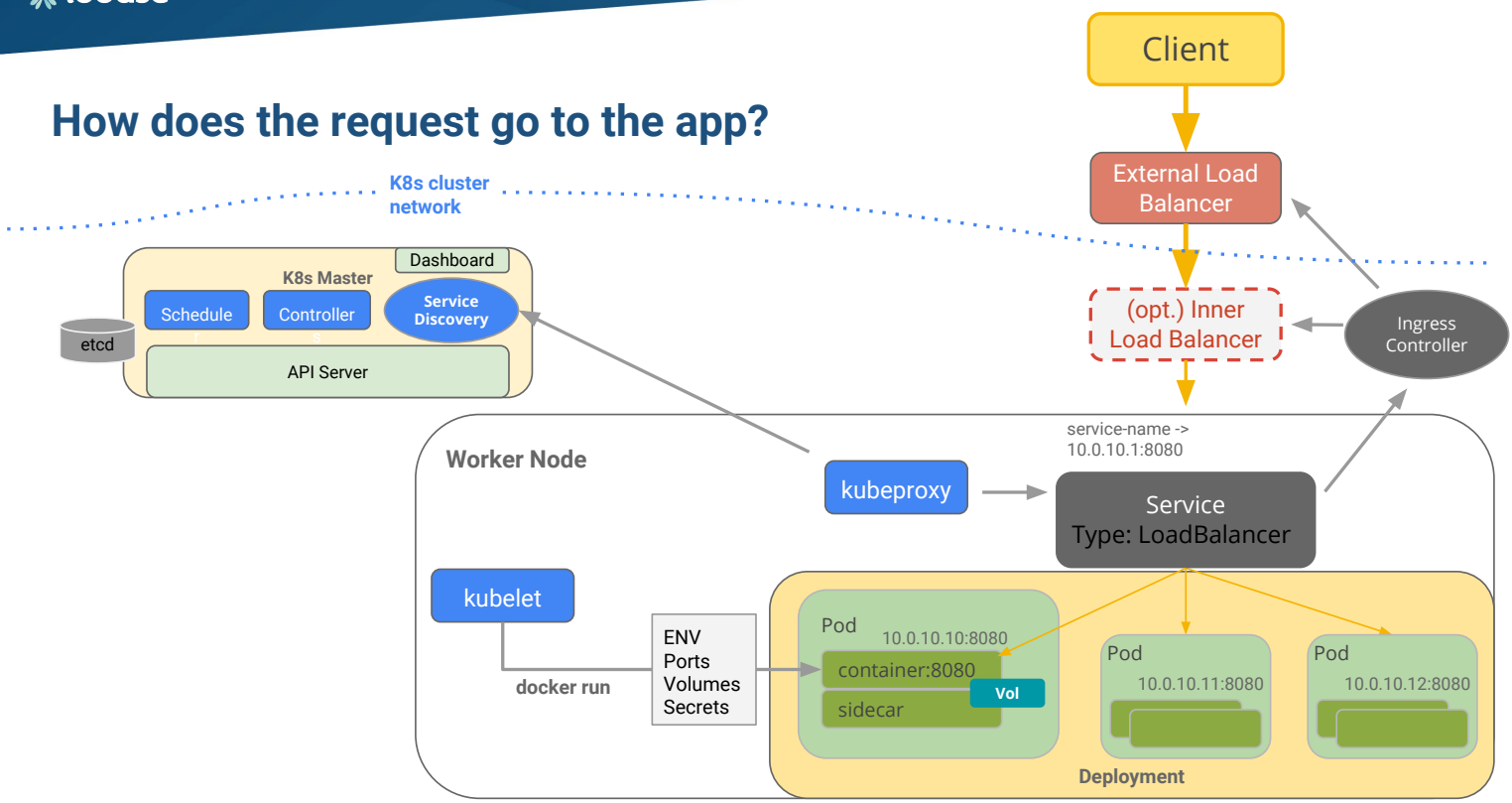
Gets a **stable** virtual IP and port

- also a DNS name



loodse

How does the request go to the app?



Task: Create a service

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: test
  name: test
spec:
  type: LoadBalancer
  ports:
  - name: "8080"
    port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: that-app
```

Verification: *"kubectl get endpoints SERVICE_NAME"*

loodse

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/courses/kubernetes/kubernetes-01-playground

Katacoda > Kubernetes Playground

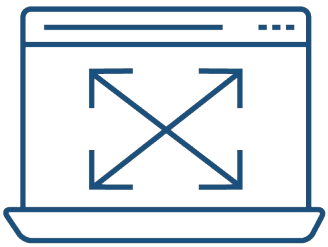
https://www.katacoda.com/loodse/training/siemens-k8s/kubernetes-01-playground

loodse

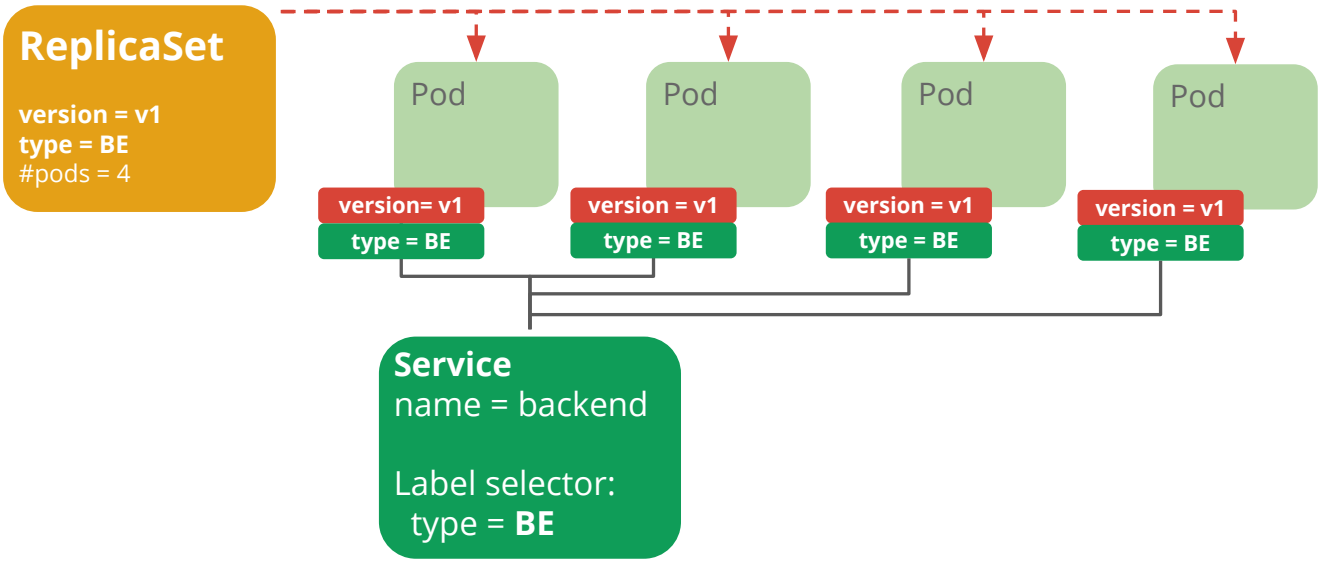
Katacoda
> Deploy Guestbook example

https://www.katacoda.com/courses/kubernetes/guestbook

How to scale?

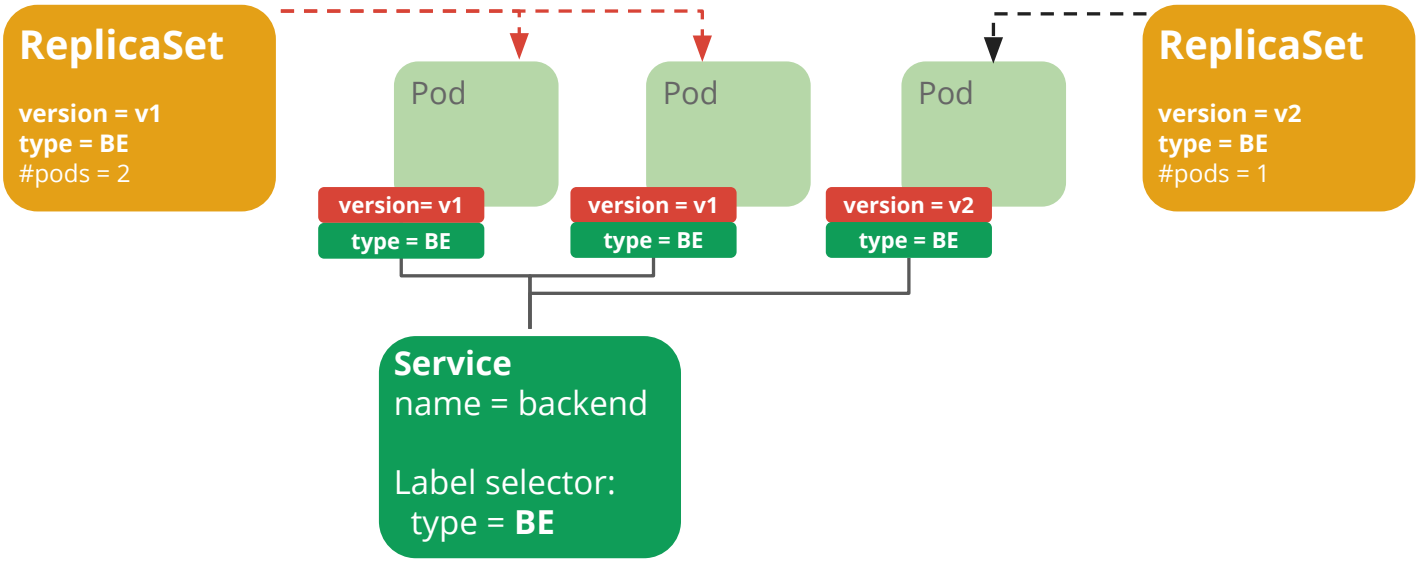


Scaling Example



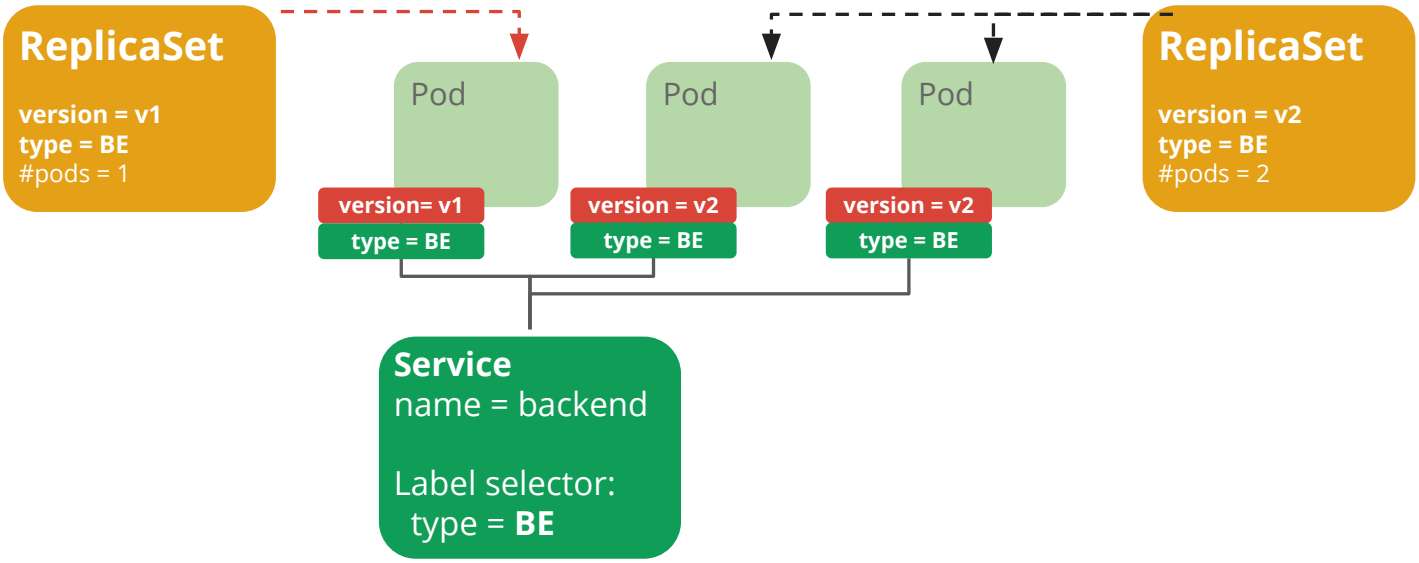
loodse

Canary

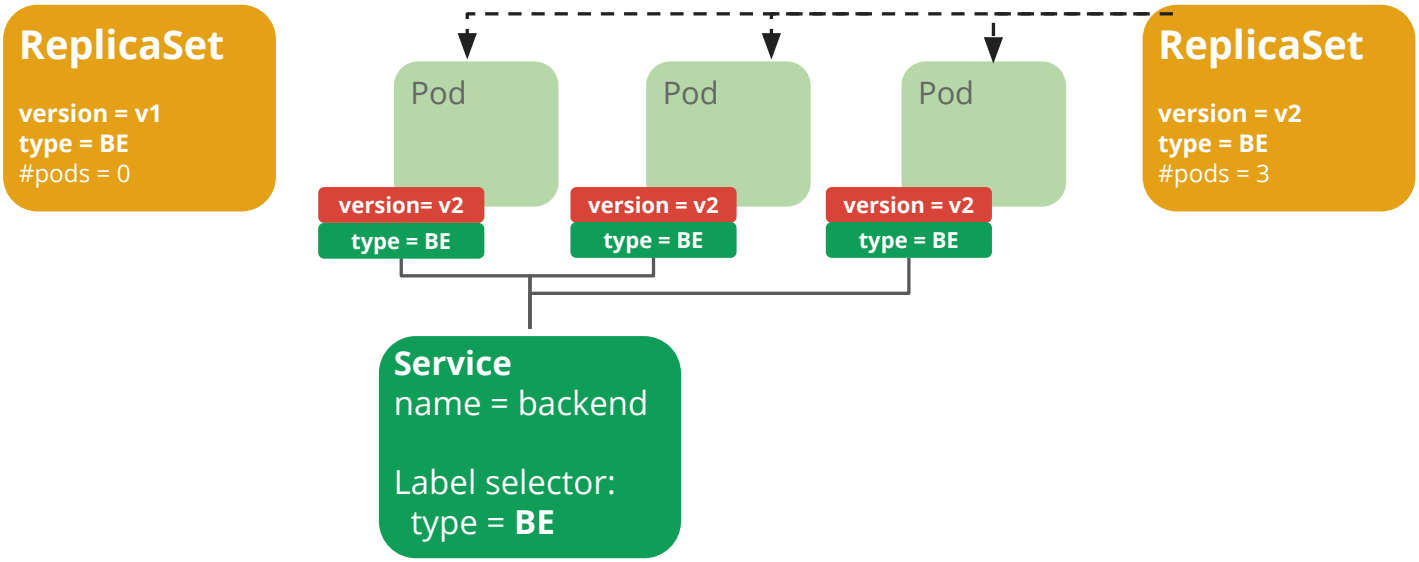


loodse

Rollout



Rollout



loodse

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/courses/kubernetes/kubernetes-01-playground

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/training/siemens-k8s/kubernetes-01-playground

loodse

Task: Add additional canary pod for service

- Create a new *ReplicaSet*
- Make sure your existing Service captures pods of that *ReplicaSet*
- Verify this by checking the *Endpoints* of your *Service*

loodse

Reliable Deployments

loodse

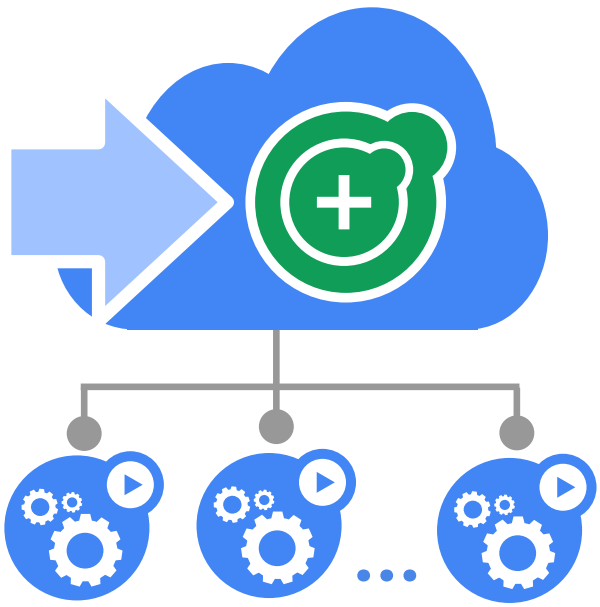
Deployments: Updates as a Service

Reliable mechanism for creating, updating and managing Pods

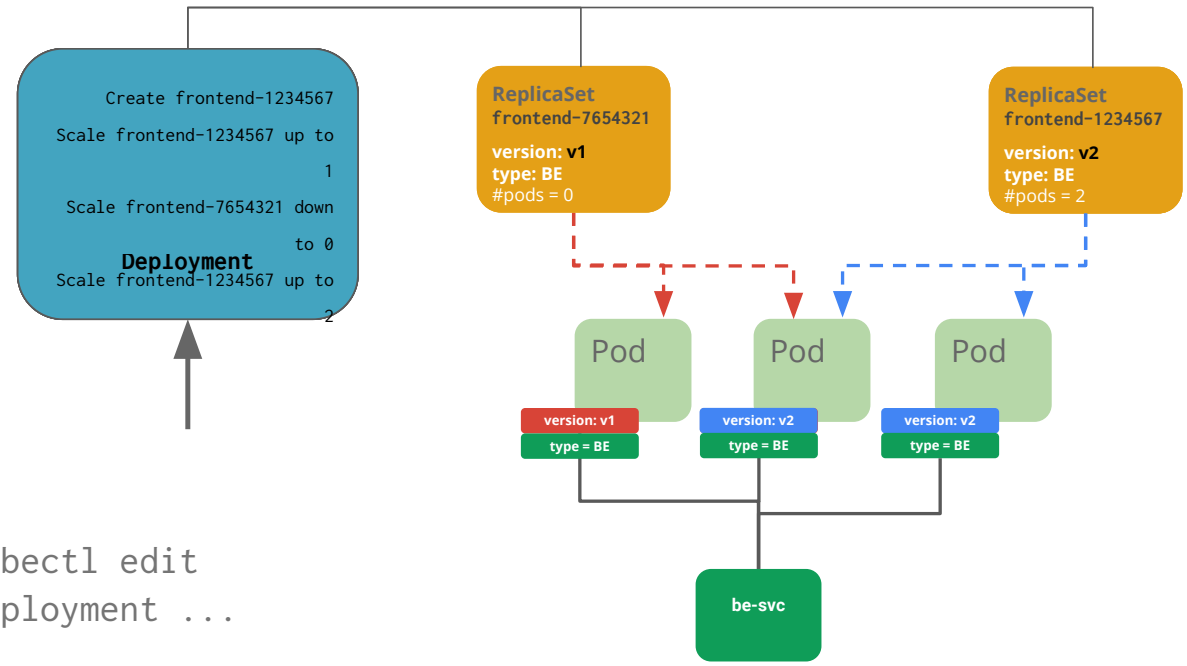
Deployment manages replica changes, including rolling updates and scaling

Edit Deployment configurations in place with kubectl edit or kubectl apply

Managed rollouts and rollbacks



Rollout




```
kubectl edit
deployment ...
```

loodse

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/courses/kubernetes/kubernetes-01-playground

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/training/siemens-k8s/kubernetes-01-playground

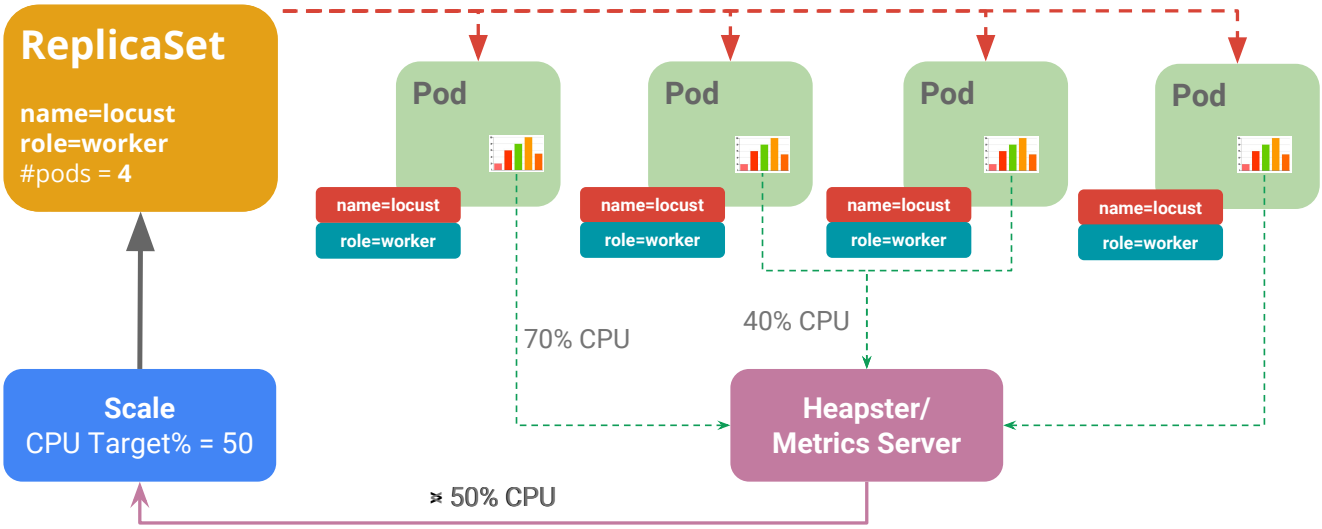
loodse

Task: Create a Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test-deployment
  template:
    metadata:
      labels:
        app: test-deployment
    spec:
      containers:
      - image: helloworld
        name: helloworld
```

- Create the `Deployment`
- Change it
- Watch how a new *`ReplicaSet`* and *Pod* is created

Horizontal Pod Autoscaling



loodse

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/courses/kubernetes/kubernetes-01-playground

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/training/siemens-k8s/kubernetes-01-playground

loodse

Task: Create a HorizontalPodAutoscaler

```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: my-autoscaler
spec:
  maxReplicas: 5
  minReplicas: 1
  scaleTargetRef:
    apiVersion: extensions/v1beta1
    kind: Deployment
    name: my-deployment
  targetCPUUtilizationPercentage: 50
```

- Create the HPA
- Create enough load so it scales your deployment

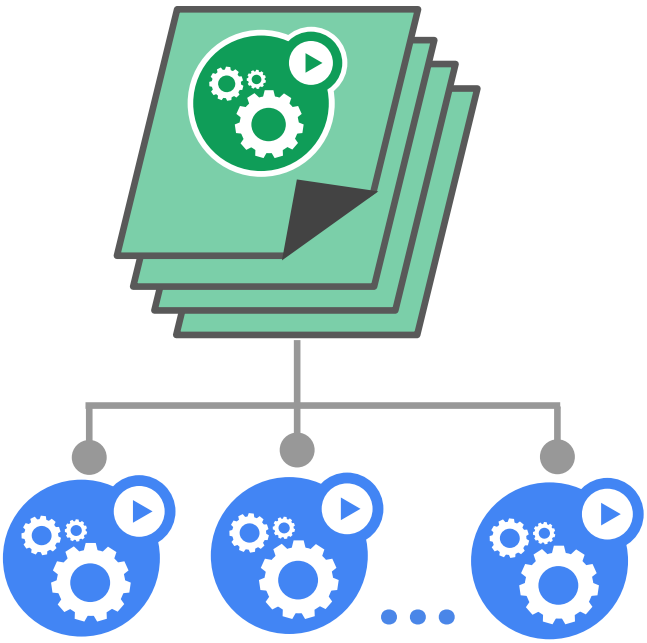
Jobs

Run-to-completion, as opposed to run-forever

- Express parallelism vs. required completions
- Workflow: restart on failure
- Build/test: don't restart on failure

Aggregates success/failure counts

Built for batch and big-data work



loodse

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/courses/kubernetes/kubernetes-01-playground

Katacoda > Kubernetes Playground

https://www.katacoda.com/loodse/training/siemens-k8s/kubernetes-01-playground

loodse

Task: Create a job

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: my-job
spec:
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
  backoffLimit: 4
```

Graceful Termination

Goal: Give pods time to clean up

- finish in-flight operations
- log state
- flush to disk
- 30 seconds by default

Catch **SIGTERM**, cleanup, exit ASAP

Pod status “Terminating”

Declarative: ‘DELETE’ appears as an object field in the API

loodse

Multi-Zone Cluster

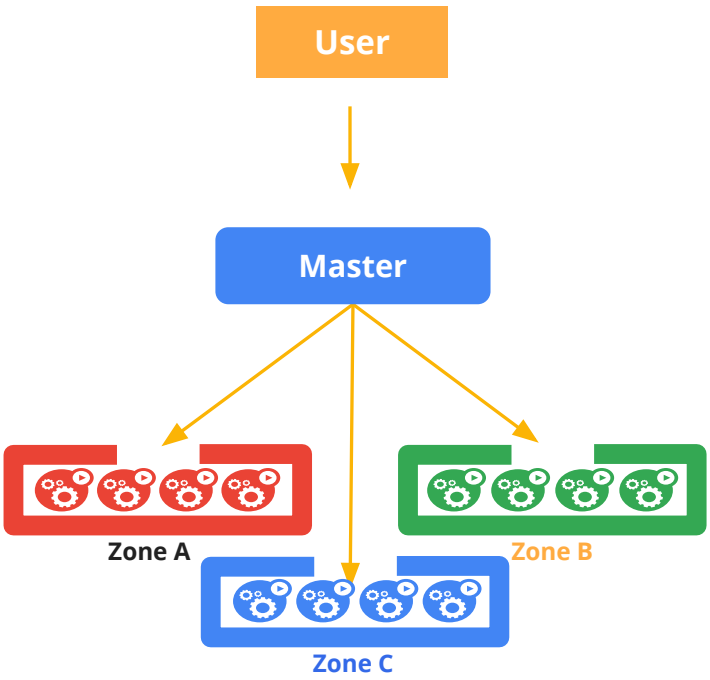
Goal: zone-fault tolerance for applications

Zero API changes relative to kubernetes

- Create services, replication controllers, etc. exactly as usual
- Meta Information (no default behaviours)

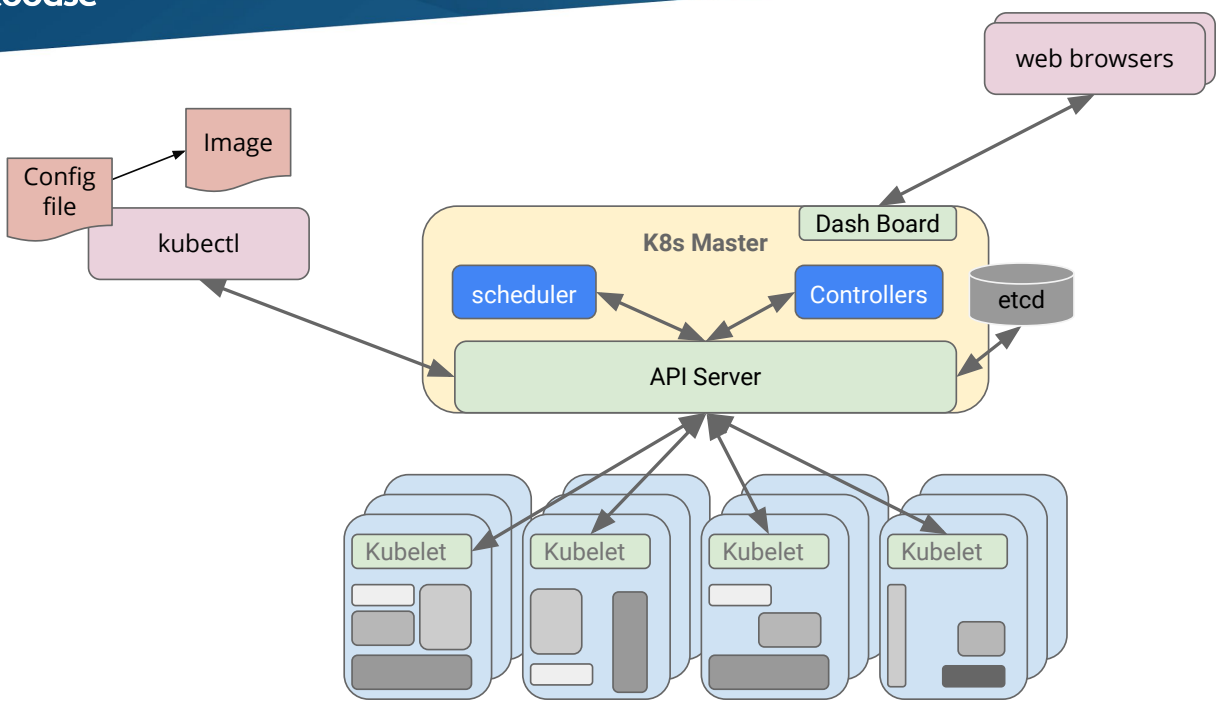
Nodes and PersistentVolumes are labelled with their availability zone

- Fully automatic for GKE, GCE, AWS
- Manual for on-premise and other cloud providers (for now)



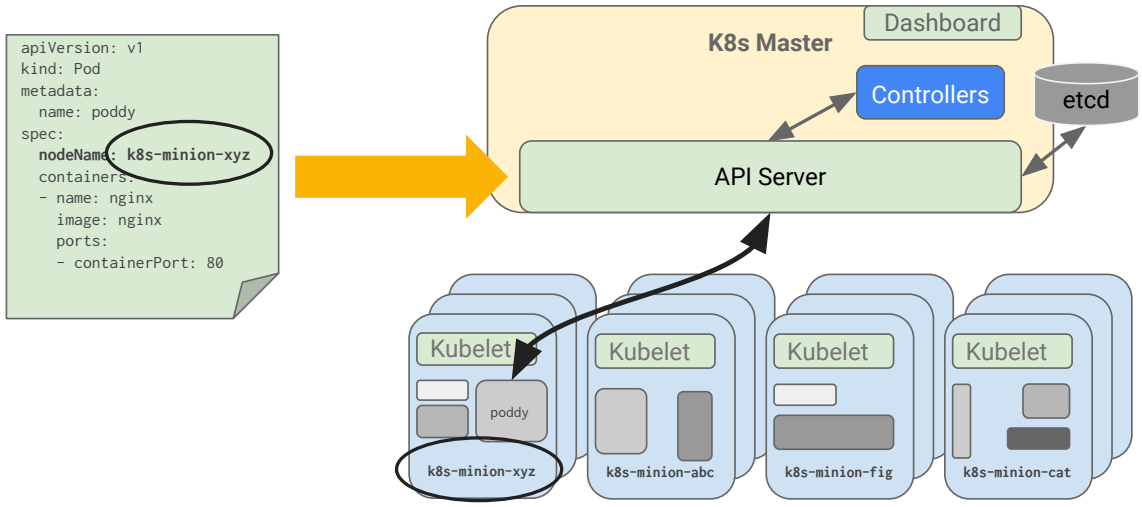
Scheduling

loodse



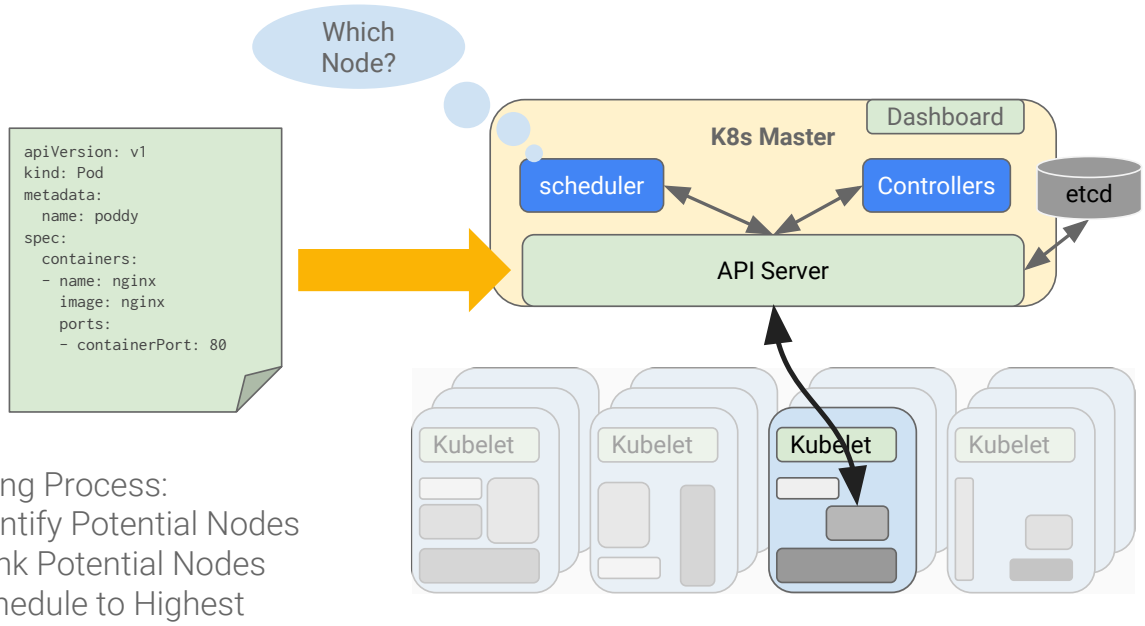
loodse

Kubernetes without a scheduler



loodse

Kubernetes with a scheduler



Scheduling Process:

- Identify Potential Nodes
- Rank Potential Nodes
- Schedule to Highest Ranked Node

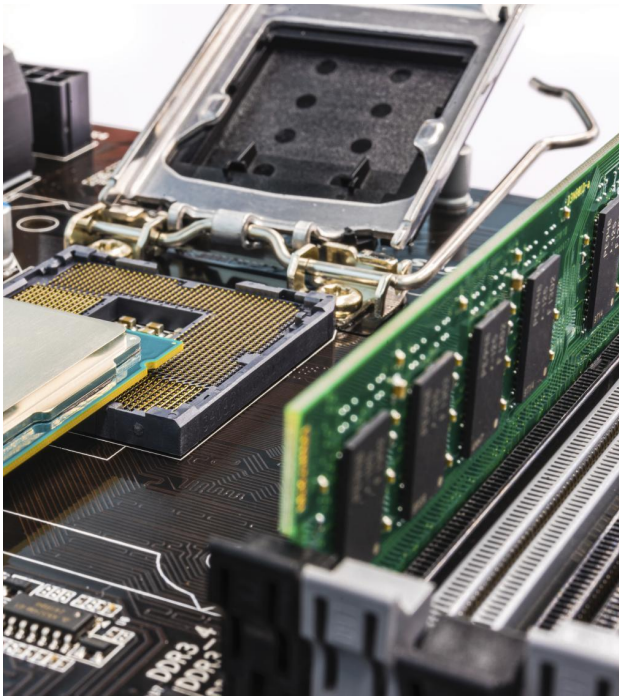
Kubernetes Resources

A Resource is something that can be **requested**, **allocated**, or **consumed** to/by a pod or a container

> **CPU**: Specified in units of Cores, what that is depends on the provider
>
> **Memory**: Specified in units of Bytes

CPU is **Compressible** (i.e. it has a ***rate*** and can be throttled)

Memory is **Incompressible**, it can't be throttled



loodse

Requests and limits

Request:

- how much of a resource you are asking to use, with a strong guarantee of availability
- scheduler will not over-commit requests

Limit:

- max amount of a resource you can access

Conclusion:

- Usage > Request: resources **might** be available
- Usage > Limit: throttled or killed

Resource-based scheduling

Provide QoS for Scheduled Pods

Per Container CPU and Memory requirements

Specified as **Request** and **Limit**

- **Best Effort** (Request == 0)
- **Burstable** (Request < Limit)
- **Guaranteed** (Request == Limit)

Best Effort Scheduling for low priority workloads improves Utilization at Google by 20%

loodse

Resource-based scheduling

my-controller.yaml

```
...
spec:
  containers:
    - name: locust
      image: gcr.io/rabbit-skateboard/guestbook:gdg-rtv
      resources:
        requests:
          memory: "300Mi"
          cpu: "100m"
        limits:
          memory: "300Mi"
          cpu: "100m"
```

CPU resource: requests vs limits

For a Pod to be *scheduled* the amount of CPU it **Requests** must be available on a single node
If it **Requests** 0 CPU it can always be scheduled

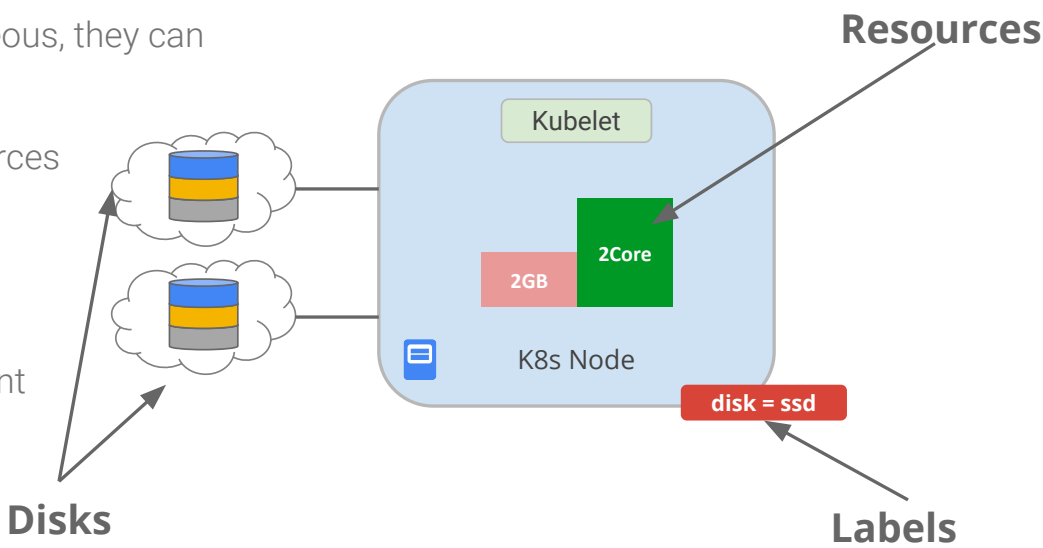
loodse

Scheduling pods: nodes

Nodes may not be heterogeneous, they can differ in important ways:

- CPU and Memory Resources
- Attached Disks
- Specific Hardware

Location may also be important

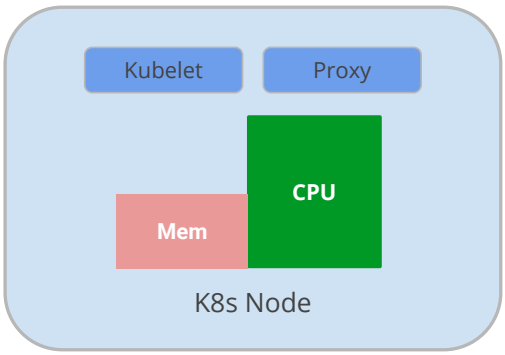


loodse

Pod scheduling: identifying potential nodes

What **CPU and Memory Resources** does it need?

Can also be used as a measure of priority



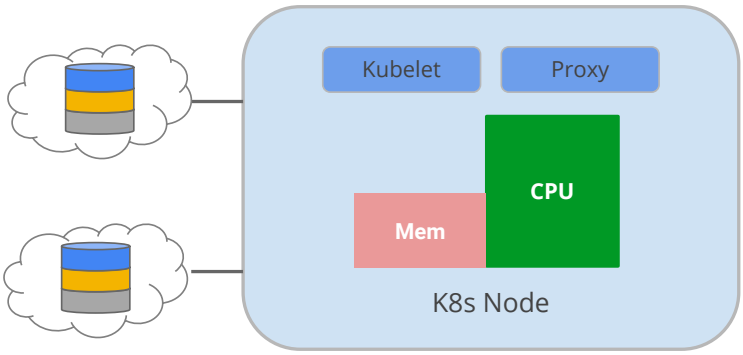
loodse

Pod scheduling: finding potential nodes

What **Resources** does it need?

What **Disk(s)** does it need (GCE PD and EBS) and can it/they be mounted without conflict?

Note: 1.1 limits to a single volume mount per node



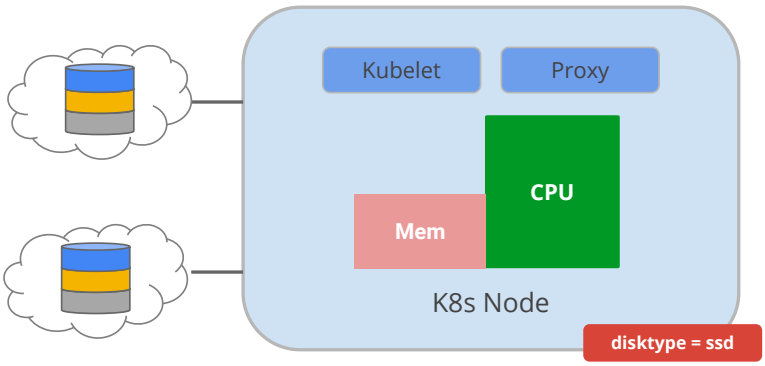
loodse

Pod scheduling: identifying potential nodes

What **Resources** does it need?

What **Disk(s)** does it need?

What node(s) can it run on (**Node Selector**)?




```
$ kubectl label nodes node-3 disktype=ssd
```

```
(pod) spec:
        nodeSelector:
          disktype: ssd
```

loodse

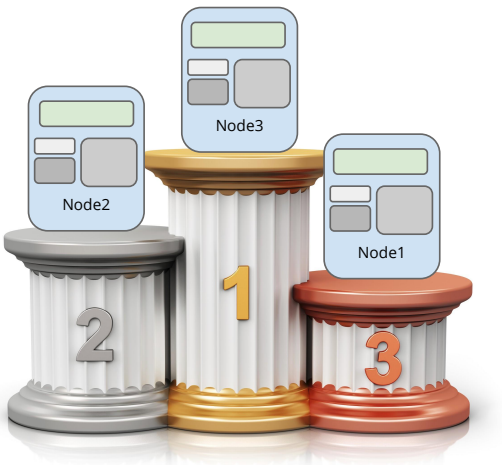
Pod scheduling: ranking potential nodes

Prefer node with most free resource left after the pod is deployed

Prefer nodes with the specified label

Minimise number of Pods from the same service on the same node

CPU and Memory is balanced after the Pod is deployed **[Default]**



loodse

nodeAffinity

Can be ‘Required’ or ‘Preferred’ during scheduling

Can be ‘Required’ during execution (Node labels can change)

Will eventually replace NodeSelector

If you specify both nodeSelector and nodeAffinity, both must be satisfied

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/e2e-az-name
            operator: In
            values:
            - e2e-az1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: another-node-label-key
            operator: In
            values:
            - another-node-label-value
```

https://kubernetes.io/docs/concepts/configuration/assign-pod-node/

loodse

Extending the scheduler

1. Add rules to the scheduler and recompile
2. Run your own scheduler process instead of, or as well as, the Kubernetes scheduler
3. Implement a "scheduler extender" that the Kubernetes scheduler calls out to as a final pass when making scheduling decisions

Running custom and/or multiple scheduler

If the default scheduler does not suit your needs you can implement your own scheduler. Not just that, you can even run multiple schedulers simultaneously alongside the default scheduler and instruct Kubernetes what scheduler to use for each of your pod.

1. https://kubernetes.io/docs/tasks/administer-cluster/configure-multiple-schedulers/
2. https://github.com/IBM/k8s-custom-scheduler
3. https://github.com/kelseyhightower/scheduler

loodse

Admission control

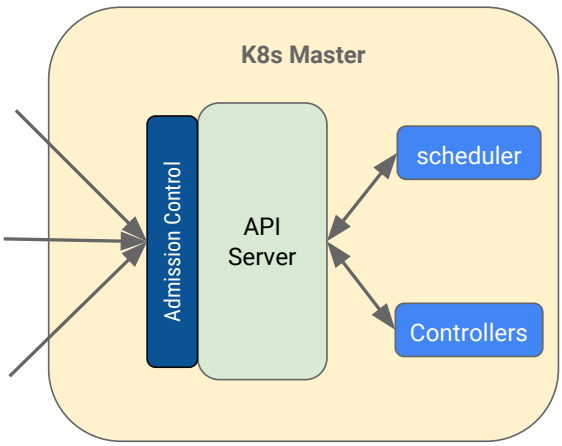
Admission Control (AC) enforces certain conditions, before a request is accepted by the API Server

AC functionality implemented as plugins which are executed in the sequence they are specified

AC is performed after AuthN checks

Enforcement usually results in either

- A Request denial
- Mutation of the Request Resource
- Mutation of related Resources



loodse

Admission Control Examples

NamespaceLifecycle

Enforces that a Namespace that is undergoing termination cannot have new objects created in it, and ensures that requests in a non-existant Namespace are rejected

LimitRanger

Observes the incoming request and ensures that it does not violate any of the constraints enumerated in the LimitRange object in a Namespace

ServiceAccount

Implements automation for serviceAccounts

ResourceQuota

Observes the incoming request and ensures that it does not violate any of the constraints enumerated in the ResourceQuota object in a Namespace.

Default plug-ins in 1.13:

--admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,PersistentVolumeClaimResize,DefaultStorageClass, DefaultTolerationSeconds,MutatingAdmissionWebhook,ValidatingAdmissionWebhook,ResourceQuota,Priority

loodse

Dynamic Admission Control & Initializers

What are admission webhooks?

Admission webhooks are HTTP callbacks that receive admission requests and do something with them. You can define two types of admission webhooks, validating admission Webhook and mutating admission webhook. With validating admission Webhooks, you may reject requests to enforce custom admission policies. With mutating admission Webhooks, you may change requests to enforce custom defaults.

What are initializers?

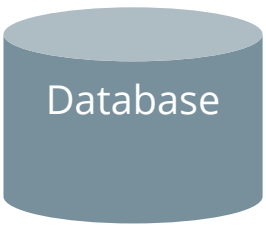
Initializer has two meanings:

- A list of pending pre-initialization tasks, stored in every object's metadata (e.g., "AddMyCorporatePolicySidecar").
- A user customized controller, which actually performs those tasks. The name of the task corresponds to the controller which performs the task. For clarity, we call them initializer controllers in this page.

Managing State

loodse

I still have questions about state!

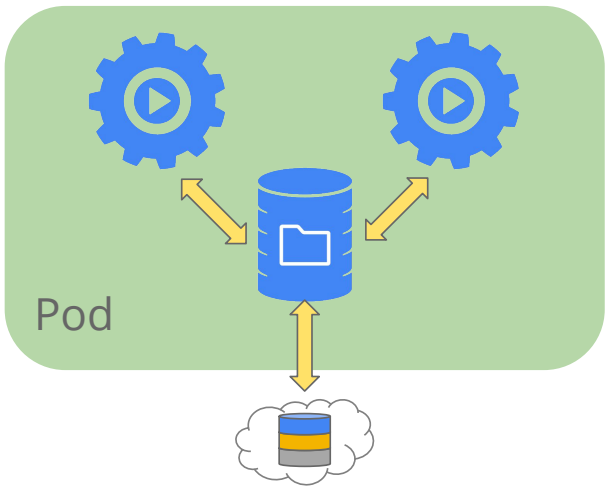


In a cluster of ephemeral containers
Application state must exist outside of the container

Volumes

Bound to the Pod that encloses it
Look like Directories to Containers
What and where they are determined by Volume Type
Many Volume options

- EmptyDir
- HostPath
- nfs (and similar services)
- Cloud Provider Block Storage



loodse

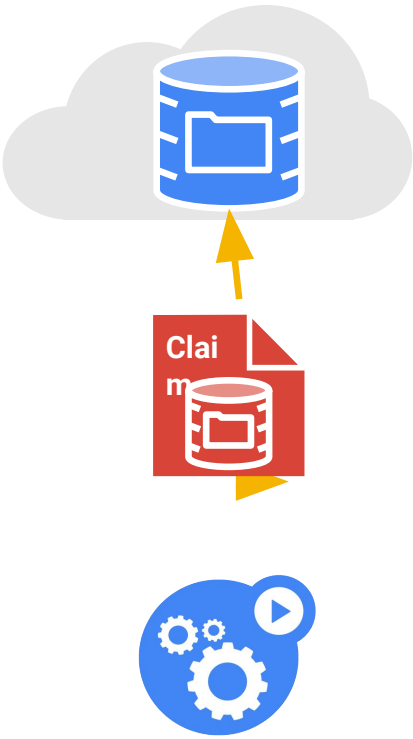
Persistent Volumes

A higher-level storage abstraction
- insulation from any one cloud environment

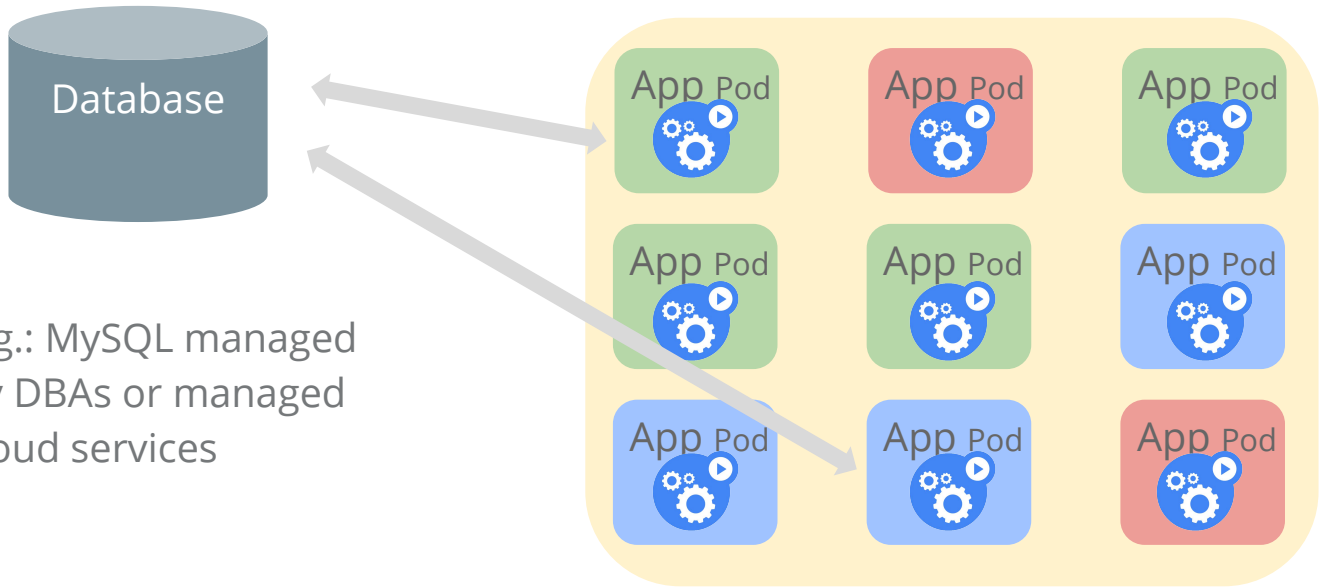
Admin provisions them, users claim them
- auto-provisioning

Independent lifetime and fate from consumers
- lives until user is done with it
- can be handed-off between pods

Dynamically “scheduled” and managed, like nodes and pods



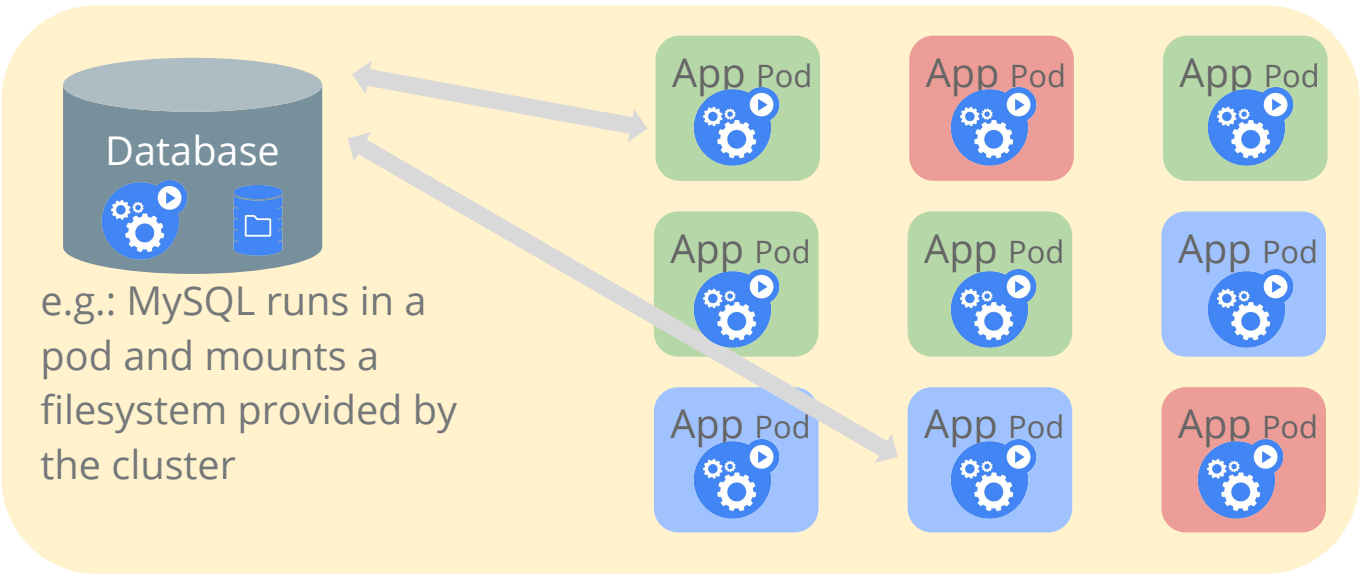
Outside the Cluster



e.g.: MySQL managed by DBAs or managed cloud services

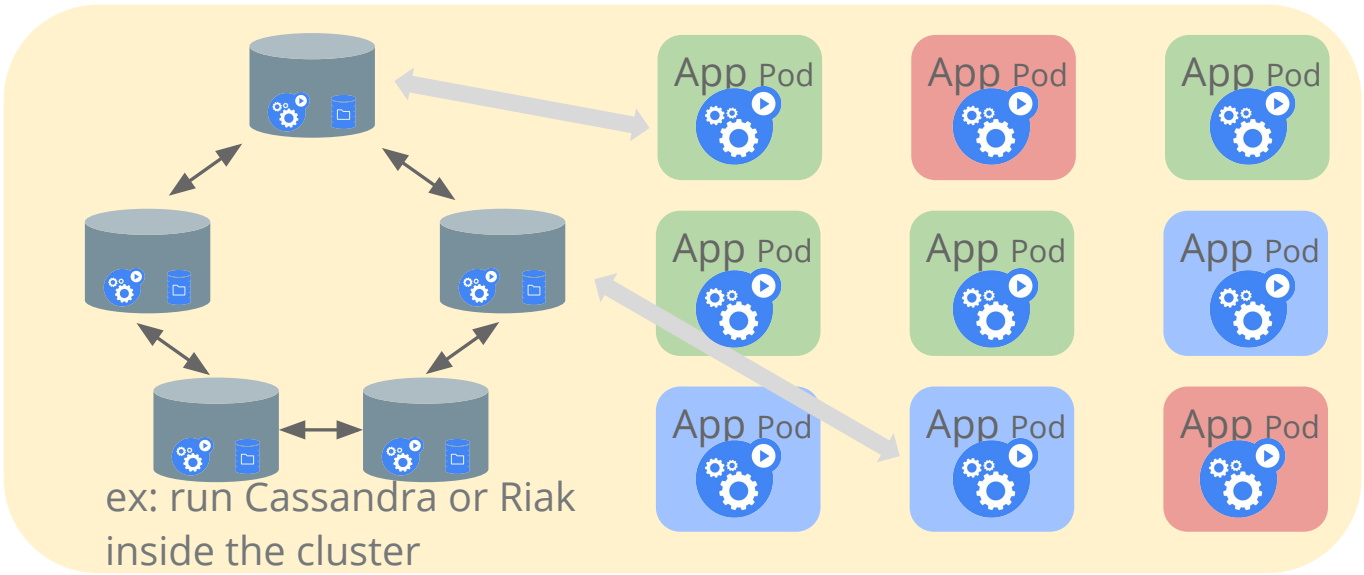
loodse

Adapt to run in the cluster



e.g.: MySQL runs in a pod and mounts a filesystem provided by the cluster

Cluster Native



loodse

ConfigMaps

Goal: manage app configuration

- ...without making overly-brittle container images

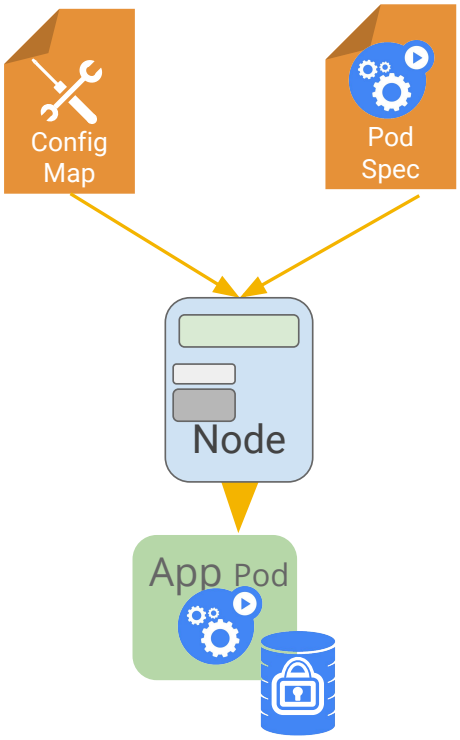
12-factor says config comes from the environment

- Kubernetes is the environment

Manage config via the Kubernetes API

Inject config as a virtual volume into your Pods

- late-binding, live-updated (atomic)
- also available as env vars



loodse

Task: Create a Pod with configmap

- `git clone https://github.com/loodse/kubernetes-lab.git`
- `cd examples/kubernetes-simple`
- `vim pod-with-configmap.yaml`
- Once you are done: `kubectl apply -f pod-with-configmap.yaml`

loodse

Secrets

Goal: grant a pod access to a secured *something*?

- don't put secrets in the container image!

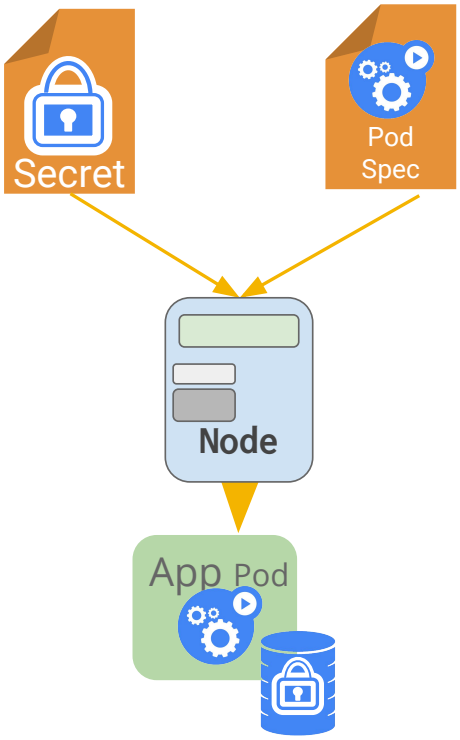
12-factor says: config comes from the environment

- Kubernetes is the environment

Manage secrets via the Kubernetes API

Inject them as virtual volumes into Pods

- late-binding
- tmpfs - never touches disk



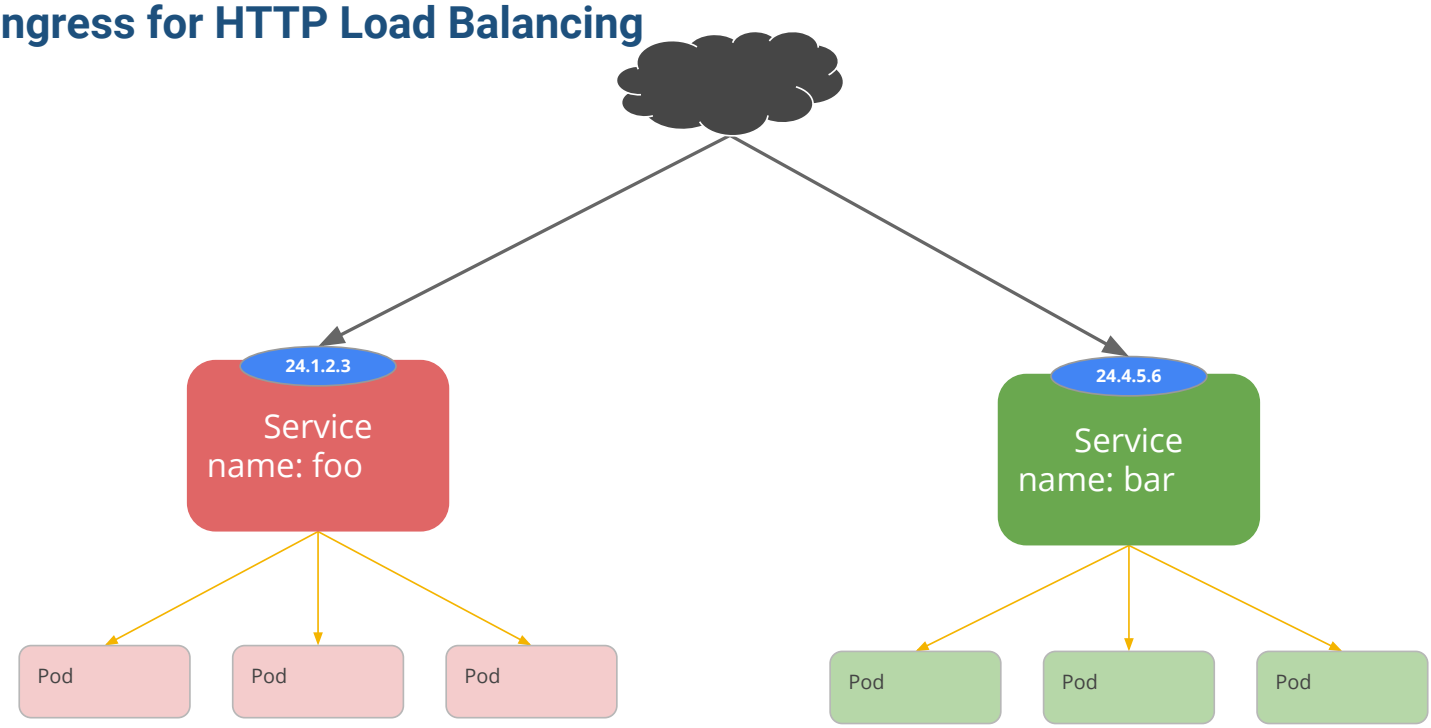
Task: Create secret and use it in a pod

- `git clone https://github.com/loodse/kubernetes-lab.git`
- `cd examples/kubernetes-simple`
- `vim pod-with-secret.yaml`
- `kubectl apply -f pod-with-secret.yaml`

Ingress

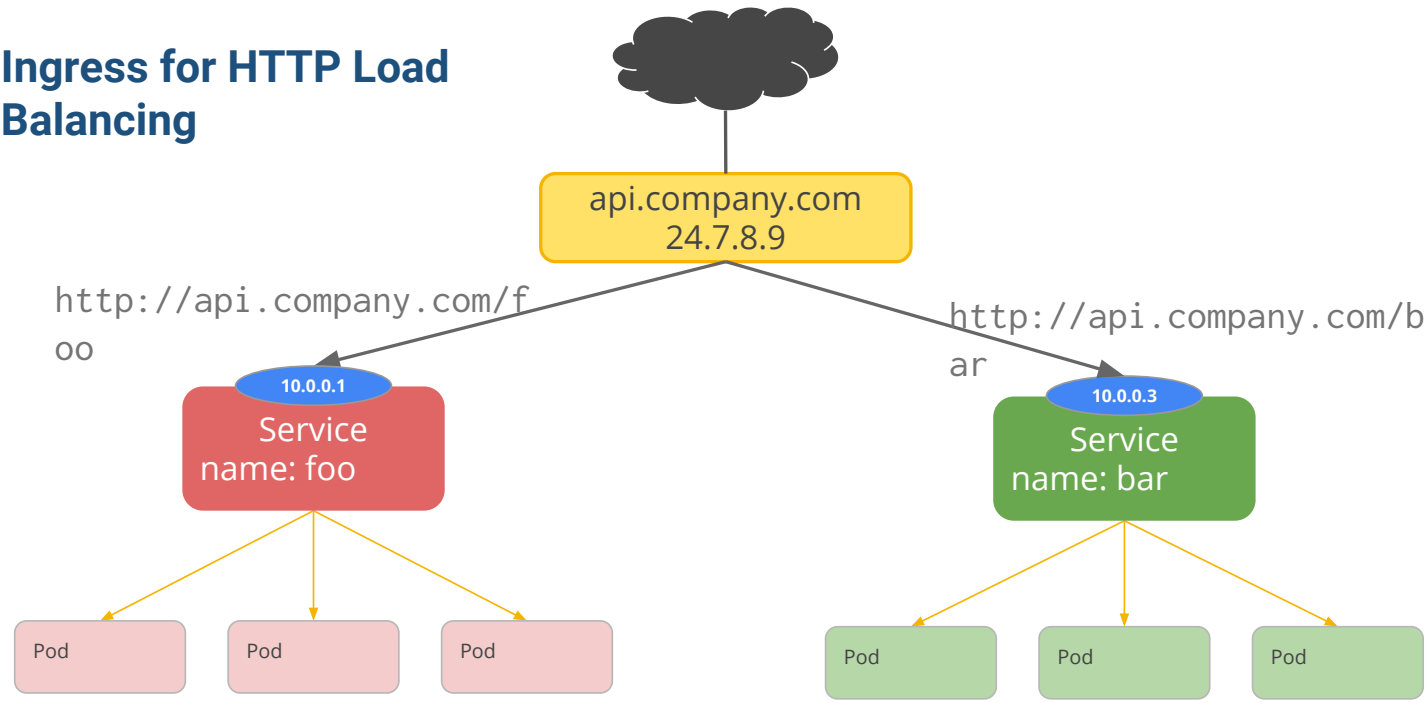
loodse

Ingress for HTTP Load Balancing



loodse

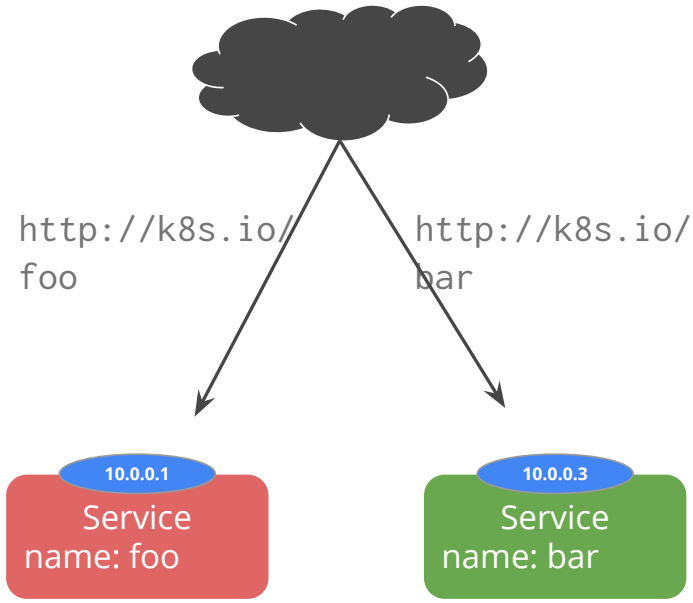
Ingress for HTTP Load Balancing



loodse

Ingress API: Simple Fanout

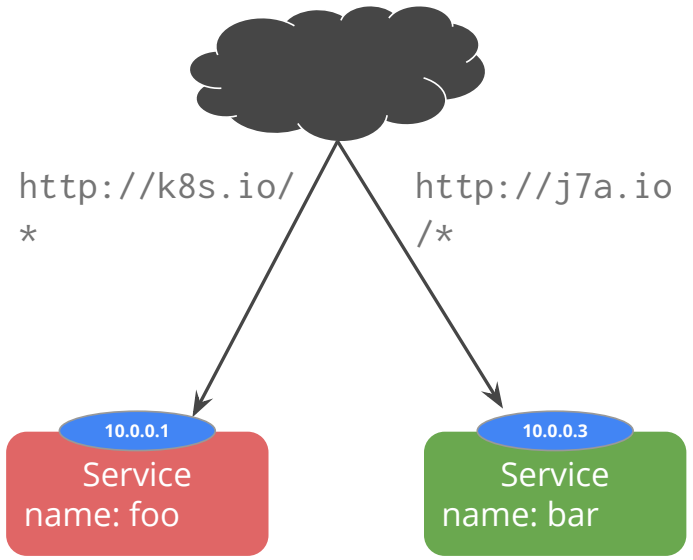
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
  - host: k8s.io
    http:
      paths:
      - path: /foo
        backend:
          serviceName: fooSvc
          servicePort: 80
      - path: /bar
        backend:
          serviceName: barSvc
          servicePort: 80
```



loodse

Ingress API: Name based virtual hosting

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
  - host: k8s.io
    http:
      paths:
      - backend:
          serviceName: k8sSvc
          servicePort: 80
  - host: j7a.io
    http:
      paths:
      - backend:
          serviceName: j7aSvc
          servicePort: 80
```



loodse

Ingress API

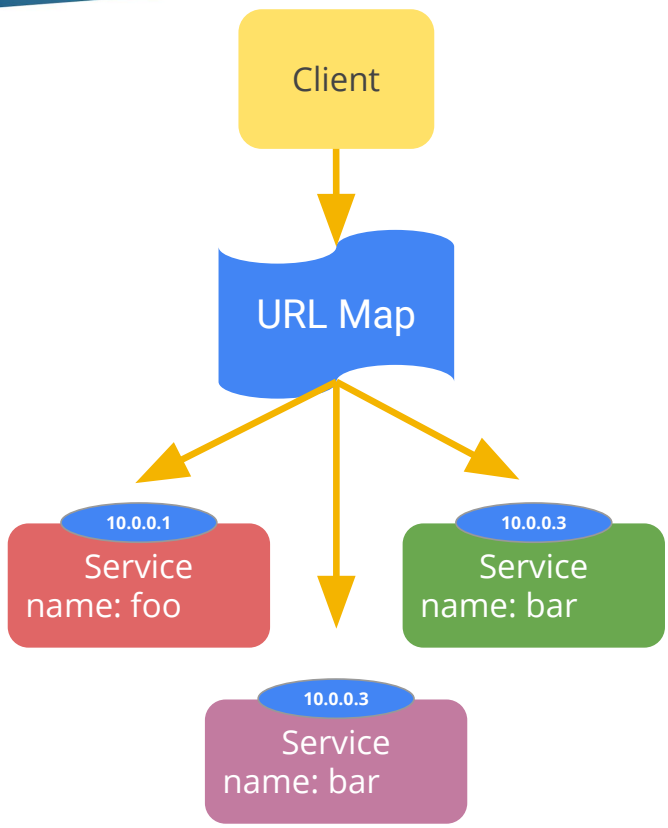
Services are assumed L3/L4

Lots of apps want HTTP/HTTPS

Ingress maps incoming traffic to backend services
- by HTTP host headers
- by HTTP URL paths

HAProxy, NGINX, AWS and GCE implementations in progress

Now with SSL!



loodse

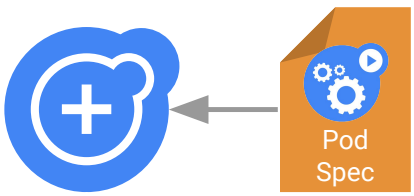
Deamonsets

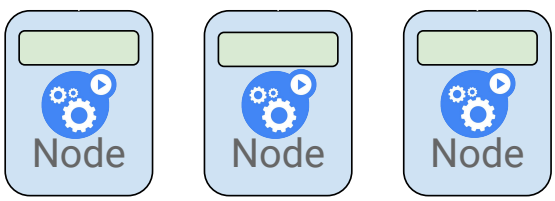
Ensure that a replica of a given pod runs on every node or a subset nodes

Like an RC but targets a set of nodes by selector

Examples:

- Agent based services: DataDog, Sysdig, etc
- Daemon process: Storage, Logs, Monitoring





loodse

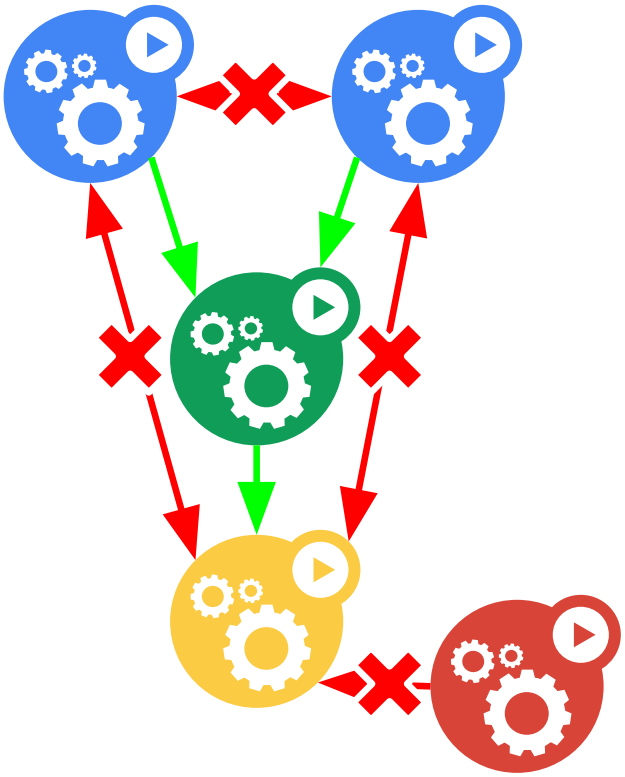
Network Policy

Describe the DAG of your app, enforce it in the network

Restrict Pod-to-Pod traffic or across Namespaces

Designed by the network SIG

- implementations for Calico, OpenShift, Romana, OpenContrail (so far)



loodse

Pod Security Policies (still beta)

A Pod Security Policy is a cluster-level resource that controls security sensitive aspects of the pod specification. The *PodSecurityPolicy* objects define a set of conditions that a pod must run with in order to be accepted into the system, as well as defaults for the related fields.

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: example
spec:
  privileged: false  # Don't allow privileged
pods!
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
  - '*'
```

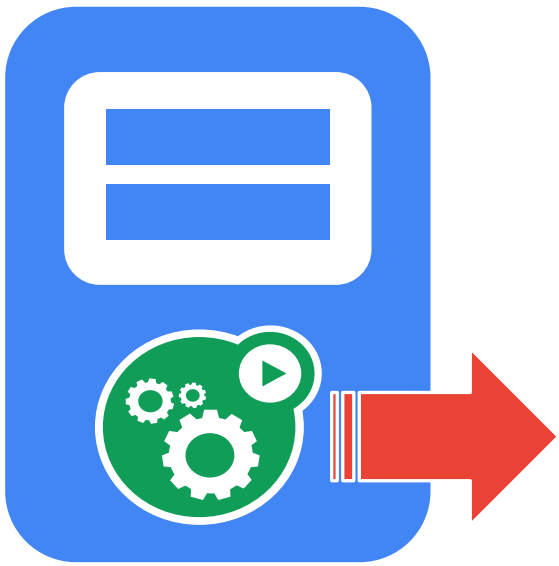
loodse

Node Drain

Goal: Evacuate a node for maintenance
- e.g. kernel upgrades

CLI: kubectl drain
- disallow scheduling
- allow grace period for pods to terminate
- kill pods

When done: kubectl uncordon
- the node rejoins the cluster



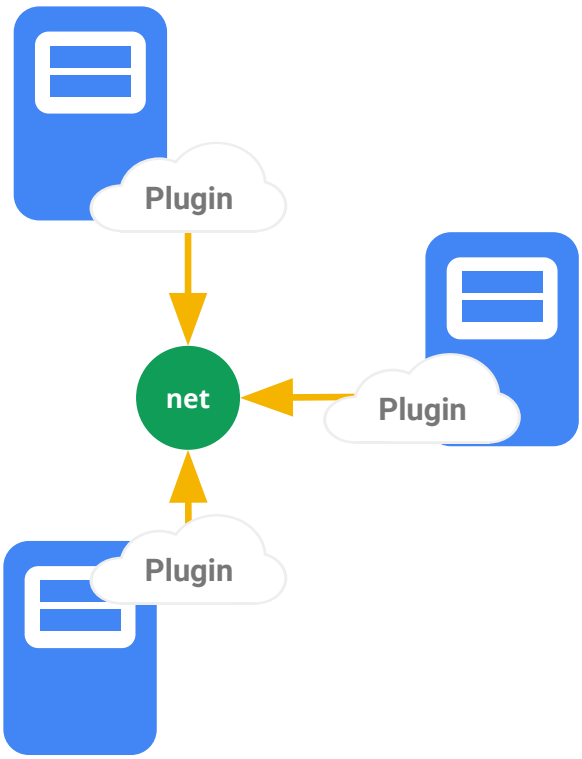
loodse

Network Plugins

Introduced in Kubernetes v1.0

Uses **CNI**
- Simple exec interface
- Not using Docker libnetwork
  - but can defer to Docker for networking

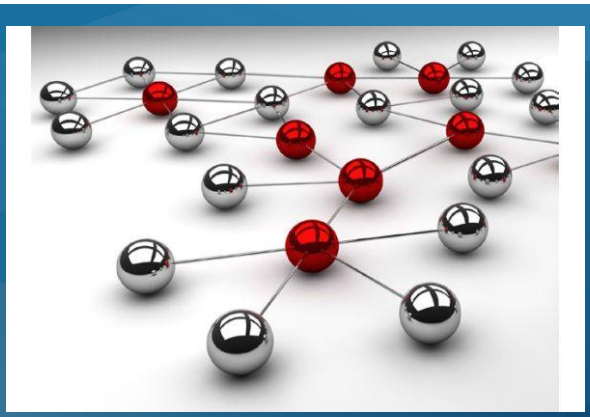
Cluster admins can customize their installs
- DHCP, MACVLAN, Flannel, custom



More...

- Cron (scheduled jobs)
- Custom metrics
- “Apply” a config (more declarative)
- Machine-generated Go clients (less deps!)
- Volume usage stats
- Multi-scheduler support
- Node affinity and anti-affinity
- More volume types
- Out-of-process volume plugin
- GUI
- Pod hostname and FQDN
- Better isolation
- Multi-cluster federation
- API federation
- Private Docker registry
- External DNS integration
- Volume classes and provisioning
- Node fencing
- DiY Cloud Provider plugins
- More container runtimes (e.g. Rkt, CRI-O)
- Better auth{n,z}
- Big data integrations
- Device scheduling (e.g. GPUs)

Cluster Addons



Monitoring

Run Prometheus (defacto standard)
- Uses Pull model
- Uses HTTP/HTTPS
- Metrics exported in plain text

Run Heapster as a pod in the cluster
- just another pod, no special access
- aggregate stats

Run Influx and Grafana in the cluster
- more pods
- alternately: store in Google Cloud Monitoring

Or plug in your own!
- e.g. Google Cloud Stackdriver, Instana APM

loodse

Logging

Run fluentd as a pod on each node
- gather logs from all containers
- export to elasticsearch

Run Elasticsearch as a pod in the cluster
- just another pod, no special access
- aggregate logs

Run Kibana in the cluster
- yet another pod
- alternately: store in Google Cloud Logging

Or plug in your own!
- e.g. Google Cloud Logging

loodse

DNS

Run SkyDNS as a pod in the cluster (old approach)

- kube2sky bridges Kubernetes API -> SkyDNS
- Tell kubelets about it (static service IP)

Strictly optional, but practically required

- LOTS of things depend on it
- Probably will become more integrated

CoreDNS - DNS and Service Discovery (new)

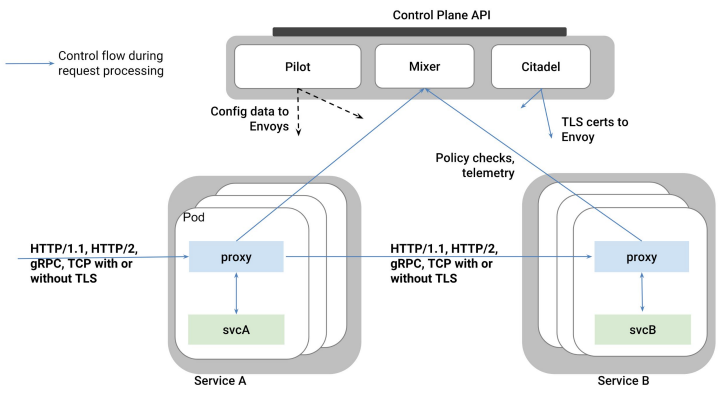
- Open Source DNS designed for Kubernetes
- Chains plugins. Each plugin performs a DNS function, e.g:
  - Kubernetes service discovery
  - Prometheus metrics
  - Rewriting queries

loodse

Service Mesh Istio

An open platform to connect, manage, and secure microservices:

- **Traffic Management**. Control the flow of traffic and API calls between services.
- **Service Identity and Security**. Provide services in the mesh with a verifiable identity and provide the ability to protect service traffic.
- **Policy Enforcement**. Apply organizational policy to the interaction between services. Policy changes are made by configuring the mesh, not by changing application code.
- **Telemetry**. Gain understanding of the dependencies between services and the nature and flow of traffic between them.
- **Integration and Customization**. The policy enforcement component can be extended and customized to integrate with existing solutions for ACLs, logging, monitoring, quotas, auditing and more

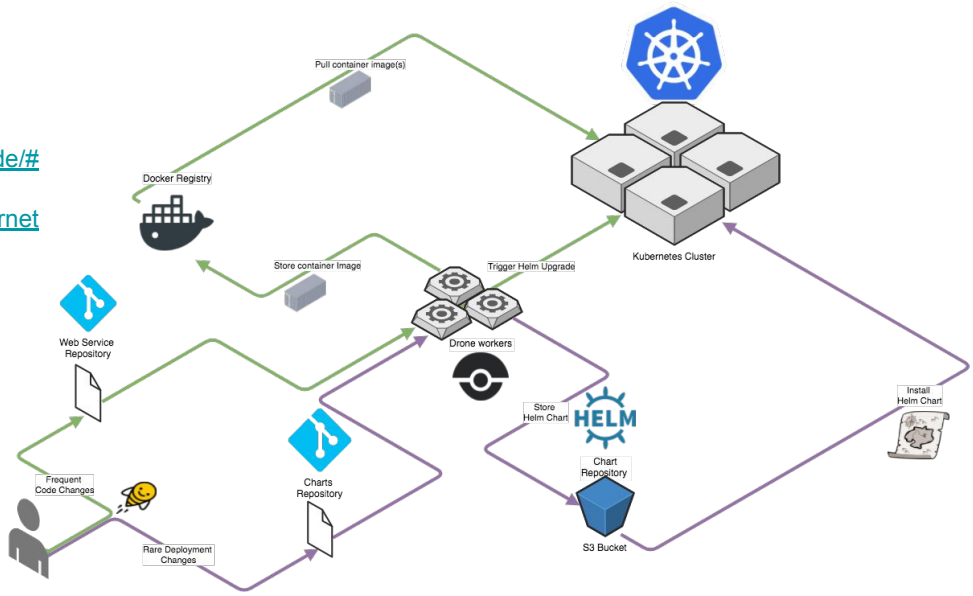


loodse

Package Manager HELM

How to start:

- https://github.com/loodse/helm-training
- https://docs.helm.sh/chart_template_guide/#the-chart-template-developer-s-guide
- https://www.katacoda.com/courses/kubernetes/helm-package-manager



Thank you for your attention

Contact
info@loodse.com
www.loodse.com

loodse

loodse

12 Factor App

1. Codebase
2. Dependencies
3. Config
4. Backing services
5. Build, release, run
6. Processes
7. Port binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin processes