

# Kubernetes

An Introduction

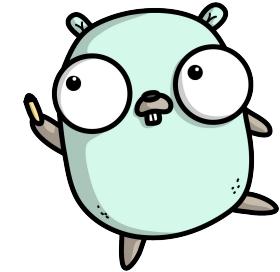




# Before We Begin

## Requirements:

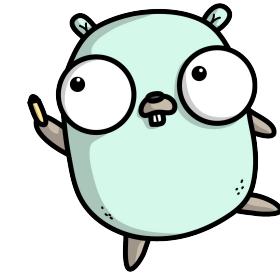
- **Minikube:**  
<https://github.com/kubernetes/minikube>
- **Virtualbox\*:**  
<https://www.virtualbox.org/wiki/Downloads>
- **kubectl:**  
<https://kubernetes.io/docs/tasks/tools/install-kubectl/>
- **k8s-intro-tutorials repo:**  
<https://github.com/mrbobbytables/k8s-intro-tutorials>



# Before We Begin



**Go here for install instructions**



<https://goo.gl/1HV3Tu>

# Kubernetes

An Introduction

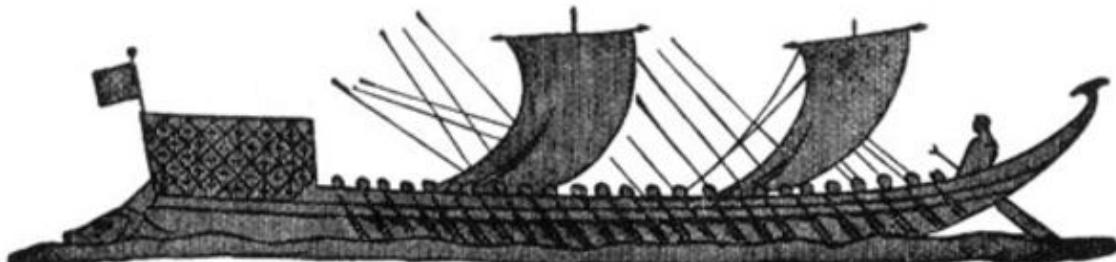


# Project Overview

# What Does “Kubernetes” Mean?



Greek for “pilot” or  
“Helmsman of a ship”



[Image Source](#)





# What is Kubernetes?

- Project that was spun out of Google as an open source container orchestration platform.
- Built from the lessons learned in the experiences of developing and running Google's Borg and Omega.
- Designed from the ground-up as a **loosely coupled** collection of components centered around deploying, maintaining and scaling workloads.



# What Does Kubernetes do?

- Known as the **linux kernel 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.



# 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.”



# What can Kubernetes REALLY do?

- Autoscale Workloads
- Blue/Green Deployments
- Fire off jobs and scheduled cronjobs
- Manage Stateless and Stateful Applications
- Provide native methods of service discovery
- Easily integrate and support 3rd party apps

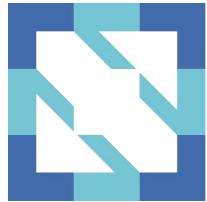
# Most Importantly...



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



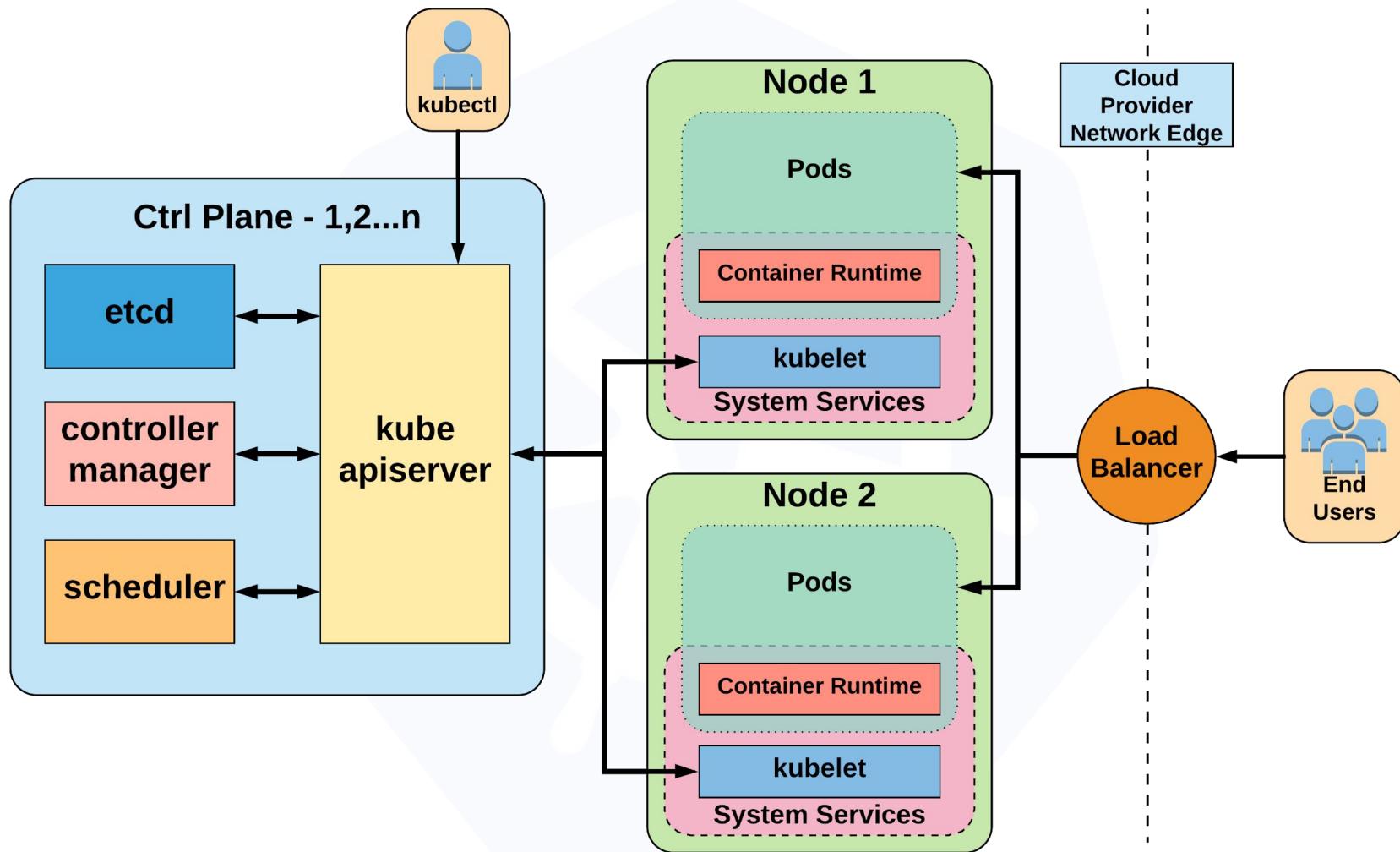
# Who “Manages” Kubernetes?



**CLOUD NATIVE  
COMPUTING FOUNDATION**

The CNCF is a child entity of the Linux Foundation and operates as a vendor neutral governance group.

# Architecture Overview



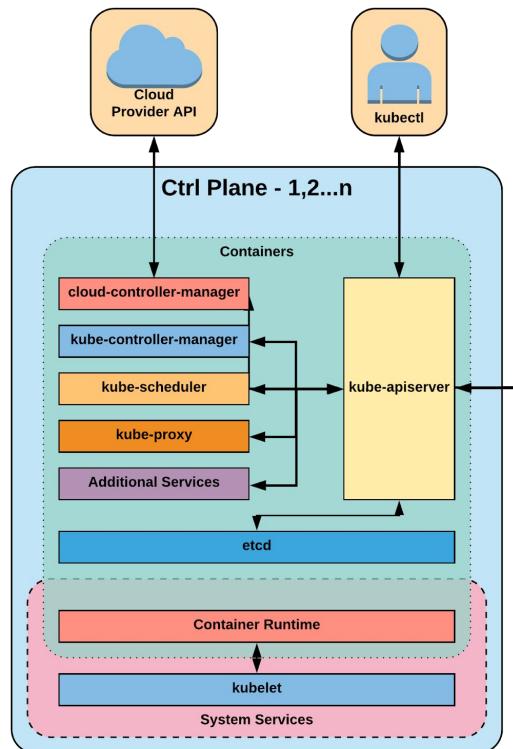
# Control Plane Components

Architecture Overview

# Control Plane Components



- kube-apiserver
- etcd
- kube-controller-manager
- kube-scheduler





# kube-apiserver

- Provides a forward facing REST interface into the kubernetes control plane and datastore.
- All clients and other applications interact with kubernetes **strictly** through the API Server.
- Acts as the gatekeeper to the cluster by handling authentication and authorization, request validation, mutation, and admission control in addition to being the front-end to the backing datastore.

# etcd



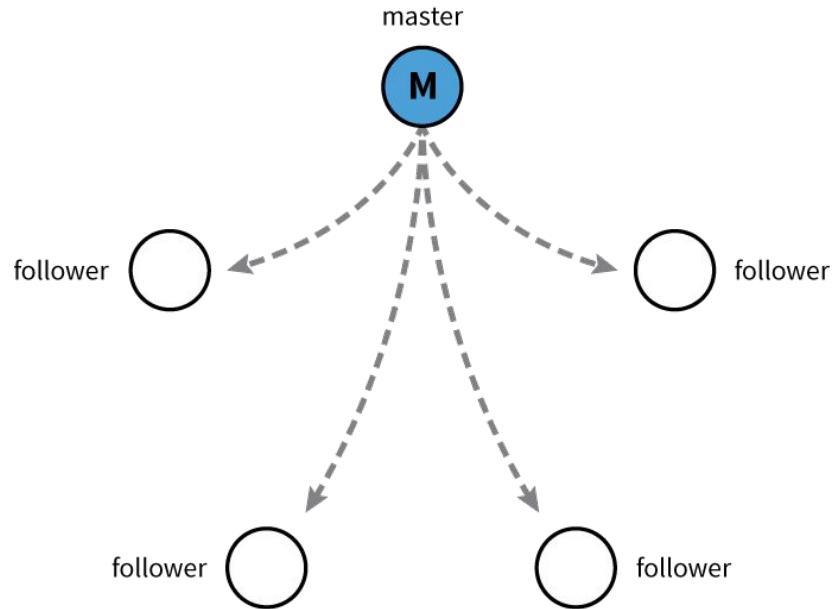
- etcd acts as the cluster datastore.
- Purpose in relation to Kubernetes is to provide a strong, consistent and highly available key-value store for persisting cluster state.
- Stores objects and config information.



# etcd



Uses “*Raft Consensus*” among a quorum of systems to create a fault-tolerant consistent “view” of the cluster.



<https://raft.github.io/>

[Image Source](#)



# kube-controller-manager

- Serves as the primary daemon that manages all core component control loops.
- Monitors the cluster state via the apiserver and **steers the cluster towards the desired state**.

List of core controllers:

<https://github.com/kubernetes/kubernetes/blob/master/cmd/kube-controller-manager/app/controllermanager.go#L344>

# kube-scheduler



- Verbose policy-rich engine that evaluates workload requirements and attempts to place it on a matching resource.
- Default scheduler uses bin packing.
- Workload Requirements can include: general hardware requirements, affinity/anti-affinity, labels, and other various custom resource requirements.

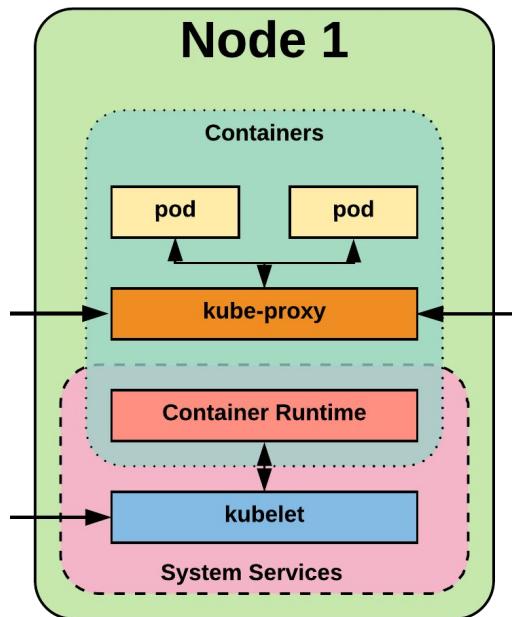
# Node Components

Architecture Overview

# Node Components



- kubelet
- kube-proxy
- Container Runtime Engine



# kubelet



- Acts as the node agent responsible for managing the lifecycle of every pod on its host.
- Kubelet understands YAML container manifests that it can read from several sources:
  - file path
  - HTTP Endpoint
  - etcd watch acting on any changes
  - HTTP Server mode accepting container manifests over a simple API.



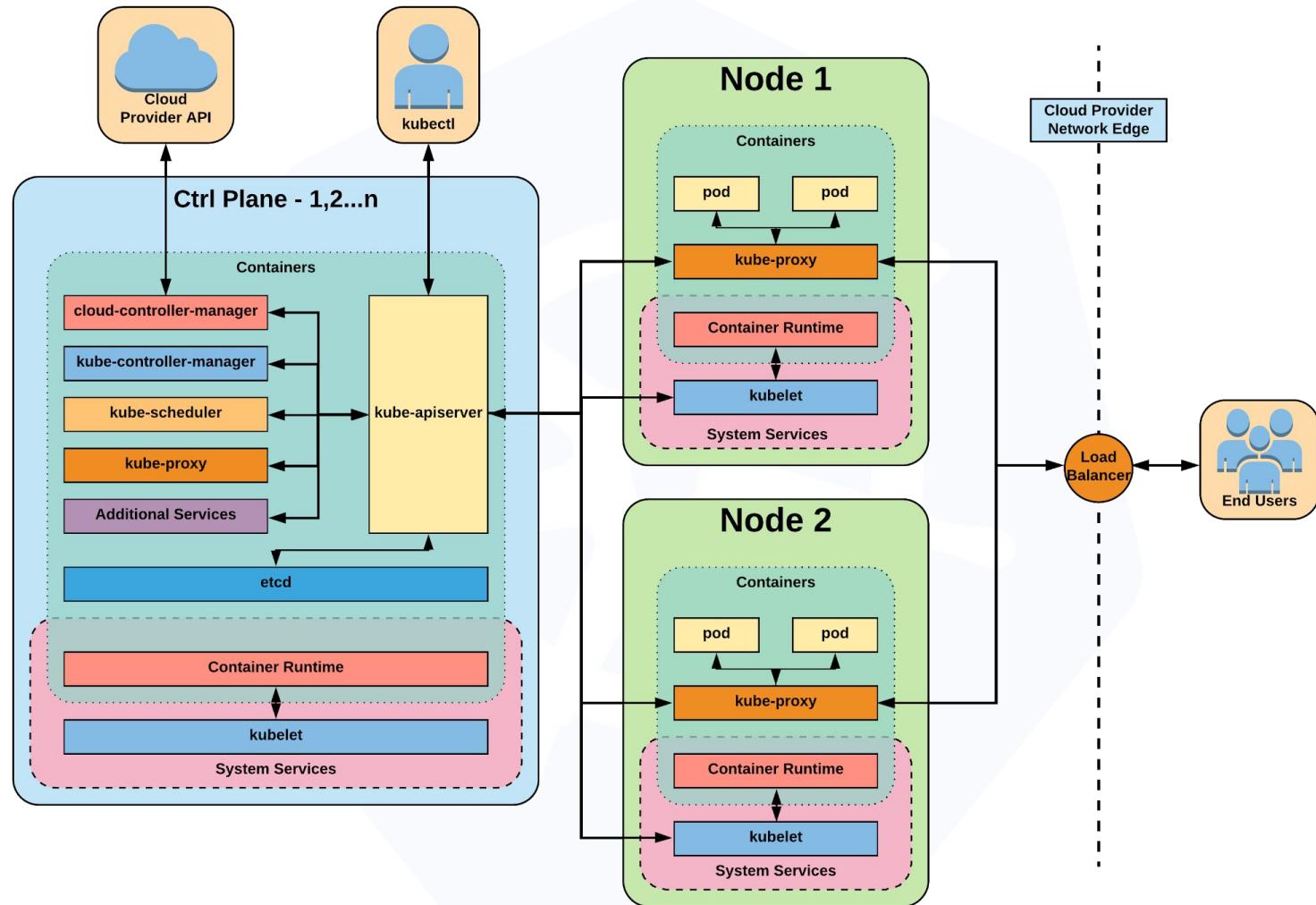
# kube-proxy

- Manages the network rules on each node.
- Performs connection forwarding or load balancing for Kubernetes cluster services.
- Available Proxy Modes:
  - Userspace
  - iptables
  - ipvs (default if supported)

# Container Runtime Engine



- A container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
  - Containerd (docker)
  - Cri-o
  - Rkt
  - Kata (formerly clear and hyper)
  - Virtlet (VM CRI compatible runtime)



# Optional Services

Architecture Overview



# cloud-controller-manager

- Daemon that provides cloud-provider specific knowledge and integration capability into the core control loop of Kubernetes.
- The controllers include Node, Route, Service, and add an additional controller to handle things such as **PersistentVolume** Labels.



# Cluster DNS

- Provides Cluster Wide DNS for Kubernetes Services.
  - Built on top of CoreDNS

# Kube Dashboard

A limited, general purpose web front end for the Kubernetes Cluster.



The screenshot shows the Kube Dashboard's Overview page. On the left is a sidebar with navigation links: Cluster, Namespaces, Nodes, Persistent Volumes, Roles, Storage Classes, Namespace (with a dropdown menu for 'All namespaces'), Overview (which is selected), Workloads, Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets, Discovery and Load Balancing, Ingresses, Services, Config and Storage, Config Maps, Persistent Volume Claims, Secrets, Settings, and About. The main content area has two charts: 'CPU usage' and 'Memory usage'. Below the charts are sections for 'Workloads Statuses' (showing green circles for Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, and Replication Controllers, all at 100.00%) and 'Cron Jobs' (listing 'hello' and 'midnight' with details like Namespace, Labels, Schedule, Active, Last Schedule, and Age). There are also sections for 'Daemon Sets', 'Deployments', and 'Secrets'.



# Metrics API Server

- Provides metrics for use with other Kubernetes Components.

# Networking

Architecture Overview

# Kubernetes Networking

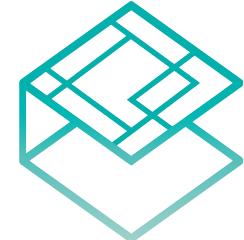


- **Pod Network**
  - Cluster-wide network used for pod-to-pod communication managed by a CNI (Container Network Interface) plugin.
- **Service Network**
  - Cluster-wide range of **Virtual IPs** managed by **kube-proxy** for service discovery.

# Container Network Interface (CNI)

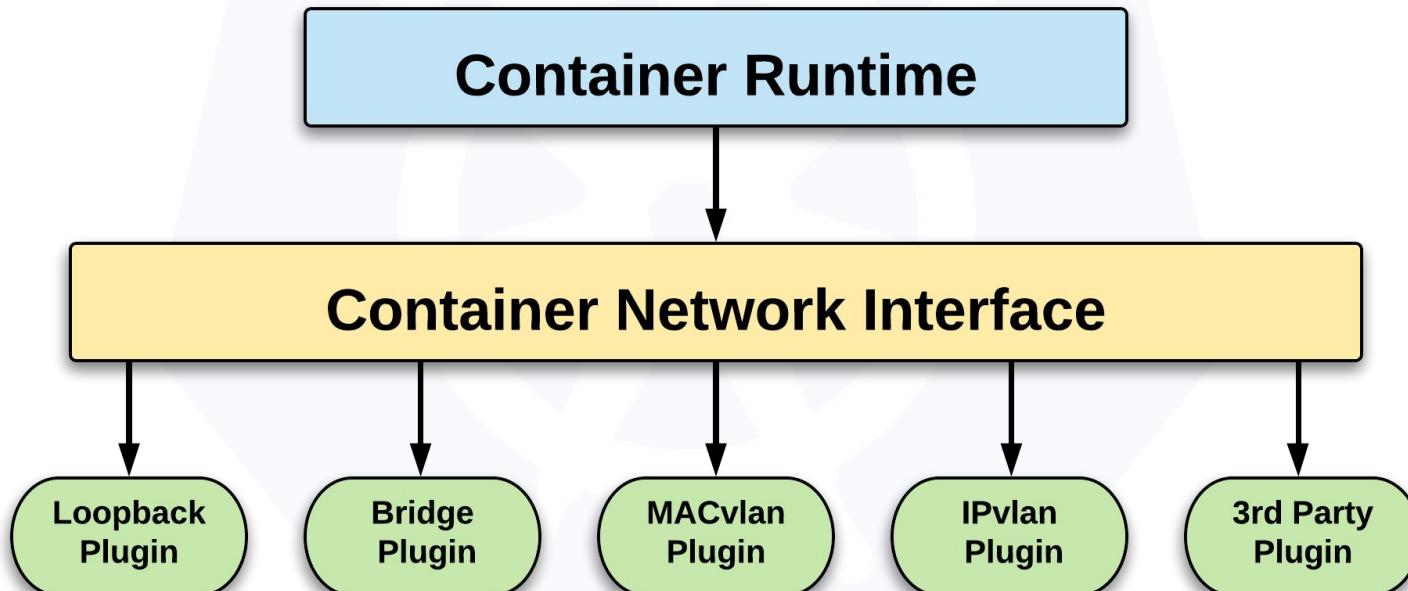


- Pod networking within Kubernetes is plumbed via the Container Network Interface (CNI).
- Functions as an interface between the container runtime and a **network implementation plugin**.
- CNCF Project
- Uses a simple JSON Schema.



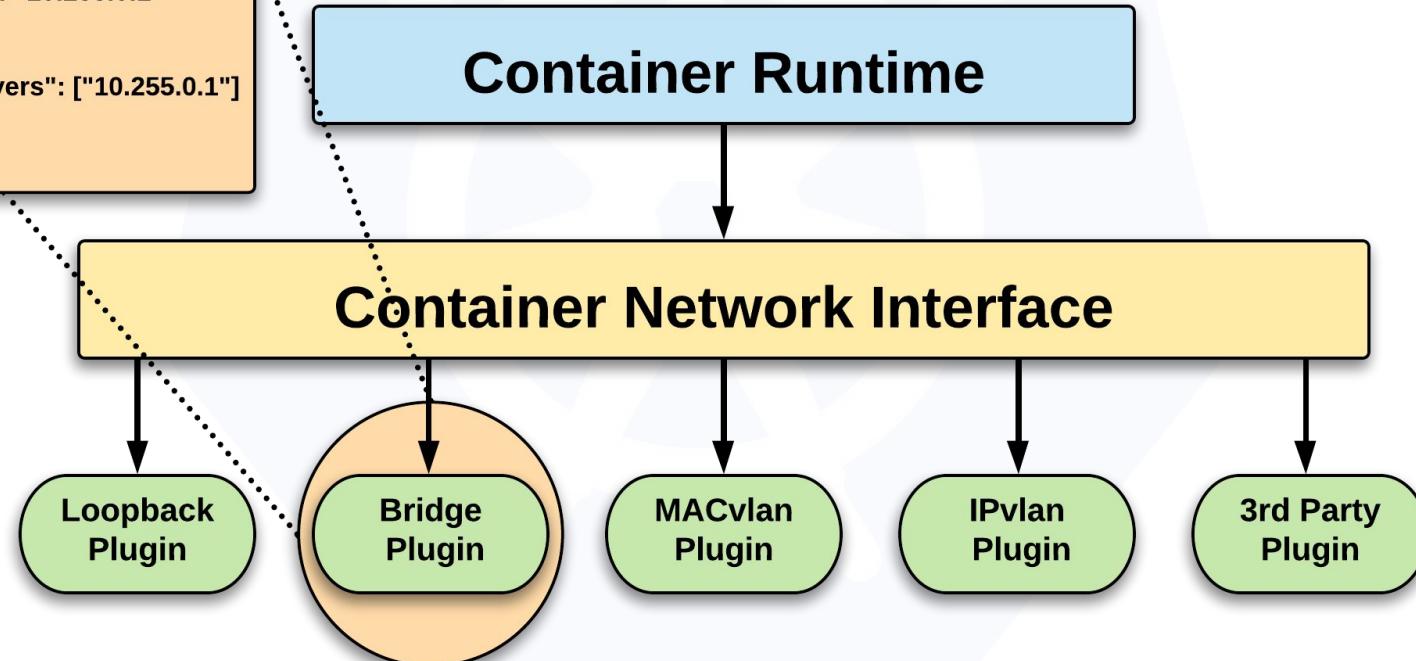
C N I

# CNI Overview



# CNI Overview

```
{  
    "cniVersion": "0.3.1",  
    "name": "examplenet",  
    "type": "bridge",  
    "bridge": "cni0",  
    "ipam": {  
        "type": "host-local",  
        "subnet": "10.255.0.0/16",  
        "gateway": "10.255.0.1"  
    },  
    "dns": {  
        "nameservers": ["10.255.0.1"]  
    }  
}
```



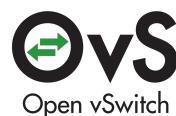
# CNI Plugins



- Amazon ECS
- Calico
- Cillium
- Contiv
- Contrail
- Flannel



- GCE
- kube-router
- Multus
- OpenVSwitch
- Romana
- Weave



# Fundamental Networking Rules



- All containers within a pod can communicate with each other unimpeded.
- All Pods can communicate with all other Pods without NAT.
- All nodes can communicate with all Pods (and vice-versa) without NAT.
- The IP that a Pod sees itself as is the same IP that others see it as.



# Fundamentals Applied

- **Container-to-Container**
  - Containers within a pod exist within the **same network namespace** and share an IP.
  - Enables intrapod communication over *localhost*.
- **Pod-to-Pod**
  - Allocated **cluster unique IP** for the duration of its life cycle.
  - Pods themselves are fundamentally ephemeral.



# Fundamentals Applied

- **Pod-to-Service**

- managed by **kube-proxy** and given a **persistent cluster unique IP**
- exists beyond a Pod's lifecycle.

- **External-to-Service**

- Handled by **kube-proxy**.
- Works in cooperation with a cloud provider or other external entity (load balancer).

# Concepts and Resources

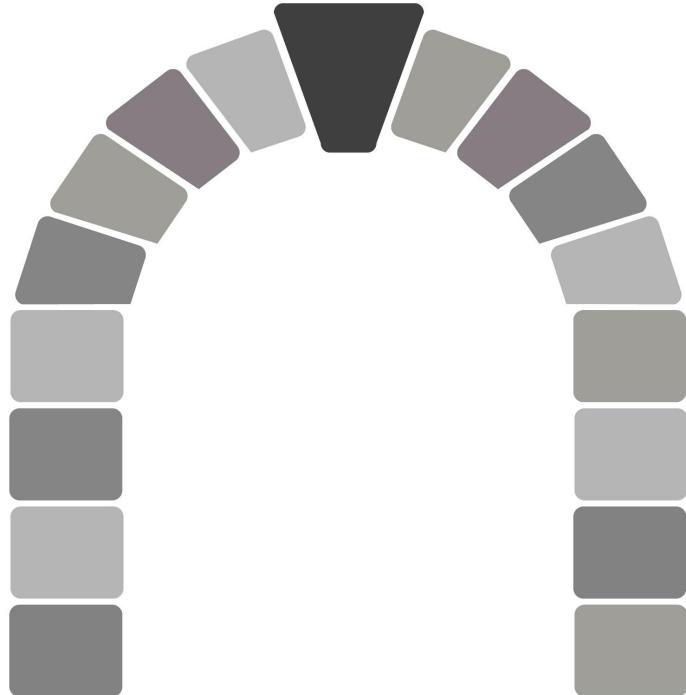
# The API and Object Model

Concepts and Resources

# API Overview



- The **REST API** is the true **keystone** of Kubernetes.
- **Everything** within the Kubernetes is as an **API Object**.



[Image Source](#)



# API Groups

- Designed to make it extremely simple to both understand and extend.
- An API Group is a **REST compatible path** that acts as the type descriptor for a Kubernetes object.
- Referenced within an object as the **apiVersion** and **kind**.

Format:

`/apis/<group>/<version>/<resource>`

Examples:

`/apis/apps/v1/deployments`

`/apis/batch/v1beta1/cronjobs`



# API Versioning

- Three tiers of API maturity levels.
- Also referenced within the object `apiVersion`.

Format:

`/apis/<group>/<version>/<resource>`

Examples:

`/apis/apps/v1/deployments`

`/apis/batch/v1beta1/cronjobs`

- **Alpha:** Possibly buggy, And may change. **Disabled by default.**
- **Beta:** Tested and considered stable. However API Schema may change. **Enabled by default.**
- **Stable:** Released, stable and API schema will not change. **Enabled by default.**



# Object Model

- Objects are a “*record of intent*” or a persistent entity that represent the desired state of the object within the cluster.
- All objects **MUST** have `apiVersion`, `kind`, and `pose` the nested fields `metadata.name`, `metadata.namespace`, and `metadata.uid`.



# Object Model Requirements

- `apiVersion`: Kubernetes API version of the Object
- `kind`: Type of Kubernetes Object
- `metadata.name`: Unique name of the Object
- `metadata.namespace`: Scoped environment name that the object belongs to (will default to current).
- `metadata.uid`: The (generated) uid for an object.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
  namespace: default
  uid: f8798d82-1185-11e8-94ce-080027b3c7a6
```



# Object Expression - YAML

- Files or other representations of Kubernetes Objects are generally represented in YAML.
- A “*Human Friendly*” data serialization standard.
- Uses white space (specifically spaces) alignment to denote ownership.
- Three basic data types:
  - **mappings** - hash or dictionary,
  - **sequences** - array or list
  - **scalars** - string, number, boolean etc

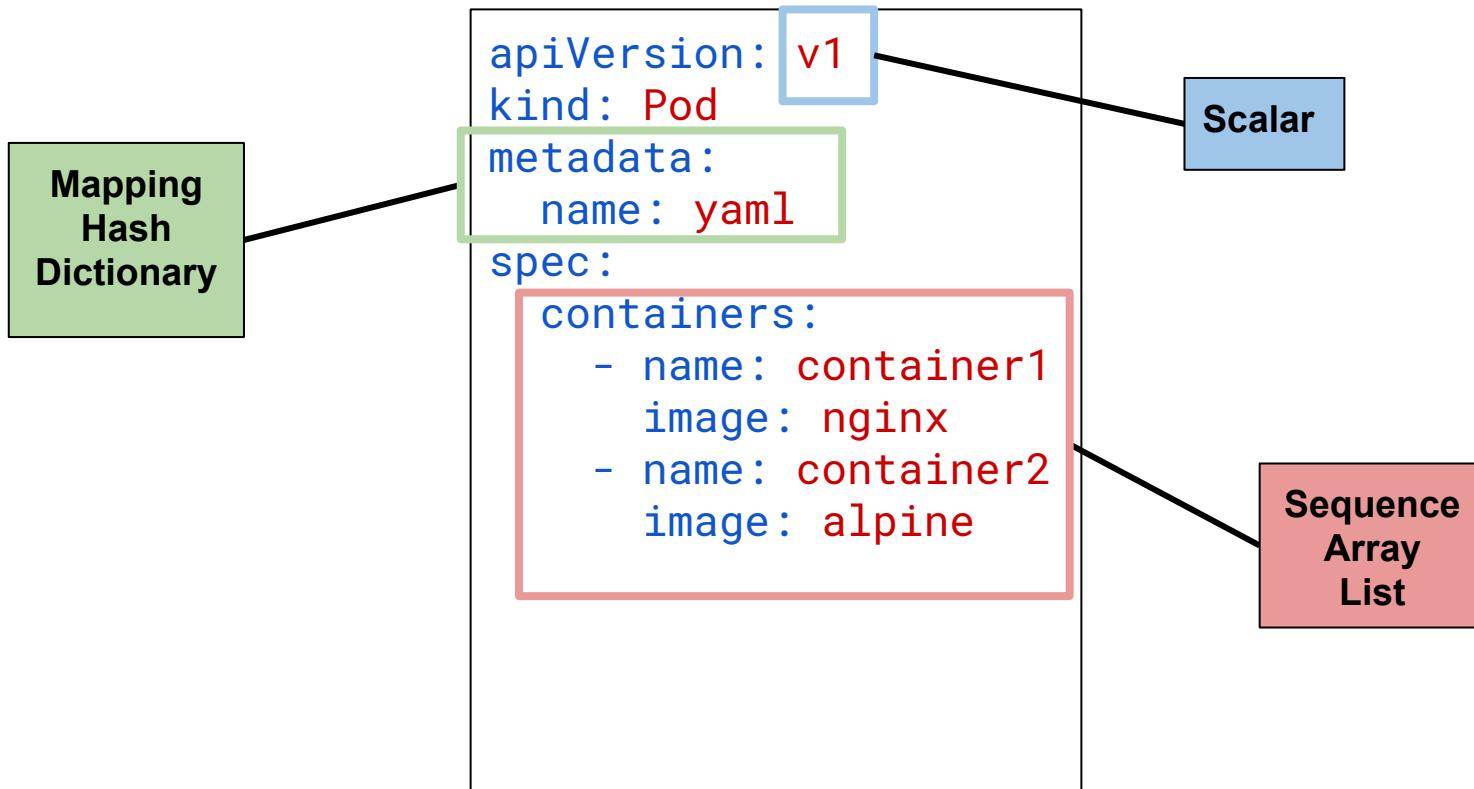


# Object Expression - YAML

```
apiVersion: v1
kind: Pod
metadata:
  name: yaml
spec:
  containers:
    - name: container1
      image: nginx
    - name: container2
      image: alpine
```



# Object Expression - YAML





# YAML vs JSON

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      ports:
        - containerPort: 80
```

```
{
  "apiVersion": "v1",
  "kind": "Pod",
  "metadata": {
    "name": "pod-example"
  },
  "spec": {
    "containers": [
      {
        "name": "nginx",
        "image": "nginx:stable-alpine",
        "ports": [ { "containerPort": 80 } ]
      }
    ]
  }
}
```



# Object Model - Workloads

- Workload related objects within Kubernetes have an additional two nested fields **spec** and **status**.
  - **spec** - Describes the **desired state** or **configuration** of the object to be created.
  - **status** - Is managed by Kubernetes and describes the **actual state** of the object and its history.

# Workload Object Example



## Example Object

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      ports:
        - containerPort: 80
```

## Example Status Snippet

```
status:
  conditions:
    - lastProbeTime: null
      lastTransitionTime: 2018-02-14T14:15:52Z
      status: "True"
      type: Ready
    - lastProbeTime: null
      lastTransitionTime: 2018-02-14T14:15:49Z
      status: "True"
      type: Initialized
    - lastProbeTime: null
      lastTransitionTime: 2018-02-14T14:15:49Z
      status: "True"
      type: PodScheduled
```

# Using the API

(aka, using the CLI)

# Kubectl



The Kubernetes command-line tool, kubectl, allows you to run commands against Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs

<https://kubernetes.io/docs/tasks/tools/install-kubectl/>

# Core Objects

- Namespaces
- Pods
- Labels
- Selectors
- Services

# Core Concepts



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

**Namespaces**

**Pods**

**Labels**

**Services**

**Selectors**

# Namespaces



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

```
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
```



# Default Namespaces

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

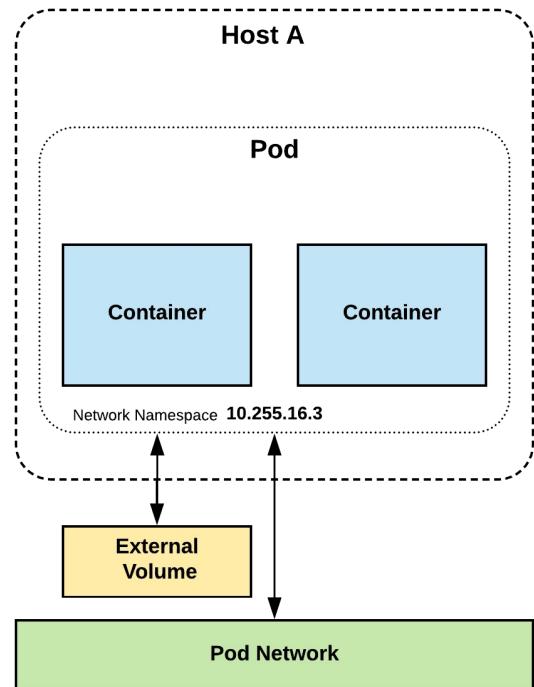
```
$ kubectl get ns --show-labels
```

NAME	STATUS	AGE	LABELS
default	Active	11h	<none>
kube-public	Active	11h	<none>
kube-system	Active	11h	<none>

# Pod



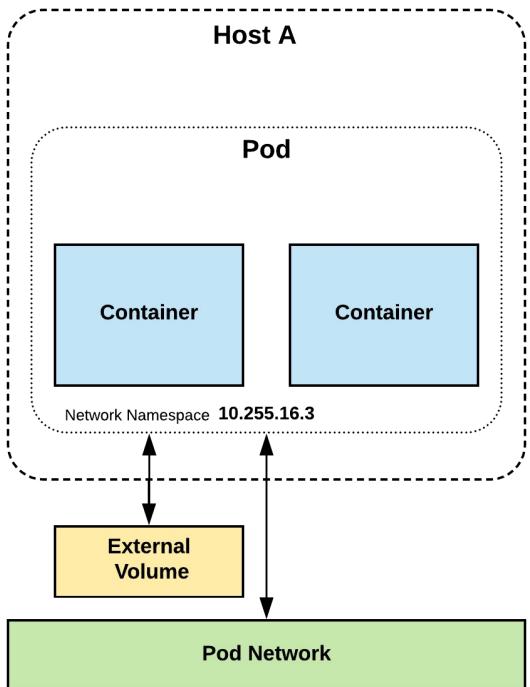
- **Atomic unit** or smallest “*unit of work*” of Kubernetes.
- Foundational building block of Kubernetes Workloads.
- Pods are one or more containers that share volumes, a network namespace, and are a part of a **single context**.



# Pods



They are  
also  
Ephemeral!



# Pod Examples



```
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      ports:
        - containerPort: 80
```

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-container-example
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      ports:
        - containerPort: 80
      volumeMounts:
        - name: html
          mountPath: /usr/share/nginx/html
    - name: content
      image: alpine:latest
      command: ["/bin/sh", "-c"]
      args:
        - while true; do
            date >> /html/index.html;
            sleep 5;
        done
      volumeMounts:
        - name: html
          mountPath: /html
      volumes:
        - name: html
          emptyDir: {}
```

# Key Pod Container Attributes



- `name` - The name of the container
- `image` - The container image
- `ports` - array of ports to expose.  
Can be granted a friendly name and protocol may be specified
- `env` - array of environment variables
- `command` - Entrypoint array (equiv to Docker `ENTRYPOINT`)
- `args` - Arguments to pass to the command (equiv to Docker `CMD`)

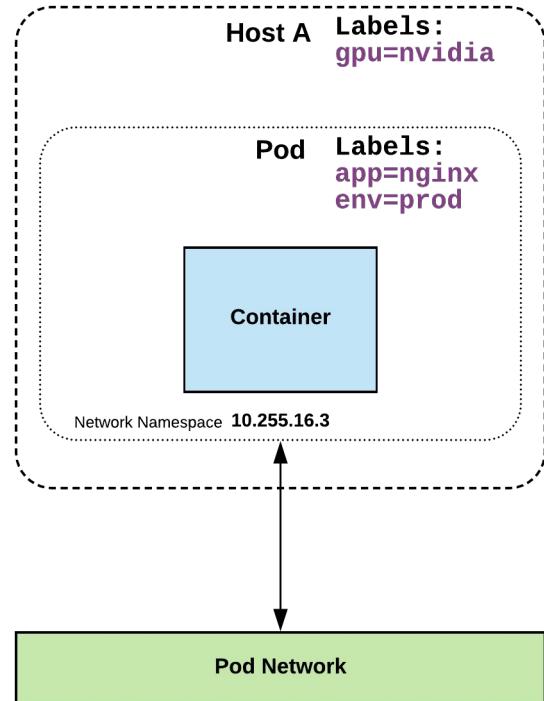
## Container

```
name: nginx
image: nginx:stable-alpine
ports:
- containerPort: 80
  name: http
  protocol: TCP
env:
- name: MYVAR
  value: isAwesome
command: ["/bin/sh", "-c"]
args: ["echo ${MYVAR}"]
```

# Labels



- key-value pairs that are used to identify, describe and group together related sets of objects or resources.
- **NOT** characteristic of uniqueness.
- Have a strict syntax with a slightly limited character set\*.

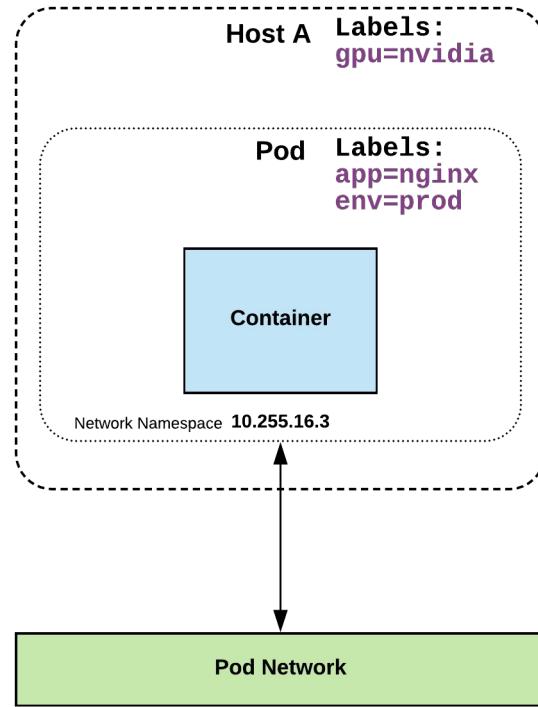


\* <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#syntax-and-character-set>

# Label Example



```
apiVersion: v1
kind: Pod
metadata:
  name: pod-label-example
labels:
  app: nginx
  env: prod
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
    - containerPort: 80
```



# Selectors

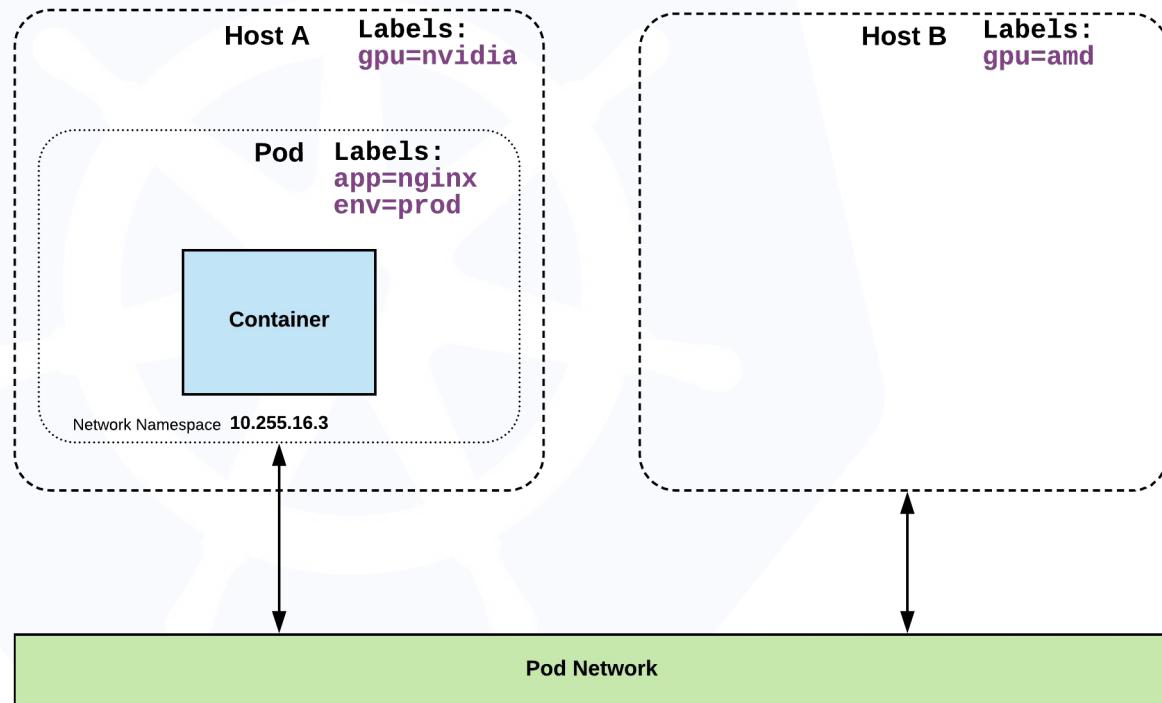
Selectors use labels to filter or select objects, and are used throughout Kubernetes.



```
apiVersion: v1
kind: Pod
metadata:
  name: pod-label-example
labels:
  app: nginx
  env: prod
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
    - containerPort: 80
nodeSelector:
  gpu: nvidia
```

# Selector Example

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-label-example
  labels:
    app: nginx
    env: prod
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
    - containerPort: 80
  nodeSelector:
    gpu: nvidia
```



# Selector Types



**Equality based** selectors allow for simple filtering (=, ==, or !=).

```
selector:  
  matchLabels:  
    gpu: nvidia
```

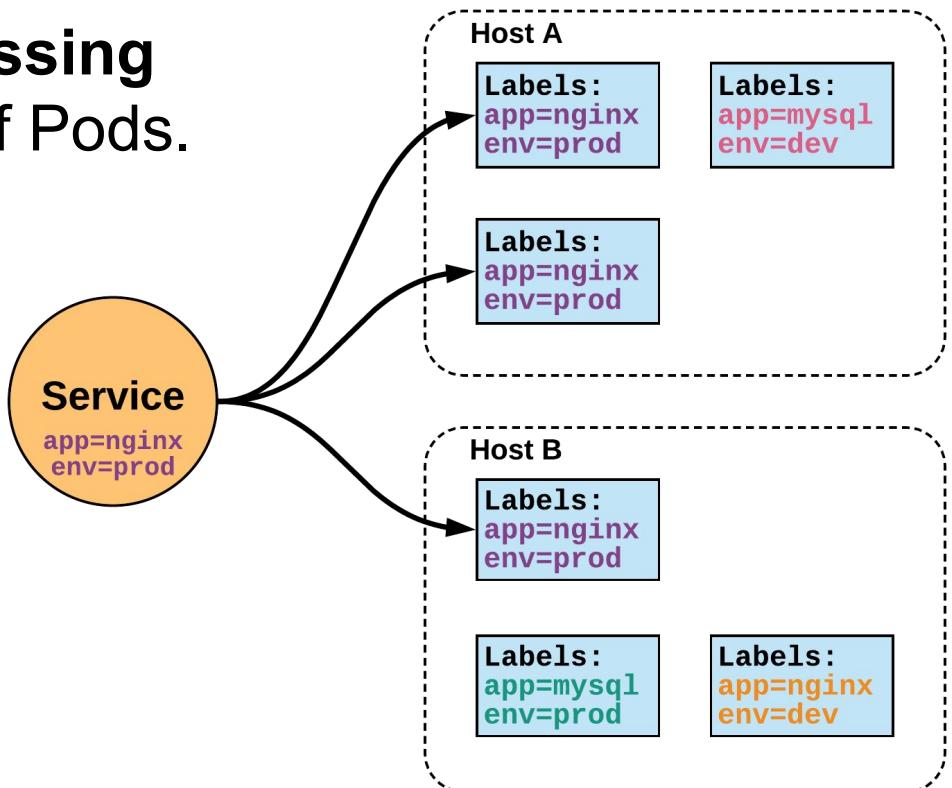
**Set-based** selectors are supported on a limited subset of objects. However, they provide a method of filtering on a set of values, and supports multiple operators including: **in**, **notin**, and **exist**.

```
selector:  
  matchExpressions:  
    - key: gpu  
      operator: in  
      values: ["nvidia"]
```

# Services



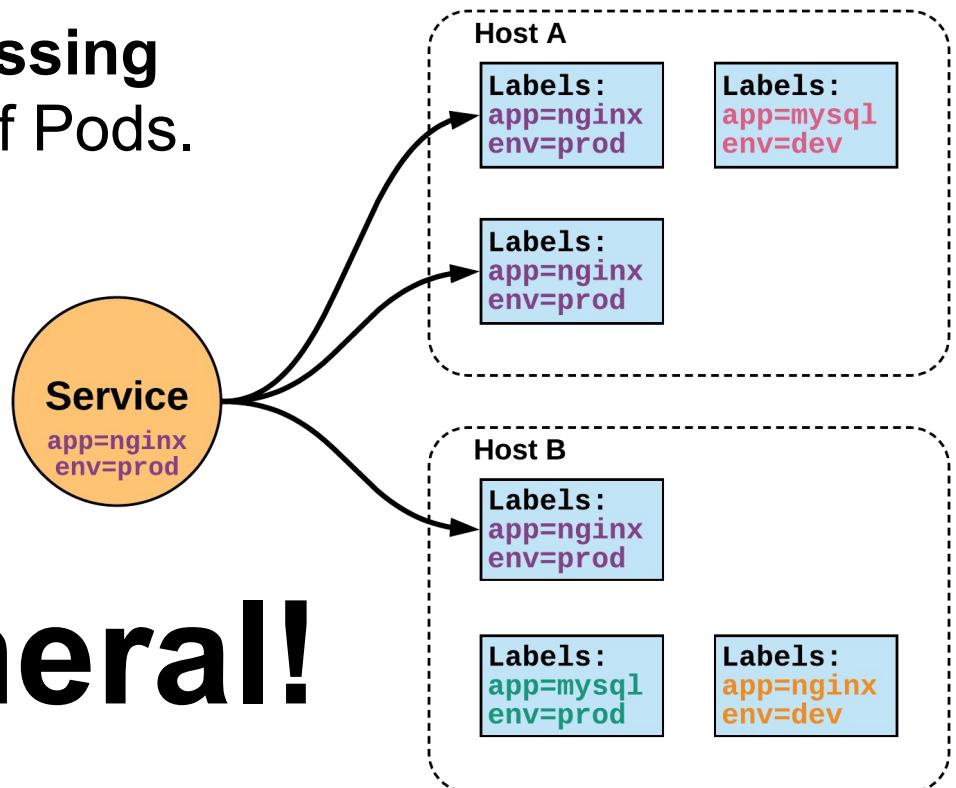
- **Unified method of accessing** the exposed workloads of Pods.
- **Durable resource**
  - static cluster IP
  - static namespaced DNS name



# Services



- **Unified method of accessing** the exposed workloads of Pods.
- **Durable resource**
  - static cluster IP
  - static namespaced DNS name



# NOT Ephemerall!



# Services

- **Unified method of accessing** the exposed workloads of Pods.
- **Durable resource** (unlike Pods)
  - static cluster-unique IP
  - static namespaced DNS name

**<service name>. <namespace>. svc.cluster.local**

# Services



- Target Pods using **equality based selectors**.
- Uses **kube-proxy** to provide simple load-balancing.
- **kube-proxy** acts as a daemon that creates **local entries** in the host's iptables for every service.



# Service Types

There are 4 major service types:

- **ClusterIP** (default)
- **NodePort**
- **LoadBalancer**
- **ExternalName**

# ClusterIP Service



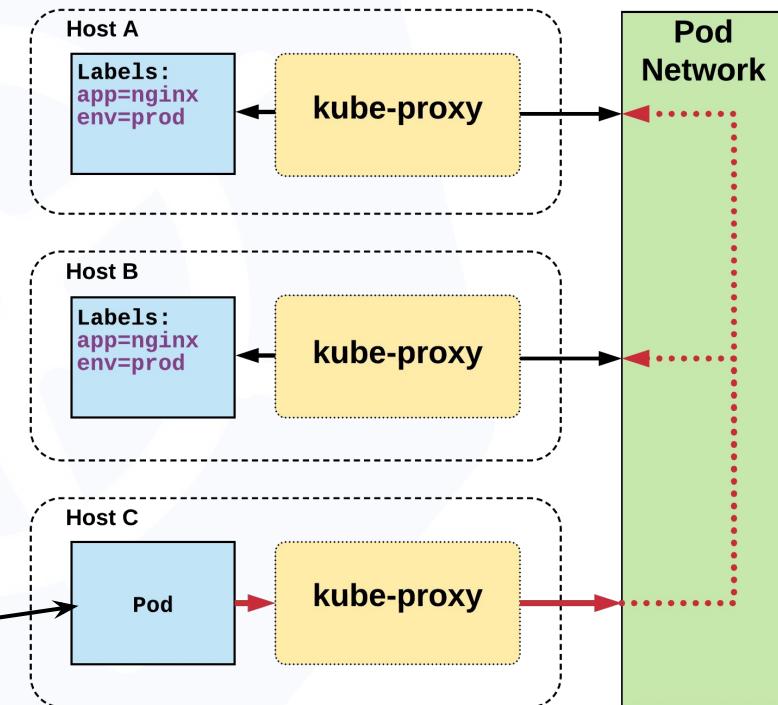
**ClusterIP** services exposes a service on a strictly cluster internal virtual IP.

```
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  selector:
    app: nginx
    env: prod
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

# Cluster IP Service

```
Name: example-prod  
Selector: app=nginx,env=prod  
Type: ClusterIP  
IP: 10.96.28.176  
Port: <unset> 80/TCP  
TargetPort: 80/TCP  
Endpoints: 10.255.16.3:80,  
           10.255.16.4:80
```

```
/ # nslookup example-prod.default.svc.cluster.local  
Name: example-prod.default.svc.cluster.local  
Address 1: 10.96.28.176 example-prod.default.svc.cluster.local
```



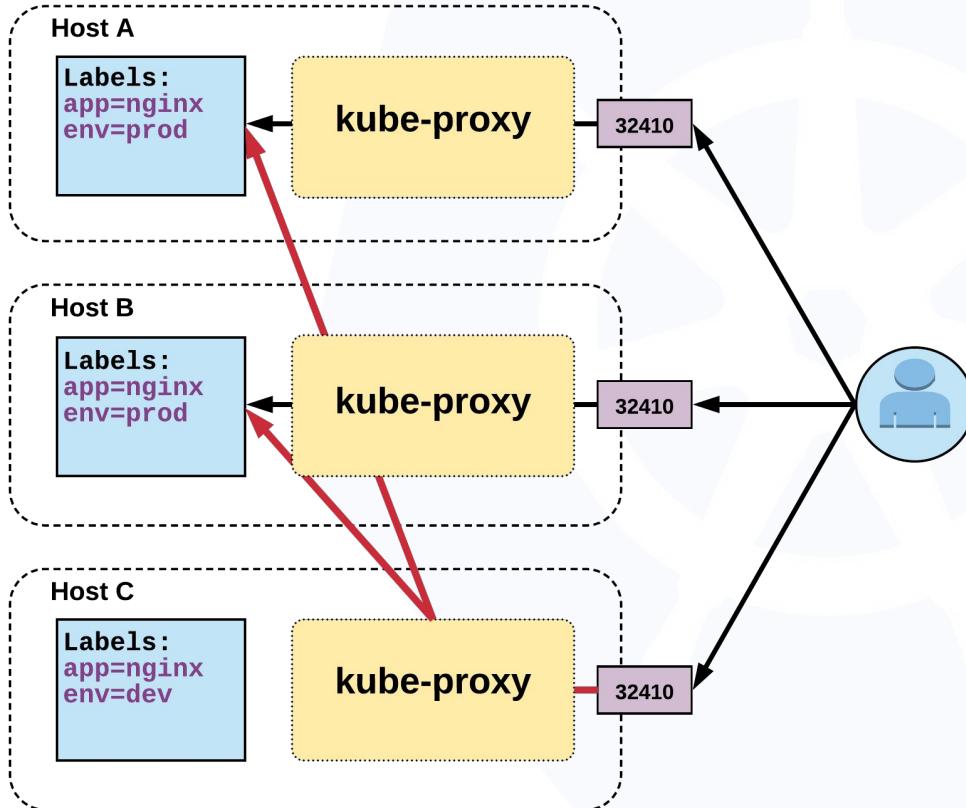
# NodePort Service



- **NodePort** services extend the **ClusterIP** service.
- Exposes a port on every node's IP.
- Port can either be statically defined, or dynamically taken from a range between 30000-32767.

```
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: NodePort
  selector:
    app: nginx
    env: prod
  ports:
    - nodePort: 32410
      protocol: TCP
      port: 80
      targetPort: 80
```

# NodePort Service



Name:	example-prod
Selector:	app=nginx, env=prod
Type:	NodePort
IP:	10.96.28.176
Port:	<unset> 80/TCP
TargetPort:	80/TCP
NodePort:	<unset> 32410/TCP
Endpoints:	10.255.16.3:80, 10.255.16.4:80

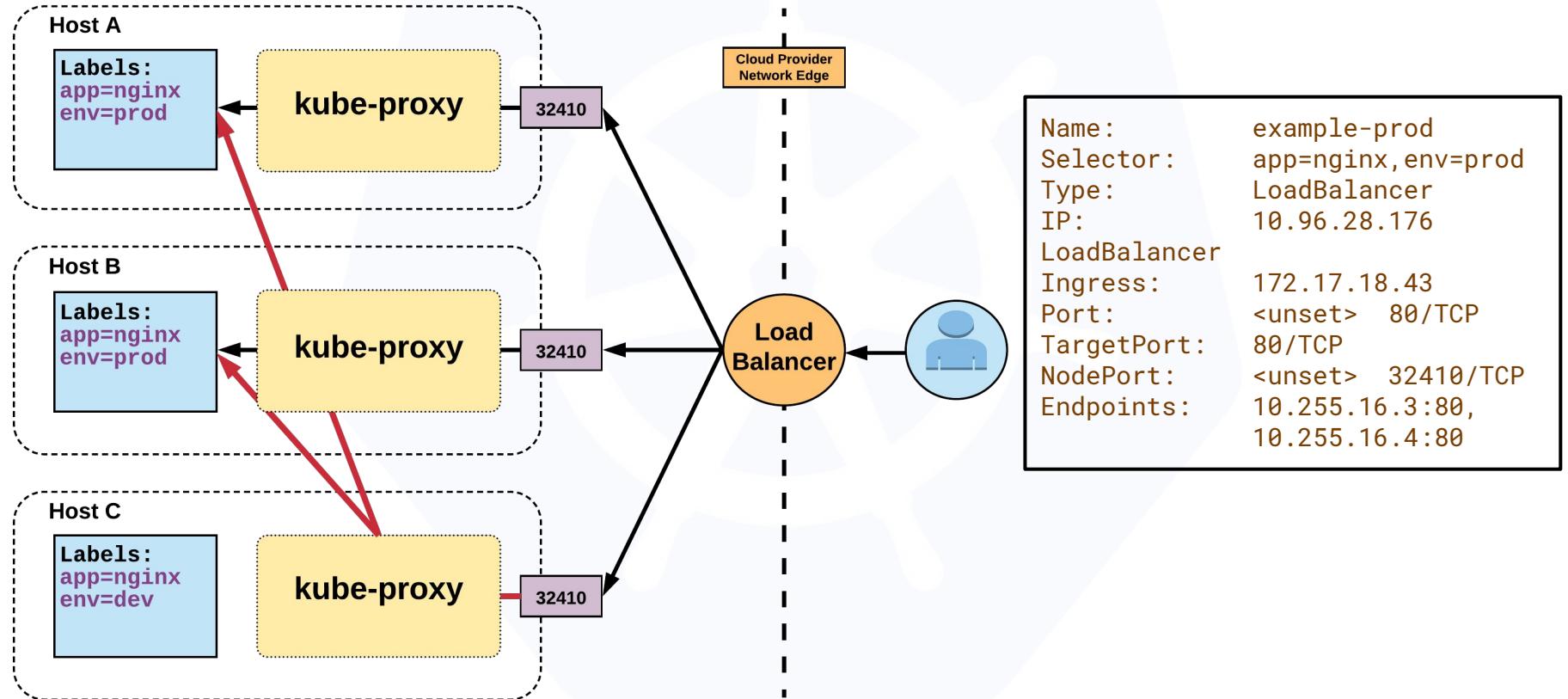
# LoadBalancer Service



- **LoadBalancer** services extend **NodePort**.
- Works in conjunction with an external system to map a cluster external IP to the exposed service.

```
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: LoadBalancer
  selector:
    app: nginx
    env: prod
  ports:
    protocol: TCP
    port: 80
    targetPort: 80
```

# LoadBalancer Service





# ExternalName Service

- **ExternalName** is used to reference endpoints **OUTSIDE** the cluster.
- Creates an internal **CNAME** DNS entry that aliases another.

```
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: ExternalName
  spec:
    externalName: example.com
```

# Workloads

- **ReplicaSet**
- **Deployment**
- **DaemonSet**
- **StatefulSet**
- **Job**
- **CronJob**



# Workloads

Workloads within Kubernetes are higher level objects that manage Pods or other higher level objects.

In **ALL CASES** a Pod Template is included, and acts the base tier of management.



# Pod Template

- Workload Controllers manage instances of Pods based off a provided template.
- Pod Templates are Pod specs with limited metadata.
- Controllers use Pod Templates to make actual pods.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
  labels:
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx
```

```
template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
      - name: nginx
        image: nginx
```



# ReplicaSet

- Primary method of managing pod replicas and their lifecycle.
- Includes their scheduling, scaling, and deletion.
- Their job is simple: **Always ensure the desired number of pods are running.**



# ReplicaSet



- **replicas**: The desired number of instances of the Pod.
- **selector**: The label selector for the **ReplicaSet** will manage **ALL** Pod instances that it targets; whether it's desired or not.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-example
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  template:
    <pod template>
```

# ReplicaSet



```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-example
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  template:
    metadata:
      labels:
        app: nginx
        env: prod
    spec:
      containers:
        - name: nginx
          image: nginx:stable-alpine
          ports:
            - containerPort: 80
```

```
$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
rs-example-9l4dt   1/1    Running   0          1h
rs-example-b7bcg   1/1    Running   0          1h
rs-example-mk112   1/1    Running   0          1h
```

```
$ kubectl describe rs rs-example
Name:         rs-example
Namespace:    default
Selector:    app=nginx,env=prod
Labels:      app=nginx
             env=prod
Annotations: <none>
Replicas:    3 current / 3 desired
Pods Status: 3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=nginx
           env=prod
  Containers:
    nginx:
      Image:      nginx:stable-alpine
      Port:       80/TCP
      Environment: <none>
      Mounts:     <none>
      Volumes:    <none>
  Events:
    Type  Reason          Age   From           Message
    ----  -----          ----  ----
    Normal SuccessfulCreate 16s   replicaset-controller  Created pod: rs-example-mk112
    Normal SuccessfulCreate 16s   replicaset-controller  Created pod: rs-example-b7bcg
    Normal SuccessfulCreate 16s   replicaset-controller  Created pod: rs-example-9l4dt
```

# Deployment



- Declarative method of managing Pods via **ReplicaSets**.
- Provide rollback functionality and update control.
- Updates are managed through the **pod-template-hash** label.
- Each iteration creates a unique label that is assigned to both the **ReplicaSet** and subsequent Pods.



# Deployment



- **revisionHistoryLimit**: The number of previous iterations of the Deployment to retain.
- **strategy**: Describes the method of updating the Pods based on the **type**. Valid options are **Recreate** or **RollingUpdate**.
  - **Recreate**: All existing Pods are killed before the new ones are created.
  - **RollingUpdate**: Cycles through updating the Pods according to the parameters: **maxSurge** and **maxUnavailable**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-example
spec:
  replicas: 3
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    <pod template>
```

# RollingUpdate Deployment

Updating pod template generates a new **ReplicaSet** revision.

```
R1 pod-template-hash:  
676677fff  
R2 pod-template-hash:  
54f7ff7d6d
```

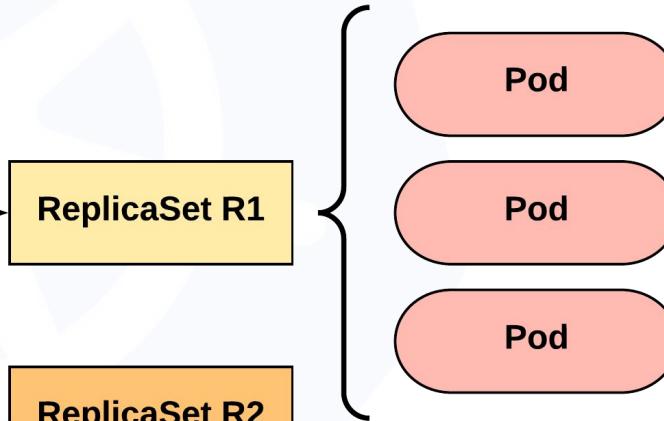
```
Deployment  
Revision 1
```

```
ReplicaSet R1
```

```
ReplicaSet R2
```

```
$ kubectl get replicaset  
NAME          DESIRED  CURRENT  READY   AGE  
mydep-6766777fff  3        3        3      5h
```

```
$ kubectl get pods  
NAME          READY   STATUS    RESTARTS  AGE  
mydep-6766777fff-9r2zn  1/1     Running   0        5h  
mydep-6766777fff-hsfz9  1/1     Running   0        5h  
mydep-6766777fff-sjxhf  1/1     Running   0        5h
```



# RollingUpdate Deployment

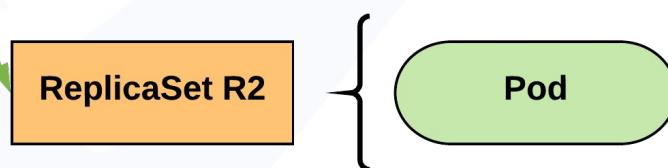
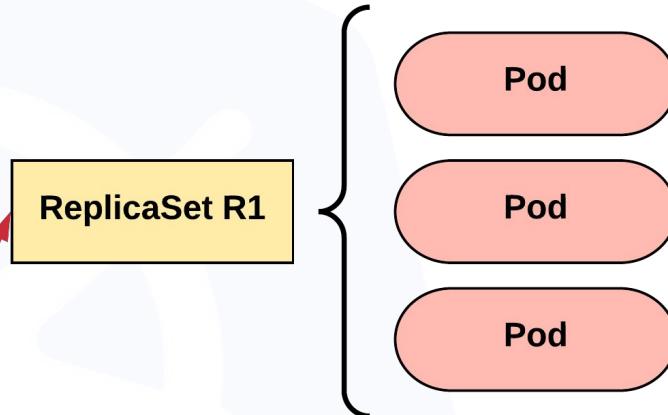
New **ReplicaSet** is initially scaled up based on [maxSurge](#).

```
R1 pod-template-hash:  
676677fff  
R2 pod-template-hash:  
54f7ff7d6d
```

Deployment  
Revision 2

```
$ kubectl get replicaset  
NAME          DESIRED  CURRENT  READY   AGE  
mydep-54f7ff7d6d  1        1        1      5s  
mydep-6766777fff 2        3        3      5h
```

```
$ kubectl get pods  
NAME          READY   STATUS    RESTARTS   AGE  
mydep-54f7ff7d6d-9gv1l  1/1     Running   0          2s  
mydep-6766777fff-9r2zn  1/1     Running   0          5h  
mydep-6766777fff-hsfz9  1/1     Running   0          5h  
mydep-6766777fff-sjxhf  1/1     Running   0          5h
```



# RollingUpdate Deployment

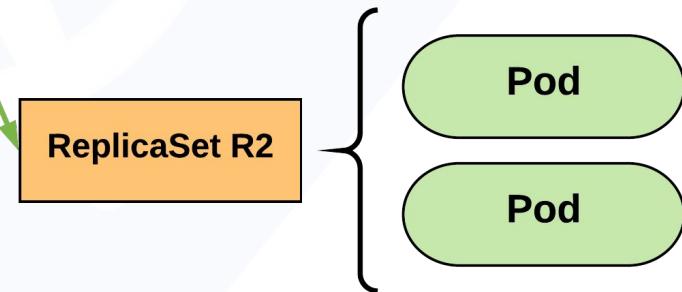
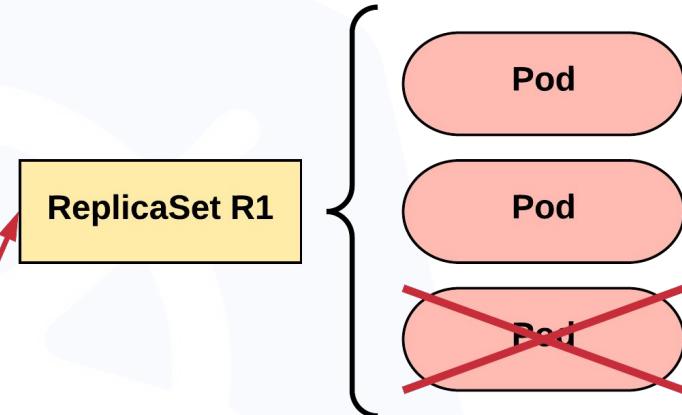
Phase out of old Pods managed by **maxSurge** and **maxUnavailable**.

```
R1 pod-template-hash:  
676677fff  
R2 pod-template-hash:  
54f7ff7d6d
```

Deployment  
Revision 2

```
$ kubectl get replicaset  
NAME          DESIRED  CURRENT  READY   AGE  
mydep-54f7ff7d6d  2        2        2      8s  
mydep-676677fff  2        2        2      5h
```

```
$ kubectl get pods  
NAME          READY   STATUS    RESTARTS   AGE  
mydep-54f7ff7d6d-9gv1l  1/1     Running   0          5s  
mydep-54f7ff7d6d-cqv1q  1/1     Running   0          2s  
mydep-676677fff-9r2zn  1/1     Running   0          5h  
mydep-676677fff-hsfz9  1/1     Running   0          5h
```



# RollingUpdate Deployment

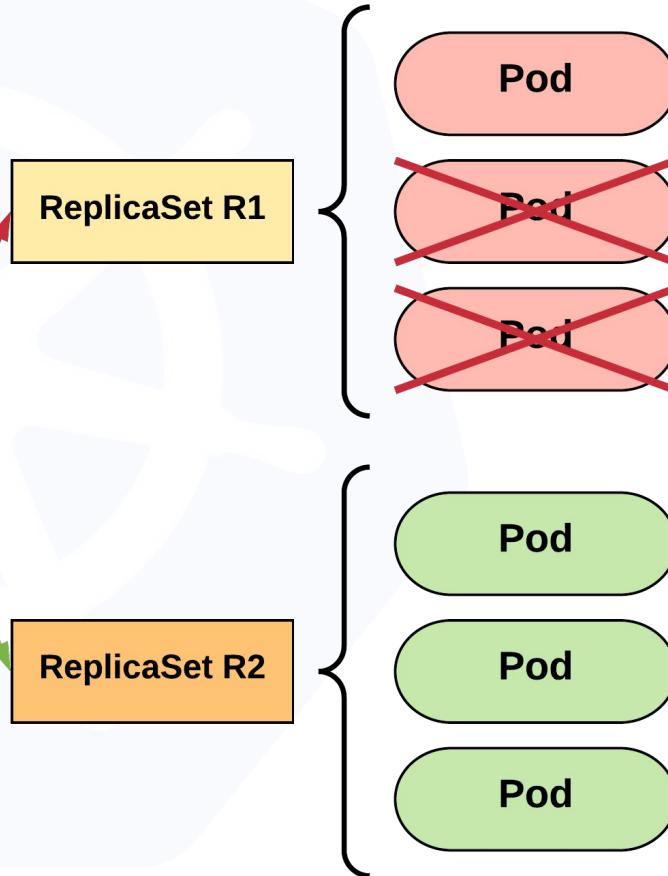
Phase out of old Pods managed by **maxSurge** and **maxUnavailable**.

```
R1 pod-template-hash:  
676677fff  
R2 pod-template-hash:  
54f7ff7d6d
```

Deployment  
Revision 2

```
$ kubectl get replicaset  
NAME          DESIRED  CURRENT  READY   AGE  
mydep-54f7ff7d6d  3        3        3      10s  
mydep-676677fff  0        1        1      5h
```

```
$ kubectl get pods  
NAME          READY   STATUS    RESTARTS   AGE  
mydep-54f7ff7d6d-9gv1l  1/1     Running   0          7s  
mydep-54f7ff7d6d-cqvlq  1/1     Running   0          5s  
mydep-54f7ff7d6d-gccr6  1/1     Running   0          2s  
mydep-676677fff-9r2zn  1/1     Running   0          5h
```



# RollingUpdate Deployment

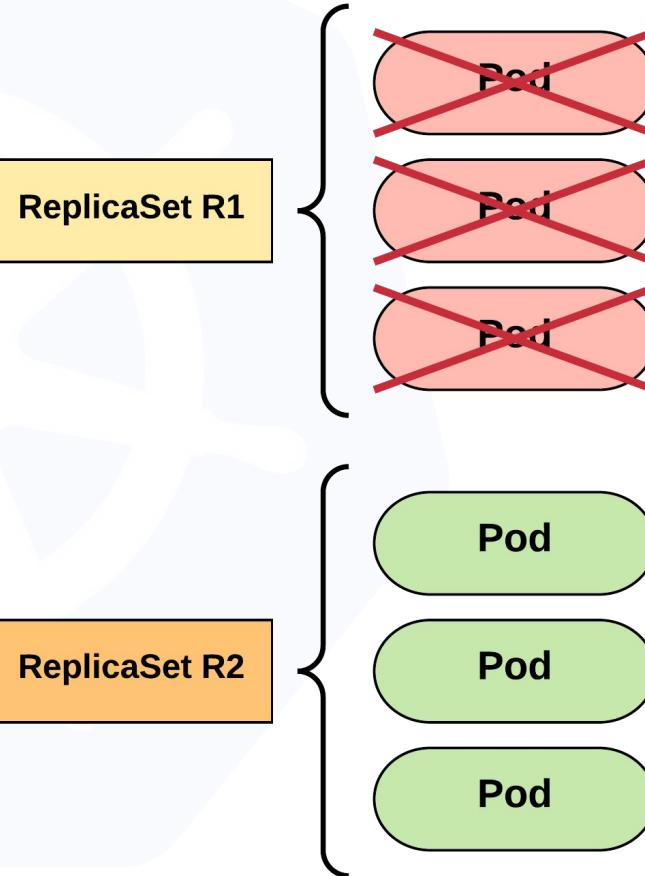
Phase out of old Pods managed by **maxSurge** and **maxUnavailable**.

```
R1 pod-template-hash:  
676677fff  
R2 pod-template-hash:  
54f7ff7d6d
```

Deployment  
Revision 2

```
$ kubectl get replicaset  
NAME          DESIRED  CURRENT  READY   AGE  
mydep-54f7ff7d6d  3        3        3      13s  
mydep-676677fff  0        0        0      5h
```

```
$ kubectl get pods  
NAME          READY   STATUS    RESTARTS   AGE  
mydep-54f7ff7d6d-9gv11  1/1     Running   0          10s  
mydep-54f7ff7d6d-cqv1q  1/1     Running   0          8s  
mydep-54f7ff7d6d-gccr6  1/1     Running   0          5s
```



# RollingUpdate Deployment

Updated to new deployment revision completed.

```
R1 pod-template-hash:  
676677fff  
R2 pod-template-hash:  
54f7ff7d6d
```

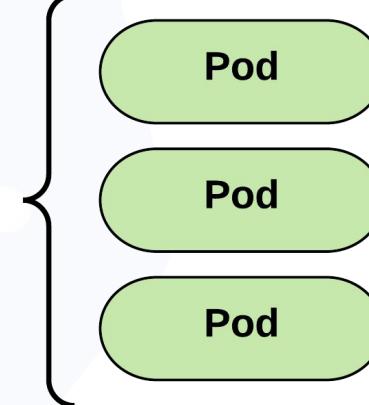
Deployment  
Revision 2

ReplicaSet R1

ReplicaSet R2

```
$ kubectl get replicaset  
NAME          DESIRED   CURRENT   READY   AGE  
mydep-54f7ff7d6d   3         3         3      15s  
mydep-676677fff   0         0         0      5h
```

```
$ kubectl get pods  
NAME          READY   STATUS    RESTARTS   AGE  
mydep-54f7ff7d6d-9gv11   1/1     Running   0          12s  
mydep-54f7ff7d6d-cqv1q   1/1     Running   0          10s  
mydep-54f7ff7d6d-gccr6   1/1     Running   0          7s
```





# DaemonSet

- Ensure that all nodes matching certain criteria will run an instance of the supplied Pod.
- They **bypass** default scheduling mechanisms.
- Are ideal for cluster wide services such as log forwarding, or health monitoring.
- Revisions are managed via a **controller-revision-hash** label.





# DaemonSet

- `revisionHistoryLimit`: The number of previous iterations of the DaemonSet to retain.
- `updateStrategy`: Describes the method of updating the Pods based on the `type`. Valid options are `RollingUpdate` or `onDelete`.
  - `RollingUpdate`: Cycles through updating the Pods according to the value of `maxUnavailable`.
  - `onDelete`: The new instance of the Pod is deployed **ONLY** after the current instance is deleted.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds-example
spec:
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
  template:
    spec:
      nodeSelector:
       .nodeType: edge
<pod template>
```

# DaemonSet



- `spec.template.spec.nodeSelector`:  
The primary selector used to target nodes.
- **Default Host Labels:**
  - `kubernetes.io/hostname`
  - `beta.kubernetes.io/os`
  - `beta.kubernetes.io/arch`
- **Cloud Host Labels:**
  - `failure-domain.beta.kubernetes.io/zone`
  - `failure-domain.beta.kubernetes.io/region`
  - `beta.kubernetes.io/instance-type`

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds-example
spec:
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
  template:
    spec:
      nodeSelector:
       .nodeType: edge
      <pod template>
```



# DaemonSet

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds-example
spec:
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      nodeSelector:
       .nodeType: edge
      containers:
        - name: nginx
          image: nginx:stable-alpine
          ports:
            - containerPort: 80
```

```
$ kubectl get pods
NAME             READY   STATUS  RESTARTS AGE
ds-example-x8kkz 1/1     Running 0       1m
```

```
$ kubectl describe ds ds-example
Name:           ds-example
Selector:       app=nginx,env=prod
Node-Selector:  nodeType=edge
Labels:         app=nginx
                env=prod
Annotations:   <none>
Desired Number of Nodes Scheduled: 1
Current Number of Nodes Scheduled: 1
Number of Nodes Scheduled with Up-to-date Pods: 1
Number of Nodes Scheduled with Available Pods: 1
Number of Nodes Misscheduled: 0
Pods Status:  1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=nginx
           env=prod
  Containers:
    nginx:
      Image:      nginx:stable-alpine
      Port:       80/TCP
      Environment: <none>
      Mounts:    <none>
      Volumes:   <none>
  Events:
    Type      Reason          Age   From           Message
    ----      -----         ----  ----
    Normal   SuccessfulCreate 48s   daemonset-controller  Created pod: ds-example-x8kkz
```

# DaemonSet



```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds-example
spec:
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
  template:
    metadata:
      labels:
        app: nginx
  spec:
    nodeSelector:
     .nodeType: edge
    containers:
      - name: nginx
        image: nginx:stable-alpine
        ports:
          - containerPort: 80
```

```
$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
ds-example-x8kkz   1/1     Running   0          1m
```

```
$ kubectl describe ds ds-example
Name:           ds-example
Selector:       app=nginx,env=prod
Node-Selector:  nodeType=edge
Labels:         app=nginx
                env=prod
Annotations:   <none>
Desired Number of Nodes Scheduled: 1
Current Number of Nodes Scheduled: 1
Number of Nodes Scheduled with Up-to-date Pods: 1
Number of Nodes Scheduled with Available Pods: 1
Number of Nodes Misscheduled: 0
Pods Status:  1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=nginx
           env=prod
  Containers:
    nginx:
      Image:      nginx:stable-alpine
      Port:       80/TCP
      Environment: <none>
      Mounts:     <none>
      Volumes:    <none>
  Events:
    Type      Reason            Age   From               Message
    ----      -----           ----  ----              -----
    Normal   SuccessfulCreate  48s   daemonset-controller  Created pod: ds-example-x8kkz
```



# StatefulSet

- Tailored to managing Pods that must persist or maintain state.
- Pod identity including **hostname**, **network**, and **storage** **WILL** be persisted.
- Assigned a unique ordinal name following the convention of '*<statefulset name>-<ordinal index>*'.





# StatefulSet

- Naming convention is also used in Pod's network Identity and Volumes.
- Pod lifecycle will be ordered and follow consistent patterns.
- Revisions are managed via a **controller-revision-hash** label



# StatefulSet



```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sts-example
spec:
  replicas: 2
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: stateful
  serviceName: app
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      partition: 0
  template:
    metadata:
      labels:
        app: stateful
<continued>
```

<continued>

```
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      ports:
        - containerPort: 80
      volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
    - metadata:
        name: www
  spec:
    accessModes: [ "ReadWriteOnce" ]
    storageClassName: standard
    resources:
      requests:
        storage: 1Gi
```



# StatefulSet

- **revisionHistoryLimit**: The number of previous iterations of the StatefulSet to retain.
- **serviceName**: The name of the associated headless service; or a service without a **ClusterIP**.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sts-example
spec:
  replicas: 2
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: stateful
  serviceName: app
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      partition: 0
  template:
    <pod template>
```

# Headless Service

<StatefulSet Name>-<ordinal>.<service name>.<namespace>.svc.cluster.local

```
apiVersion: v1
kind: Service
metadata:
  name: app
spec:
  clusterIP: None
  selector:
    app: stateful
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

```
$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
sts-example-0  1/1    Running   0          11m
sts-example-1  1/1    Running   0          11m
```

```
/ # dig app.default.svc.cluster.local +noall +answer
; <>>> DiG 9.11.2-P1 <>>> app.default.svc.cluster.local +noall +answer
;; global options: +cmd
app.default.svc.cluster.local. 2 IN A 10.255.0.5
app.default.svc.cluster.local. 2 IN A 10.255.0.2

/ # dig sts-example-0.app.default.svc.cluster.local +noall +answer
; <>>> DiG 9.11.2-P1 <>>> sts-example-0.app.default.svc.cluster.local +noall +answer
;; global options: +cmd
sts-example-0.app.default.svc.cluster.local. 20 IN A 10.255.0.2

/ # dig sts-example-1.app.default.svc.cluster.local +noall +answer
; <>>> DiG 9.11.2-P1 <>>> sts-example-1.app.default.svc.cluster.local +noall +answer
;; global options: +cmd
sts-example-1.app.default.svc.cluster.local. 30 IN A 10.255.0.5
```

# Headless Service

<StatefulSet Name>-<ordinal>.<service name>.<namespace>.svc.cluster.local

```
apiVersion: v1
kind: Service
metadata:
  name: app
spec:
  clusterIP: None
  selector:
    app: stateful
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

```
$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
sts-example-0  1/1    Running   0          11m
sts-example-1  1/1    Running   0          11m
```

```
/ # dig app.default.svc.cluster.local +noall +answer
; <>>> DiG 9.11.2-P1 <>>> app.default.svc.cluster.local +noall +answer
;; global options: +cmd
app.default.svc.cluster.local. 2 IN A 10.255.0.5
app.default.svc.cluster.local. 2 IN A 10.255.0.2
```

```
/ # dig sts-example-0.app.default.svc.cluster.local +noall +answer
; <>>> DiG 9.11.2-P1 <>>> sts-example-0.app.default.svc.cluster.local +noall +answer
;; global options: +cmd
sts-example-0.app.default.svc.cluster.local. 20 IN A 10.255.0.2
```

```
/ # dig sts-example-1.app.default.svc.cluster.local +noall +answer
; <>>> DiG 9.11.2-P1 <>>> sts-example-1.app.default.svc.cluster.local +noall +answer
;; global options: +cmd
sts-example-1.app.default.svc.cluster.local. 30 IN A 10.255.0.5
```

# Headless Service

<StatefulSet Name>-<ordinal>.<service name>.<namespace>.svc.cluster.local

```
apiVersion: v1
kind: Service
metadata:
  name: app
spec:
  clusterIP: None
  selector:
    app: stateful
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

```
/ # dig app.default.svc.cluster.local +noall +answer
; <>>> DiG 9.11.2-P1 <>>> app.default.svc.cluster.local +noall +answer
;; global options: +cmd
app.default.svc.cluster.local. 2 IN A 10.255.0.5
app.default.svc.cluster.local. 2 IN A 10.255.0.2
```

```
$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
sts-example-0 1/1     Running   0          11m
sts-example-1 1/1     Running   0          11m
```

```
/ # dig sts-example-0.app.default.svc.cluster.local +noall +answer
; <>>> DiG 9.11.2-P1 <>>> sts-example-0.app.default.svc.cluster.local +noall +answer
;; global options: +cmd
sts-example-0.app.default.svc.cluster.local. 20 IN A 10.255.0.2
```

```
/ # dig sts-example-1.app.default.svc.cluster.local +noall +answer
; <>>> DiG 9.11.2-P1 <>>> sts-example-1.app.default.svc.cluster.local +noall +answer
;; global options: +cmd
sts-example-1.app.default.svc.cluster.local. 30 IN A 10.255.0.5
```



# StatefulSet

- **updateStrategy**: Describes the method of updating the Pods based on the **type**. Valid options are **OnDelete** or **RollingUpdate**.
  - **OnDelete**: The new instance of the Pod is deployed **ONLY** after the current instance is deleted.
  - **RollingUpdate**: Pods with an ordinal greater than the **partition** value will be updated in one-by-one in reverse order.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sts-example
spec:
  replicas: 2
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: stateful
  serviceName: app
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      partition: 0
  template:
    <pod template>
```

# StatefulSet



- **volumeClaimTemplates:**  
Template of the persistent volume(s) request to use for each instance of the StatefulSet.

```
spec:  
  containers:  
    - name: nginx  
      image: nginx:stable-alpine  
      ports:  
        - containerPort: 80  
      volumeMounts:  
        - name: www  
          mountPath: /usr/share/nginx/html  
  volumeClaimTemplates:  
    - metadata:  
        name: www  
    spec:  
      accessModes: [ "ReadWriteOnce" ]  
      storageClassName: standard  
      resources:  
        requests:  
          storage: 1Gi
```

# VolumeClaimTemplate

<Volume Name>-<StatefulSet Name>-<ordinal>

```
volumeClaimTemplates:  
- metadata:  
    name: www  
  spec:  
    accessModes: [ "ReadWriteOnce" ]  
    storageClassName: standard  
  resources:  
    requests:  
      storage: 1Gi
```

Persistent Volumes associated with a StatefulSet will **NOT** be automatically garbage collected when it's associated StatefulSet is deleted. They must manually be removed.

\$ kubectl get pvc	NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
	www-sts-example-0	Bound	pvc-d2f11e3b-18d0-11e8-ba4f-080027a3682b	1Gi	RWO	standard	4h
	www-sts-example-1	Bound	pvc-d3c923c0-18d0-11e8-ba4f-080027a3682b	1Gi	RWO	standard	4h



# Job

- Job controller ensures one or more pods are executed and successfully terminate.
- Will continue to try and execute the job until it satisfies the completion and/or parallelism condition.
- Pods are **NOT** cleaned up until the job itself is deleted.\*



# Job



- **backoffLimit**: The number of failures before the job itself is considered **failed**.
- **completions**: The total number of successful completions desired.
- **parallelism**: How many instances of the pod can be run concurrently.
- **spec.template.spec.restartPolicy**: Jobs only support a **restartPolicy** of type **Never** or **OnFailure**.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: job-example
spec:
  backoffLimit: 4
  completions: 4
  parallelism: 2
  template:
    spec:
      restartPolicy: Never
      <pod-template>
```

# Job



```
apiVersion: batch/v1
kind: Job
metadata:
  name: job-example
spec:
  backoffLimit: 4
  completions: 4
  parallelism: 2
  template:
    spec:
      containers:
        - name: hello
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["echo hello from $HOSTNAME!"]
  restartPolicy: Never
```

```
$ kubectl get pods --show-all
NAME           READY   STATUS    RESTARTS   AGE
job-example-dvxd2  0/1    Completed  0          51m
job-example-hknns 0/1    Completed  0          52m
job-example-tphkm 0/1    Completed  0          51m
job-example-v5fvq  0/1    Completed  0          52m
```

```
$ kubectl describe job job-example
Name:           job-example
Namespace:      default
Selector:       controller-uid=19d122f4-1576-11e8-a4e2-080027a3682b
Labels:         controller-uid=19d122f4-1576-11e8-a4e2-080027a3682b
                job-name=job-example
Annotations:    <none>
Parallelism:   2
Completions:   4
Start Time:    Mon, 19 Feb 2018 08:09:21 -0500
Pods Statuses: 0 Running / 4 Succeeded / 0 Failed
Pod Template:
  Labels:        controller-uid=19d122f4-1576-11e8-a4e2-080027a3682b
                 job-name=job-example
  Containers:
    hello:
      Image:        alpine:latest
      Port:         <none>
      Command:
        /bin/sh
        -c
      Args:
        echo hello from $HOSTNAME!
      Environment:  <none>
      Mounts:       <none>
      Volumes:      <none>
  Events:
    Type  Reason     Age   From            Message
    ----  ----      --   --              --
    Normal SuccessfulCreate 52m  job-controller  Created pod: job-example-v5fvq
    Normal SuccessfulCreate 52m  job-controller  Created pod: job-example-hknns
    Normal SuccessfulCreate 51m  job-controller  Created pod: job-example-tphkm
    Normal SuccessfulCreate 51m  job-controller  Created pod: job-example-dvxd2
```

# CronJob



An extension of the Job Controller, it provides a method of executing jobs on a cron-like schedule.

CronJobs within Kubernetes  
use **UTC ONLY**.



# CronJob



- **schedule:** The cron schedule for the job.
- **successfulJobHistoryLimit:** The number of successful jobs to retain.
- **failedJobHistoryLimit:** The number of failed jobs to retain.

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: cronjob-example
spec:
  schedule: "*/1 * * * *"
  successfulJobsHistoryLimit: 3
  failedJobsHistoryLimit: 1
  jobTemplate:
    spec:
      completions: 4
      parallelism: 2
      template:
        <pod template>
```

# CronJob



```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: cronjob-example
spec:
  schedule: "*/1 * * * *"
  successfulJobsHistoryLimit: 3
  failedJobsHistoryLimit: 1
  jobTemplate:
    spec:
      completions: 4
      parallelism: 2
      template:
        spec:
          containers:
            - name: hello
              image: alpine:latest
              command: ["/bin/sh", "-c"]
              args: ["echo hello from $HOSTNAME!"]
      restartPolicy: Never
```

```
$ kubectl get jobs
NAME           DESIRED   SUCCESSFUL   AGE
cronjob-example-1519053240  4          4          2m
cronjob-example-1519053300  4          4          1m
cronjob-example-1519053300  4          4          26s
```

```
$ kubectl describe cronjob cronjob-example
Name:           cronjob-example
Namespace:      default
Labels:         <none>
Annotations:   <none>
Schedule:      */1 * * * *
Concurrency Policy: Allow
Suspend:        False
Starting Deadline Seconds: <unset>
Selector:       <unset>
Parallelism:   2
Completions:   4
Pod Template:
  Labels:  <none>
  Containers:
    hello:
      Image:  alpine:latest
      Port:   <none>
      Command:
        /bin/sh
        -c
      Args:
        echo hello from $HOSTNAME!
      Environment:  <none>
      Mounts:      <none>
      Volumes:     <none>
  Last Schedule Time: Mon, 19 Feb 2018 09:54:00 -0500
  Active Jobs:    cronjob-example-1519052040
Events:
  Type  Reason          Age   From           Message
  ----  ----
  Normal SuccessfulCreate 3m   cronjob-controller  Created job cronjob-example-1519051860
  Normal SawCompletedJob  2m   cronjob-controller  Saw completed job: cronjob-example-1519051860
  Normal SuccessfulCreate 2m   cronjob-controller  Created job cronjob-example-1519051920
  Normal SawCompletedJob  1m   cronjob-controller  Saw completed job: cronjob-example-1519051920
  Normal SuccessfulCreate 1m   cronjob-controller  Created job cronjob-example-1519051980
```

# Storage

- **Volumes**
- **Persistent Volumes**
- **Persistent Volume Claims**
- **StorageClass**



# Storage

Pods by themselves are useful, but many workloads require exchanging data between containers, or persisting some form of data.

For this we have **Volumes**, **PersistentVolumes**, **PersistentVolumeClaims**, and **StorageClasses**.



# Volumes

- Storage that is tied to the **Pod's Lifecycle**.
- A pod can have one or more types of volumes attached to it.
- Can be consumed by any of the containers within the pod.
- Survive Pod restarts; however their durability beyond that is dependent on the Volume Type.



# Volume Types

- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- configMap
- csi
- downwardAPI
- emptyDir
- fc (fibre channel)
- flocker
- gcePersistentDisk
- gitRepo
- glusterfs
- hostPath
- iscsi
- local
- nfs
- persistentVolume  
Claim
- projected
- portworxVolume
- quobyte
- rbd
- scaleIO
- secret
- storageos
- vsphereVolume



Persistent Volume Supported

# Volumes



- **volumes**: A list of volume objects to be attached to the Pod. Every object within the list must have its own unique **name**.
- **volumeMounts**: A container specific list referencing the Pod volumes by **name**, along with their desired **mountPath**.

```
apiVersion: v1
kind: Pod
metadata:
  name: volume-example
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      volumeMounts:
        - name: html
          mountPath: /usr/share/nginx/html
          readOnly: true
    - name: content
      image: alpine:latest
      command: ["/bin/sh", "-c"]
      args:
        - while true; do
            date >> /html/index.html;
            sleep 5;
        done
      volumeMounts:
        - name: html
          mountPath: /html
  volumes:
    - name: html
      emptyDir: {}
```



# Volumes

- **volumes**: A list of volume objects to be attached to the Pod. Every object within the list must have its own unique **name**.
- **volumeMounts**: A container specific list referencing the Pod volumes by **name**, along with their desired **mountPath**.

```
apiVersion: v1
kind: Pod
metadata:
  name: volume-example
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      volumeMounts:
        - name: html
          mountPath: /usr/share/nginx/html
          readOnly: true
    - name: content
      image: alpine:latest
      command: ["/bin/sh", "-c"]
      args:
        - while true; do
            date >> /html/index.html;
            sleep 5;
        done
      volumeMounts:
        - name: html
          mountPath: /html
  volumes:
    - name: html
      emptyDir: {}
```



# Volumes

- **volumes**: A list of volume objects to be attached to the Pod. Every object within the list must have its own unique **name**.
- **volumeMounts**: A container specific list referencing the Pod volumes by **name**, along with their desired **mountPath**.

```
apiVersion: v1
kind: Pod
metadata:
  name: volume-example
spec:
  containers:
    - name: nginx
      image: nginx:stable-alpine
      volumeMounts:
        - name: html
          mountPath: /usr/share/nginx/html
          readOnly: true
    - name: content
      image: alpine:latest
      command: ["/bin/sh", "-c"]
      args:
        - while true; do
            date >> /html/index.html;
            sleep 5;
        done
      volumeMounts:
        - name: html
          mountPath: /html
  volumes:
    - name: html
      emptyDir: {}
```



# Persistent Volumes

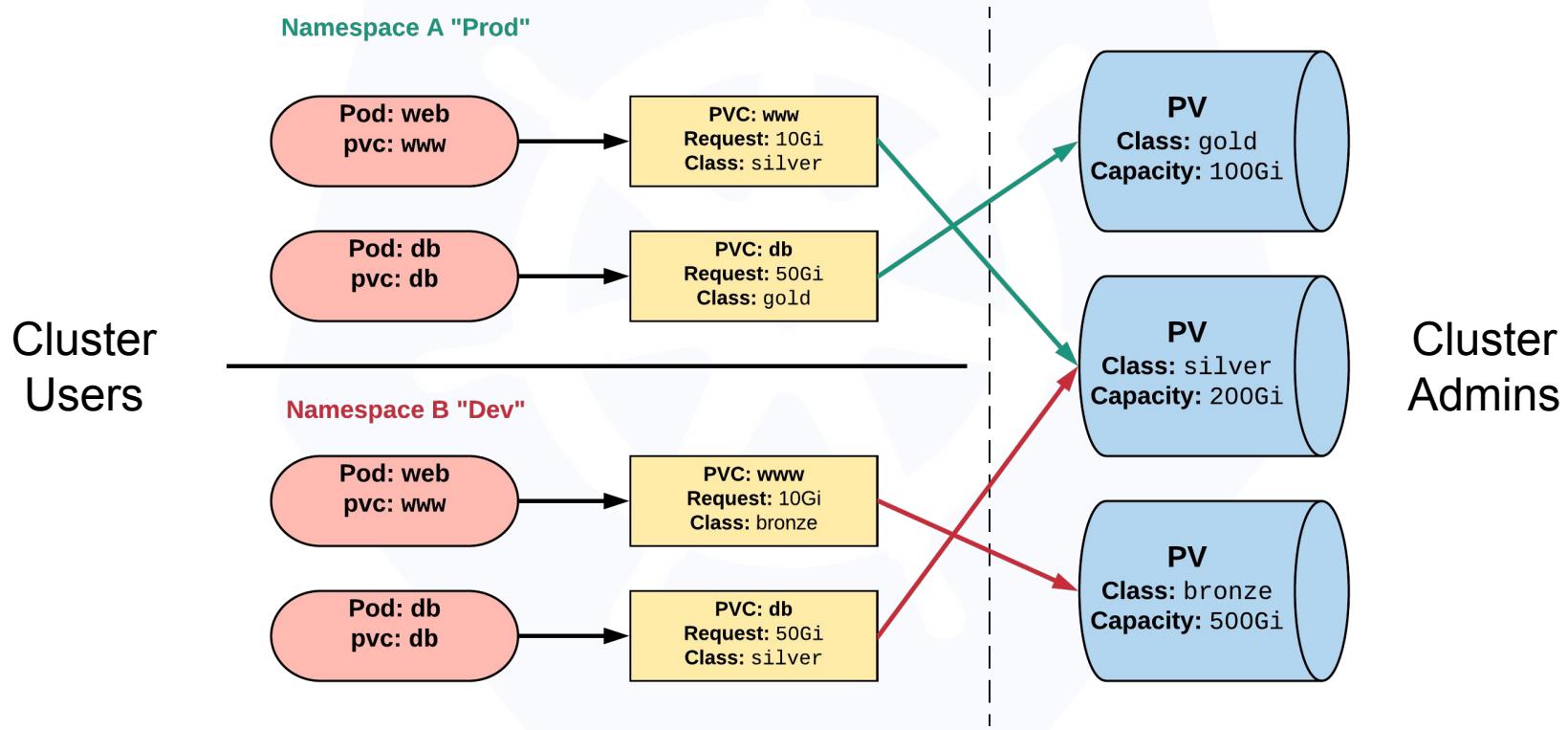
- A **PersistentVolume** (PV) represents a storage resource.
- PVs are a **cluster wide resource** linked to a backing storage provider: NFS, GCEPersistentDisk, RBD etc.
- Generally provisioned by an administrator.
- Their lifecycle is handled independently from a pod
- **CANNOT** be attached to a Pod directly. Relies on a **PersistentVolumeClaim**



# PersistentVolumeClaims

- A **PersistentVolumeClaim** (PVC) is a **namespaced** request for storage.
- Satisfies a set of requirements instead of mapping to a storage resource directly.
- Ensures that an application's '*claim*' for storage is portable across numerous backends or providers.

# Persistent Volumes and Claims



# PersistentVolume



- `capacity.storage`: The total amount of available storage.
- `volumeMode`: The type of volume, this can be either **Filesystem** or **Block**.
- `accessModes`: A list of the supported methods of accessing the volume.  
Options include:
  - **ReadWriteOnce**
  - **ReadOnlyMany**
  - **ReadWriteMany**

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```



# PersistentVolume

- **persistentVolumeReclaimPolicy:** The behaviour for PVC's that have been deleted. Options include:
  - **Retain** - manual clean-up
  - **Delete** - storage asset deleted by provider.
- **storageClassName:** Optional name of the storage class that PVC's can reference. If provided, **ONLY** PVC's referencing the name consume use it.
- **mountOptions:** Optional mount options for the PV.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```



# PersistentVolumeClaim

- **accessModes**: The selected method of accessing the storage. This **MUST** be a subset of what is defined on the target PV or Storage Class.
  - **ReadWriteOnce**
  - **ReadOnlyMany**
  - **ReadWriteMany**
- **resources.requests.storage**: The desired amount of storage for the claim
- **storageClassName**: The name of the desired Storage Class

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-sc-example
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: slow
```

# PVs and PVCs with Selectors



```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv-selector-example
  labels:
    type: hostpath
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-selector-example
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      type: hostpath
```

# PVs and PVCs with Selectors



```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv-selector-example
  labels:
    type: hostpath
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-selector-example
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      type: hostpath
```

# PV Phases



## Available

PV is ready and available to be consumed.

## Bound

The PV has been bound to a claim.

## Released

The binding PVC has been deleted, and the PV is pending reclamation.

## Failed

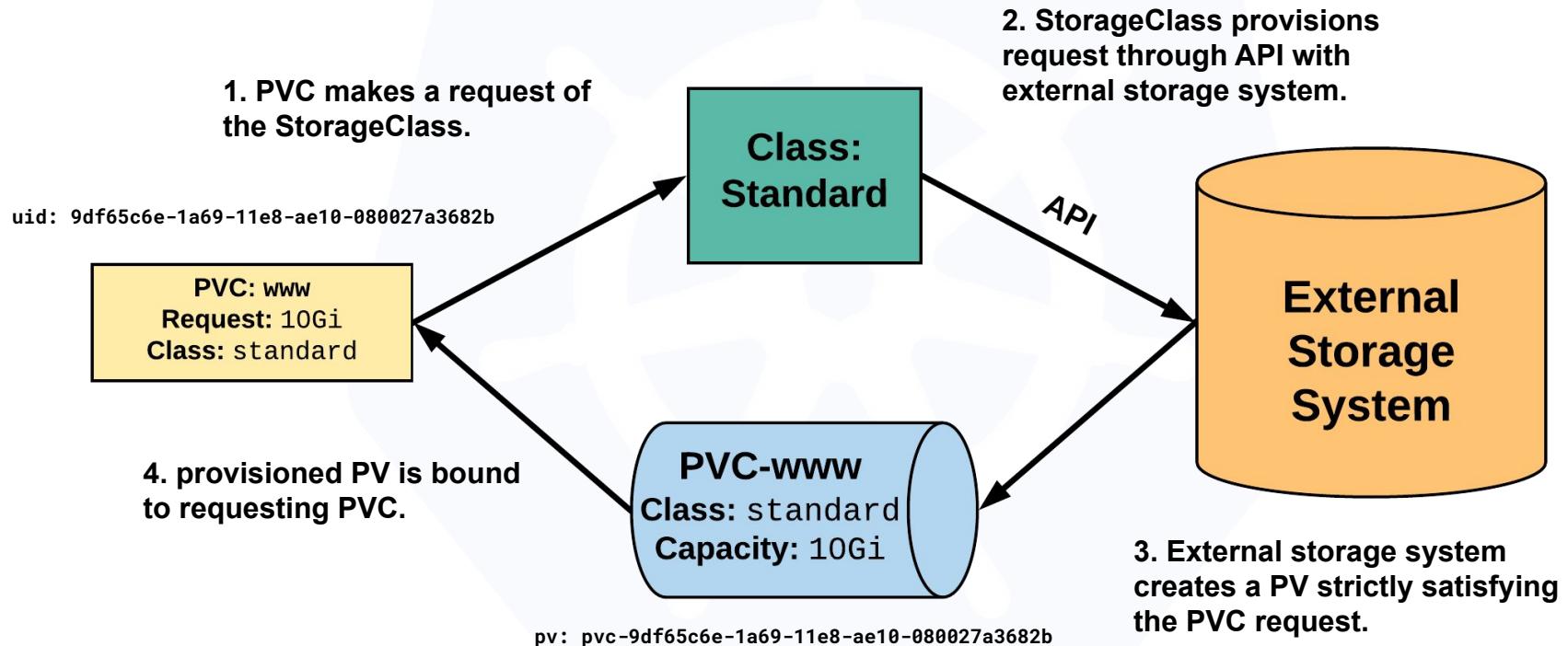
An error has been encountered attempting to reclaim the PV.



# StorageClass

- Storage classes are an abstraction on top of an external storage resource (PV)
- Work hand-in-hand with the external storage system to enable **dynamic provisioning** of storage
- Eliminates the need for the cluster admin to pre-provision a PV

# StorageClass



# StorageClass



- **provisioner**: Defines the '*driver*' to be used for provisioning of the external storage.
- **parameters**: A hash of the various configuration parameters for the provisioner.
- **reclaimPolicy**: The behaviour for the backing storage when the PVC is deleted.
  - **Retain** - manual clean-up
  - **Delete** - storage asset deleted by provider

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: standard
  provisioner: kubernetes.io/gce-pd
  parameters:
    type: pd-standard
    zones: us-central1-a, us-central1-b
  reclaimPolicy: Delete
```



# Available StorageClasses

- AWSElasticBlockStore
- AzureFile
- AzureDisk
- CephFS
- Cinder
- FC
- Flocker
- GCEPersistentDisk
- Glusterfs
- iSCSI
- Quobyte
- NFS
- RBD
- VsphereVolume
- PortworxVolume
- ScaleIO
- StorageOS
- Local



Internal Provisioner

# Configuration

- **ConfigMap**
- **Secret**

Concepts and Resources



# Configuration

Kubernetes has an integrated pattern for decoupling configuration from application or container.

This pattern makes use of two Kubernetes components: **ConfigMaps** and **Secrets**.



# ConfigMap

- Externalized data stored within kubernetes.
- Can be referenced through several different means:
  - environment variable
  - a command line argument (via env var)
  - injected as a file into a volume mount
- Can be created from a manifest, literals, directories, or files directly.

# ConfigMap



`data`: Contains key-value pairs of ConfigMap contents.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: manifest-example
data:
  state: Michigan
  city: Ann Arbor
  content: |
    Look at this,
    its multiline!
```

# ConfigMap Example

All produce a **ConfigMap** with the same content!

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: manifest-example
data:
  city: Ann Arbor
  state: Michigan
```

```
$ kubectl create configmap literal-example \
> --from-literal="city=Ann Arbor" --from-literal=state=Michigan
configmap "literal-example" created
```

```
$ cat info/city
Ann Arbor
$ cat info/state
Michigan
$ kubectl create configmap file-example --from-file=cm/city --from-file=cm/state
configmap "file-example" created
```

```
$ cat info/city
Ann Arbor
$ cat info/state
Michigan
$ kubectl create configmap dir-example --from-file=cm/
configmap "dir-example" created
```

# ConfigMap Example

All produce a **ConfigMap** with the same content!

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: manifest-example
data:
  city: Ann Arbor
  state: Michigan
```

```
$ kubectl create configmap literal-example \
> --from-literal="city=Ann Arbor" --from-literal=state=Michigan
configmap "literal-example" created
```

```
$ cat info/city
Ann Arbor
$ cat info/state
Michigan
$ kubectl create configmap file-example --from-file=cm/city --from-file=cm/state
configmap "file-example" created
```

# ConfigMap Example

All produce a **ConfigMap** with the same content!

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: manifest-example
data:
  city: Ann Arbor
  state: Michigan
```

```
$ kubectl create configmap literal-example \
> --from-literal="city=Ann Arbor" --from-literal=state=Michigan
configmap "literal-example" created
```

```
$ cat info/city
Ann Arbor
$ cat info/state
Michigan
$ kubectl create configmap file-example --from-file=cm/city --from-file=cm/state
configmap "file-example" created
```

```
$ cat info/city
Ann Arbor
$ cat info/state
Michigan
$ kubectl create configmap dir-example --from-file=cm/
configmap "dir-example" created
```

# ConfigMap Example

All produce a **ConfigMap** with the same content!

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: manifest-example
data:
  city: Ann Arbor
  state: Michigan
```

```
$ kubectl create configmap literal-example \
> --from-literal="city=Ann Arbor" --from-literal=state=Michigan
configmap "literal-example" created
```

```
$ cat info/city
Ann Arbor
$ cat info/state
Michigan
$ kubectl create configmap file-example --from-file=cm/city --from-file=cm/state
configmap "file-example" created
```

```
$ cat info/city
Ann Arbor
$ cat info/state
Michigan
$ kubectl create configmap dir-example --from-file=cm/
configmap "dir-example" created
```



# Secret

- Functionally identical to a ConfigMap.
- Stored as **base64 encoded content**.
- Encrypted at rest within etcd (**if configured!**).
- Ideal for username/passwords, certificates or other sensitive information that should not be stored in a container.
- Can be created from a manifest, literals, directories, or from files directly.

# Secret



- **type:** There are three different types of secrets within Kubernetes:
  - **docker-registry** - credentials used to authenticate to a container registry
  - **generic/Opaque** - literal values from different sources
  - **tls** - a certificate based secret
- **data:** Contains key-value pairs of base64 encoded content.

```
apiVersion: v1
kind: Secret
metadata:
  name: manifest-secret
type: Opaque
data:
  username: ZXhhbXBsZQ==
  password: bXlwYXNzd29yZA==
```

# Using ConfigMaps and Secrets

# Secret Example

All produce a **Secret** with the same content!

```
apiVersion: v1
kind: Secret
metadata:
  name: manifest-example
type: Opaque
data:
  username: ZXhhbXBsZQ==
  password: bXlwYXNzd29yZA==
```

```
$ kubectl create secret generic literal-secret \
> --from-literal=username=example \
> --from-literal=password=mypassword
secret "literal-secret" created
```

```
$ cat info/username
example
$ cat info/password
mypassword
$ kubectl create secret generic dir-secret --from-file=secret/
Secret "file-secret" created
```

```
$ cat secret/username
example
$ cat secret/password
mypassword
$ kubectl create secret generic file-secret --from-file=secret/username --from-file=secret/password
Secret "file-secret" created
```

# Secret Example

All produce a **Secret** with the same content!

```
apiVersion: v1
kind: Secret
metadata:
  name: manifest-example
type: Opaque
data:
  username: ZXhhbXBsZQ==
  password: bXlwYXNzd29yZA==
```

```
$ kubectl create secret generic literal-secret \
> --from-literal=username=example \
> --from-literal=password=mypassword
secret literal-secret created
```

```
$ cat info/username
example
$ cat info/password
mypassword
$ kubectl create secret generic dir-secret --from-file=secret/
Secret "file-secret" created
```

```
$ cat secret/username
example
$ cat secret/password
mypassword
$ kubectl create secret generic file-secret --from-file=secret/username --from-file=secret/password
Secret "file-secret" created
```

# Secret Example

All produce a **Secret** with the same content!

```
apiVersion: v1
kind: Secret
metadata:
  name: manifest-example
type: Opaque
data:
  username: ZXhhbXBsZQ==
  password: bXlwYXNzd29yZA==
```

```
$ kubectl create secret generic literal-secret \
> --from-literal=username=example \
> --from-literal=password=mypassword
secret "literal-secret" created
```

```
$ cat info/username
example
$ cat info/password
mypassword
$ kubectl create secret generic dir-secret --from-file=secret/
Secret "file-secret" created
```

```
$ cat secret/username
example
$ cat secret/password
mypassword
$ kubectl create secret generic file-secret --from-file=secret/username --from-file=secret/password
Secret "file-secret" created
```

# Secret Example

All produce a **Secret** with the same content!

```
apiVersion: v1
kind: Secret
metadata:
  name: manifest-example
type: Opaque
data:
  username: ZXhhbXBsZQ==
  password: bXlwYXNzd29yZA==
```

```
$ kubectl create secret generic literal-secret \
> --from-literal=username=example \
> --from-literal=password=mypassword
secret "literal-secret" created
```

```
$ cat info/username
example
$ cat info/password
mypassword
$ kubectl create secret generic dir-secret --from-file=secret/
Secret "file-secret" created
```

```
$ cat secret/username
example
$ cat secret/password
mypassword
$ kubectl create secret generic file-secret --from-file=secret/username --from-file=secret/password
Secret "file-secret" created
```

# Injecting as Environment Variable



```
apiVersion: batch/v1
kind: Job
metadata:
  name: cm-env-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["printenv CITY"]
      env:
        - name: CITY
          valueFrom:
            configMapKeyRef:
              name: manifest-example
              key: city
  restartPolicy: Never
```

```
apiVersion: batch/v1
kind: Job
metadata:
  name: secret-env-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["printenv USERNAME"]
      env:
        - name: USERNAME
          valueFrom:
            secretKeyRef:
              name: manifest-example
              key: username
  restartPolicy: Never
```

# Injecting as Environment Variable



```
apiVersion: batch/v1
kind: Job
metadata:
  name: cm-env-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["printenv CITY"]
          env:
            - name: CITY
              valueFrom:
                configMapKeyRef:
                  name: manifest-example
                  key: city
  restartPolicy: Never
```

```
apiVersion: batch/v1
kind: Job
metadata:
  name: secret-env-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["printenv USERNAME"]
          env:
            - name: USERNAME
              valueFrom:
                secretKeyRef:
                  name: manifest-example
                  key: username
  restartPolicy: Never
```

# Injecting in a Command



```
apiVersion: batch/v1
kind: Job
metadata:
  name: cm-cmd-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["echo Hello ${CITY}!"]
      env:
        - name: CITY
          valueFrom:
            configMapKeyRef:
              name: manifest-example
              key: city
  restartPolicy: Never
```

```
apiVersion: batch/v1
kind: Job
metadata:
  name: secret-cmd-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["echo Hello ${USERNAME}!"]
      env:
        - name: USERNAME
          valueFrom:
            secretKeyRef:
              name: manifest-example
              key: username
  restartPolicy: Never
```

# Injecting in a Command



```
apiVersion: batch/v1
kind: Job
metadata:
  name: cm-cmd-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["echo Hello ${CITY}!"]
          env:
            - name: CITY
              valueFrom:
                configMapKeyRef:
                  name: manifest-example
                  key: city
  restartPolicy: Never
```

```
apiVersion: batch/v1
kind: Job
metadata:
  name: secret-cmd-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["echo Hello ${USERNAME}!"]
          env:
            - name: USERNAME
              valueFrom:
                secretKeyRef:
                  name: manifest-example
                  key: username
  restartPolicy: Never
```



# Injecting as a Volume

```
apiVersion: batch/v1
kind: Job
metadata:
  name: cm-vol-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["cat /myconfig/city"]
        volumeMounts:
          - name: config-volume
            mountPath: /myconfig
  restartPolicy: Never
  volumes:
    - name: config-volume
      configMap:
        name: manifest-example
```

```
apiVersion: batch/v1
kind: Job
metadata:
  name: secret-vol-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["cat /mysecret/username"]
        volumeMounts:
          - name: secret-volume
            mountPath: /mysecret
  restartPolicy: Never
  volumes:
    - name: secret-volume
      secret:
        secretName: manifest-example
```



# Injecting as a Volume

```
apiVersion: batch/v1
kind: Job
metadata:
  name: cm-vol-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["cat /myconfig/city"]
        volumeMounts:
          - name: config-volume
            mountPath: /myconfig
  restartPolicy: Never
volumes:
  - name: config-volume
    configMap:
      name: manifest-example
```

```
apiVersion: batch/v1
kind: Job
metadata:
  name: secret-vol-example
spec:
  template:
    spec:
      containers:
        - name: mypod
          image: alpine:latest
          command: ["/bin/sh", "-c"]
          args: ["cat /mysecret/username"]
        volumeMounts:
          - name: secret-volume
            mountPath: /mysecret
  restartPolicy: Never
volumes:
  - name: secret-volume
    secret:
      secretName: manifest-example
```

# Putting it all Together

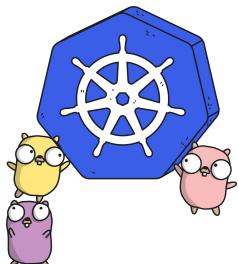


# Where to go From Here

# kubernetes.io



## Documentation Examples API Reference



The screenshot shows a web browser displaying the Kubernetes website at https://kubernetes.io. The page features a dark header with the Kubernetes logo and navigation links for Documentation, Blog, Partners, Community, Case Studies, English, and v1.12. Below the header, a main title reads "Production-Grade Container Orchestration" with the subtitle "Automated container deployment, scaling, and management". A "Learn Kubernetes Basics" button is visible. The main content area includes a brief introduction to Kubernetes and its history, followed by a diagram illustrating how multiple containers are grouped into a logical unit managed by the Kubernetes controller. At the bottom, there is a section titled "Planet Scale" with a sub-diagram showing a cluster of nodes.

**Kubernetes** is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

**Planet Scale**

Designed on the same principles that allows Google to run billions of containers a week, Kubernetes can scale without increasing your ops team.



## Kubernetes

[slack.k8s.io](https://slack.k8s.io)



70,000+ users

300+ public channels

## CNCF

[slack.cncf.io](https://slack.cncf.io)



8,200+ users

70+ public channels

# Other Communities



- **Official Forum**

<https://discuss.kubernetes.io>

- **Subreddit**

<https://reddit.com/r/kubernetes>

- **StackOverflow**

<https://stackoverflow.com/questions/tagged/kubernetes>



# Meetups



## Kubernetes

[meetup.com/topics/kubernetes/](https://meetup.com/topics/kubernetes/)



850+ groups

380,000+ members

## CNCF

[meetups.cncf.io](https://meetups.cncf.io)



200+ groups

120,000+ members

45+ countries

# 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.

# SIG List

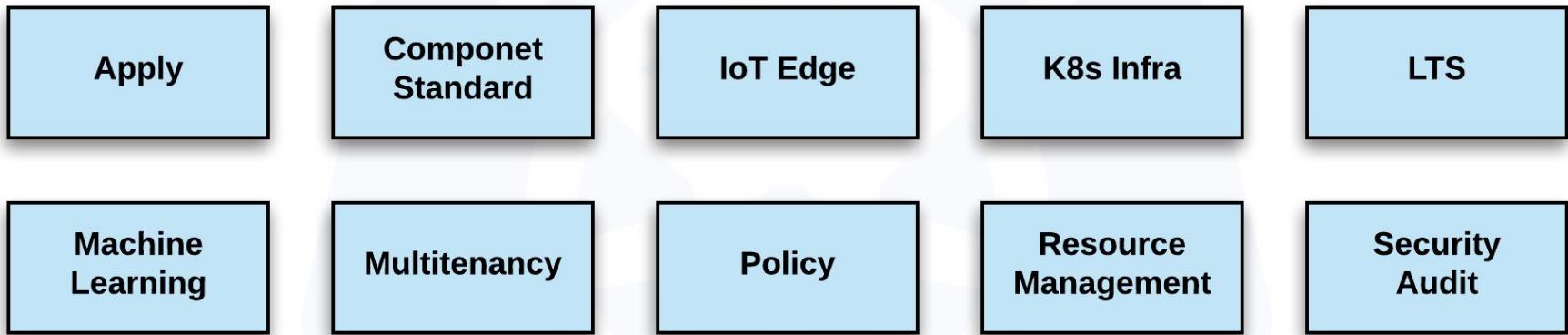
API Machinery	Apps	Architecture	Auth	Autoscaling	CLI
Cloud Provider	Cluster Lifecycle	Contributor Experience	Docs	Instrumentation	Multicloud
Network	Node	Product Management	Release	Scalability	Scheduling
Service Catalog	Storage	Testing	UI	Usability	Windows



# 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.

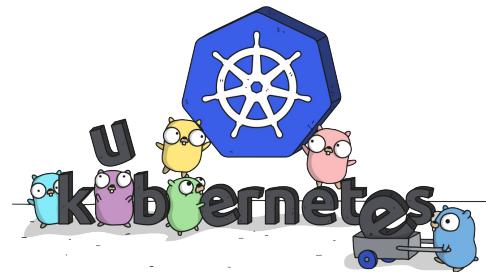
# WG List



# 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/>



# Thank you!