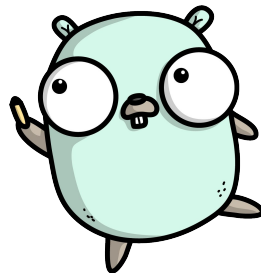


Before We Begin



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>



Getting Started with **Kubernetes**



\$ whoami - Bob



Bob Killen

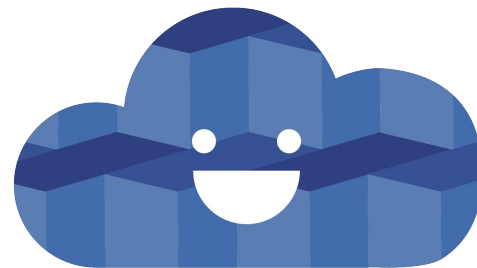
rkillen@umich.edu

Senior Research Cloud Administrator

CNCF Ambassador

Github: [@mrbobbytables](https://github.com/mrbobbytables)

Twitter: [@mrbobbytables](https://twitter.com/mrbobbytables)



 **CLOUD NATIVE**
COMPUTING FOUNDATION

AMBASSADOR

\$ whoami - Jeff



Jeffrey Sica

jsica@umich.edu

Senior Research Database Administrator

Github: [@jeefy](#)

Twitter: [@jeefy](#)



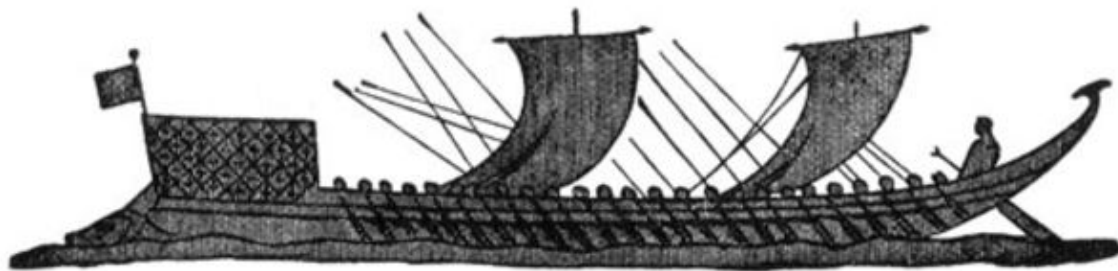
The background of the slide is a solid blue color. In the center, there is a large, faint, light-blue outline of the Kubernetes logo, which is a ship's steering wheel. Overlaid on this background is the text "What is Kubernetes?" in a bold, white, sans-serif font. The text is centered and occupies the middle portion of the slide.

What is Kubernetes?

What Does “Kubernetes” Mean?



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



[Image Source](#)





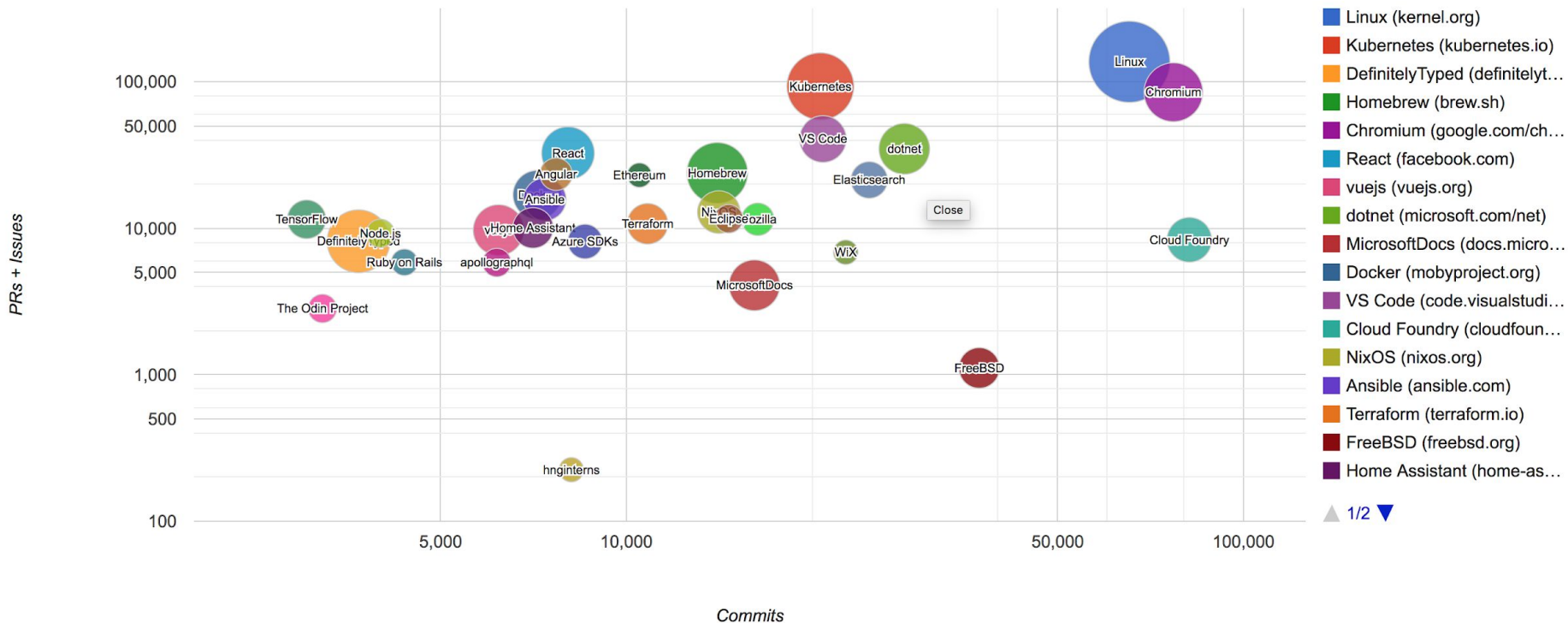
What is Kubernetes?

- Originally sprung out of decades of container experience from inside Google (Borg, Omega, LMCTFY, etc.)
- Independent OSS project within the CNCF
- Production ready since July 2015.
- Automates deployment, scaling, and management of application containers

Kubernetes Stats



Top-30 projects in 2017





What Does Kubernetes do?

- The “**linux kernel of distributed systems**”
- Abstracts away the underlying hardware
- You **declare** a state, and Kubernetes’ main purpose is to make that happen
- Handles placement and scheduling of containers on nodes
- Provides basic monitoring, logging, and health checking
- Enables containers to discover each other (important!)

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.

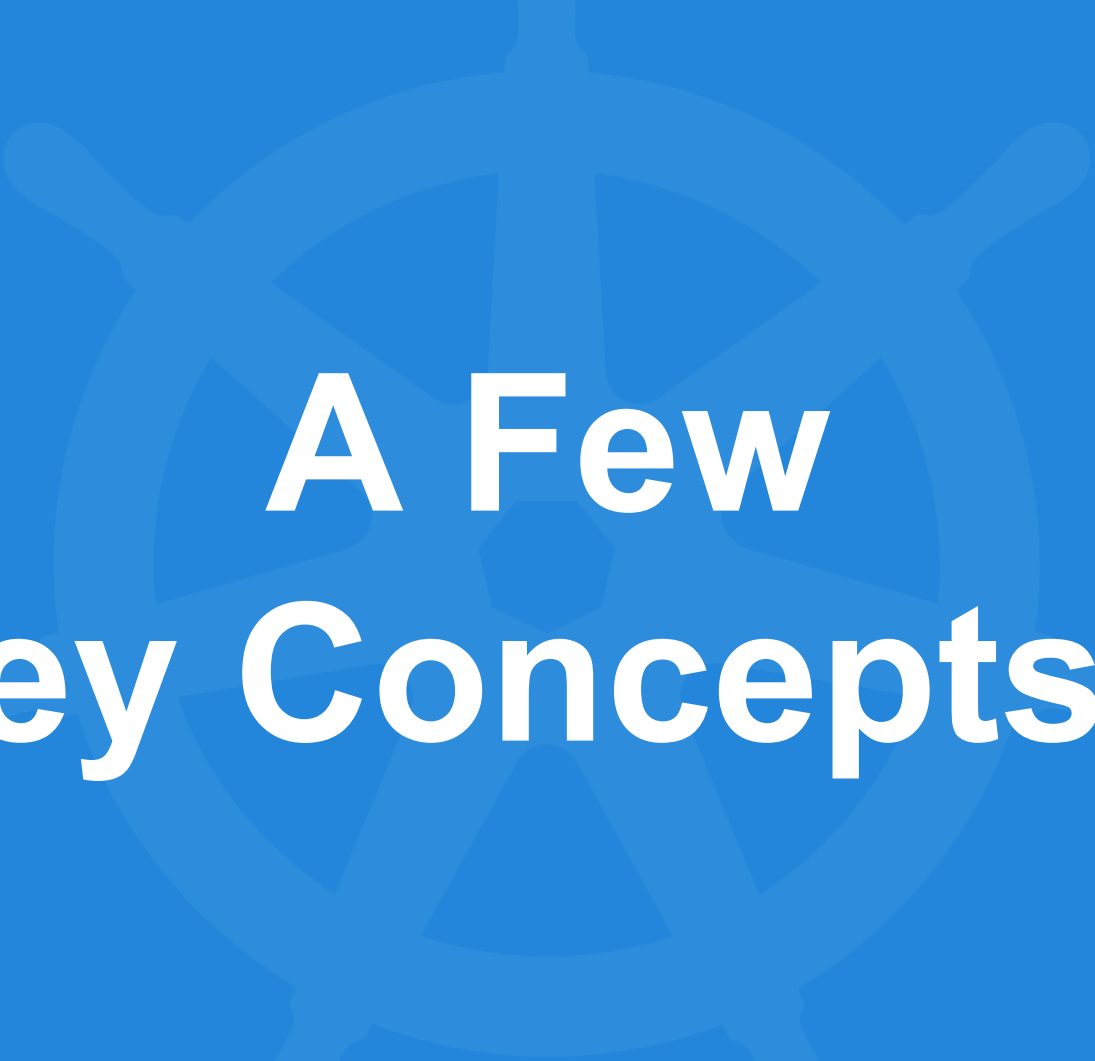
- **Me:** “I want 3 healthy instances of redis to always be running.”
- **Kubernetes:** “Okay, I’ll ensure there are always 3 instances up and running.”
- **Kubernetes:** “Oh look, one has died. I’m going to attempt to spin up a new one.”

Most Importantly...



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



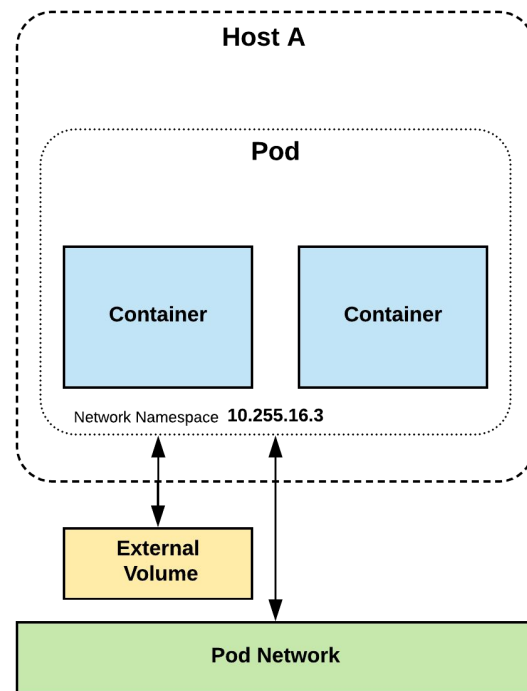


A Few Key Concepts...

Pods



- A pod is the **atomic unit** 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**.

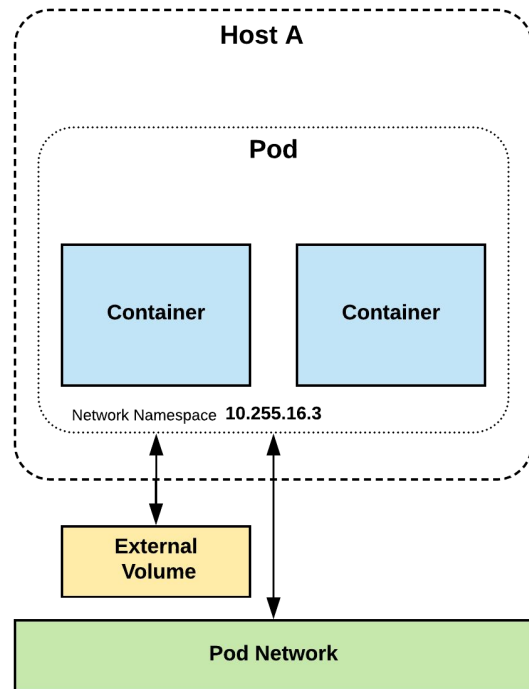


Pods



**They are
also
Ephemeral!**

(higher level objects manage replicas, fault-tolerance etc)



Services

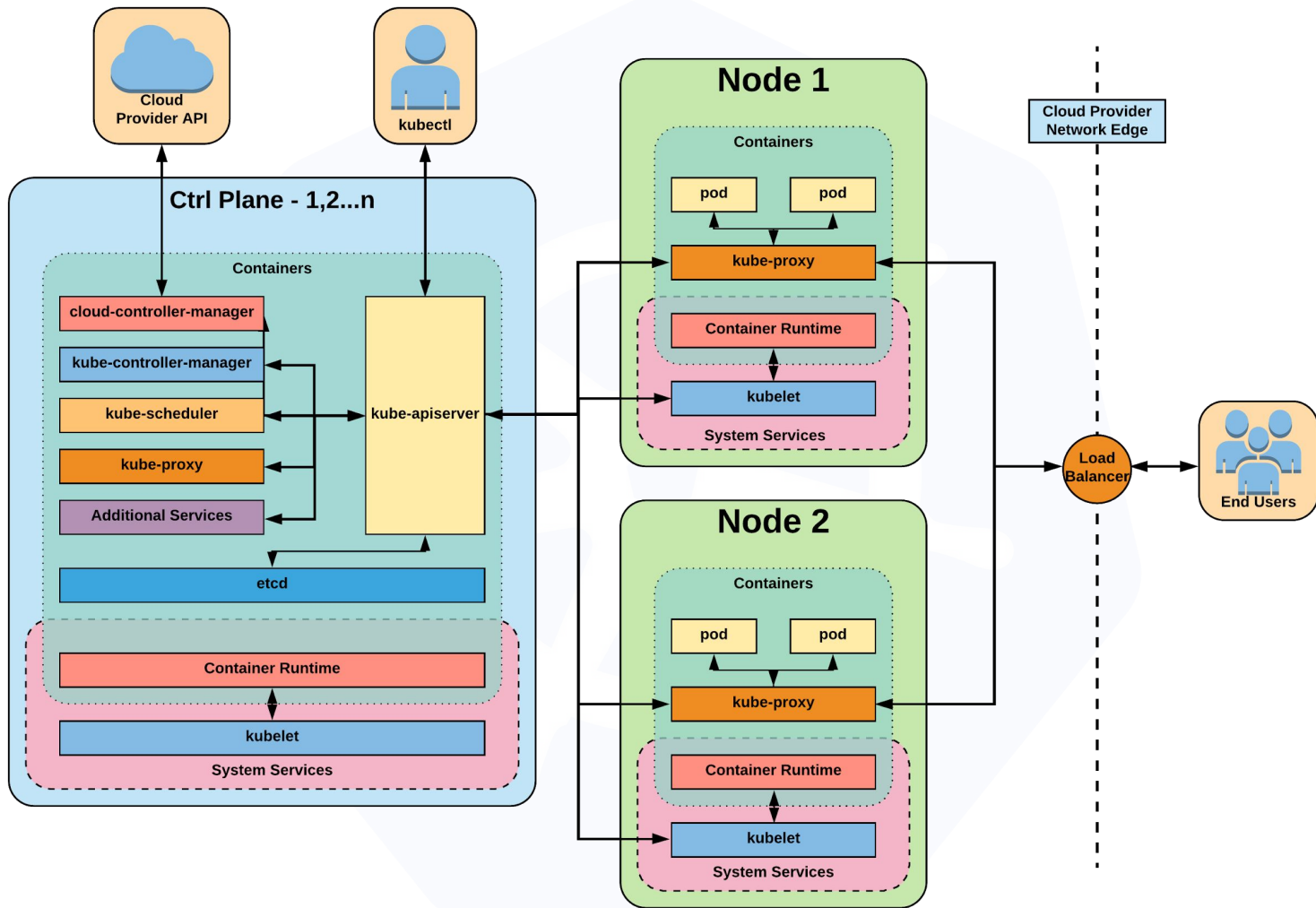


- Services within Kubernetes are the **unified method of accessing** the exposed workloads of Pods.
- They are a **durable resource** (unlike Pods)
- Given a static cluster-unique IP, and in conjunction with kube-dns a static DNS name following the format of:

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



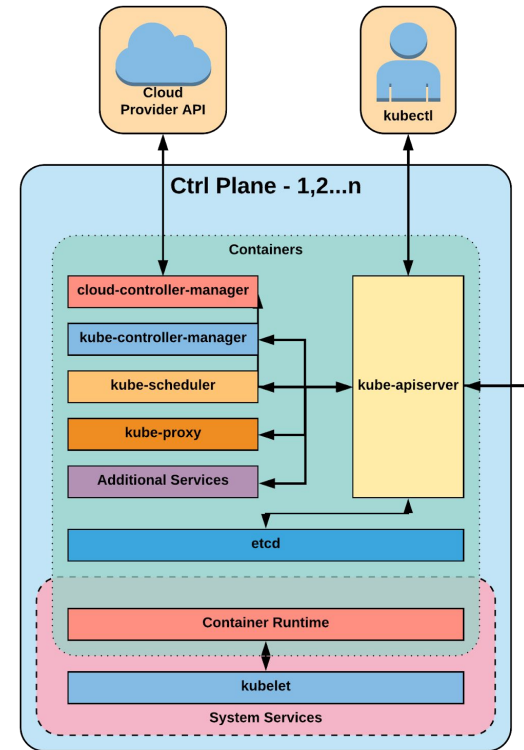
Architecture Overview



Control Plane Components



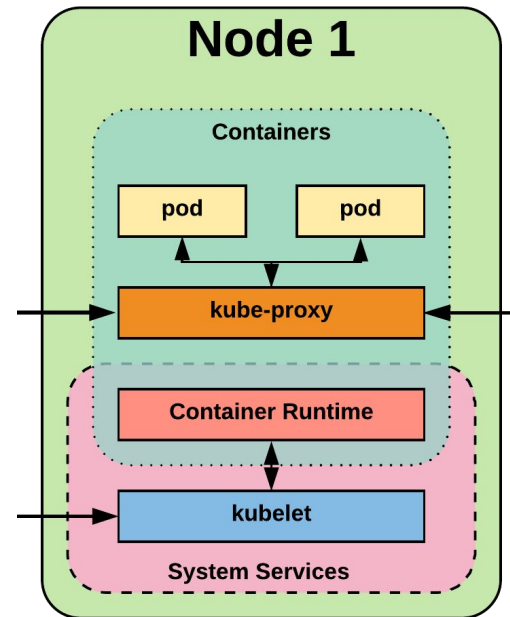
- kube-apiserver
- etcd
- kube-controller-manager
- kube-scheduler



Node Components



- kubelet
- kube-proxy
- Container Runtime Engine



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.

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.

The API and Object Model



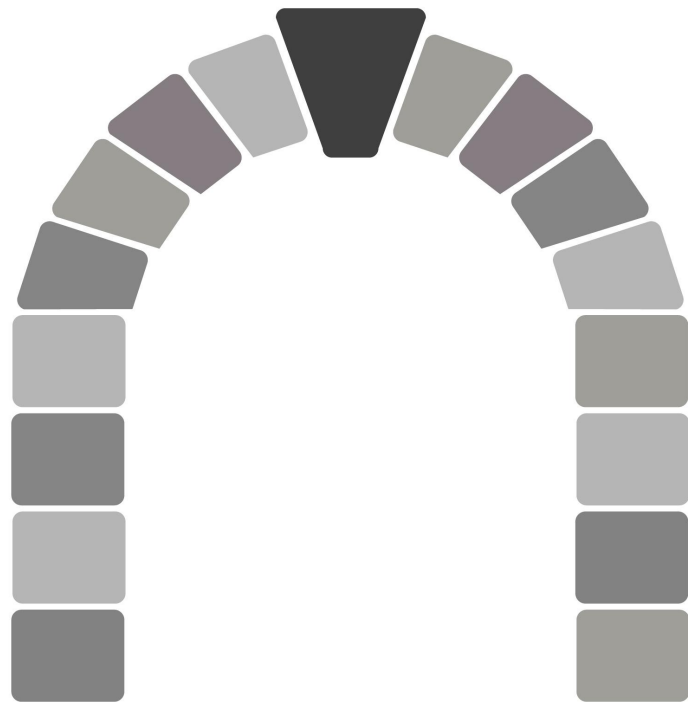
Concepts and Resources

API Overview



The **REST API** is the true **keystone** of Kubernetes.

Everything within the Kubernetes platform is treated as an **API Object** and has a corresponding entry in the API itself.



[Image Source](#)

Object Model



- Objects within Kubernetes are a “*record of intent*”
 - Persistent entity that represent the desired state of the object within the cluster.
- At a minimum all objects **MUST** have an `apiVersion`, `kind`, and poses 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
```

Using the API

(aka, using the CLI)



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 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.

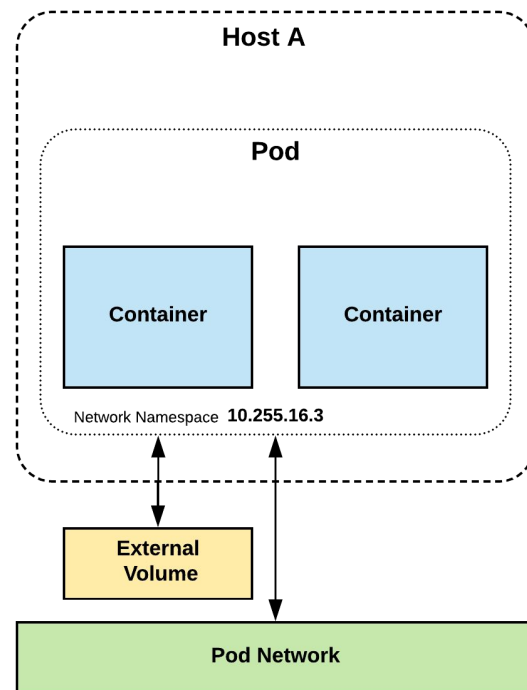
```
$ kubectl get ns --show-labels
```

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

Pods



- A pod is the **atomic unit** of Kubernetes.
- It is the 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**.



Pod Examples



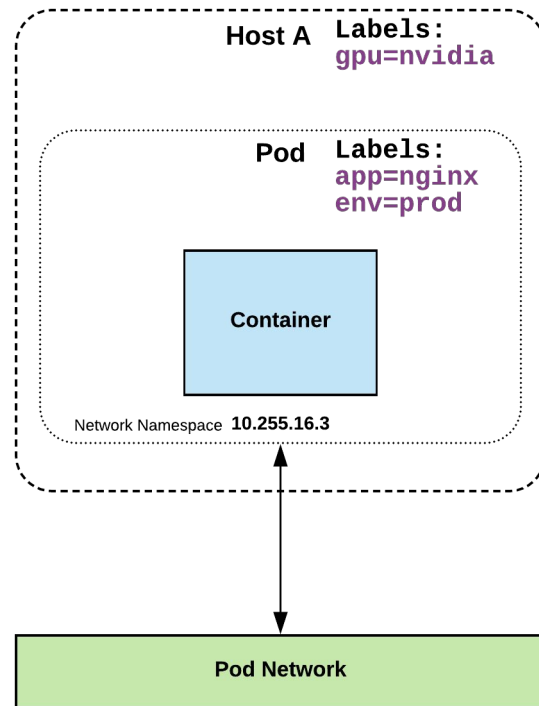
```
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
    - containerPort: 80
```

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-container-example
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    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: {}
```

Labels



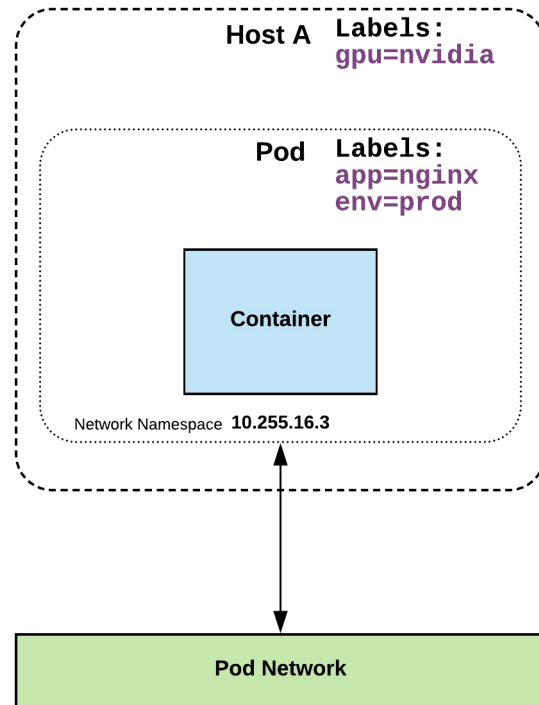
- Labels are key-value pairs that are used to identify, describe and group together related sets of objects or resources.



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

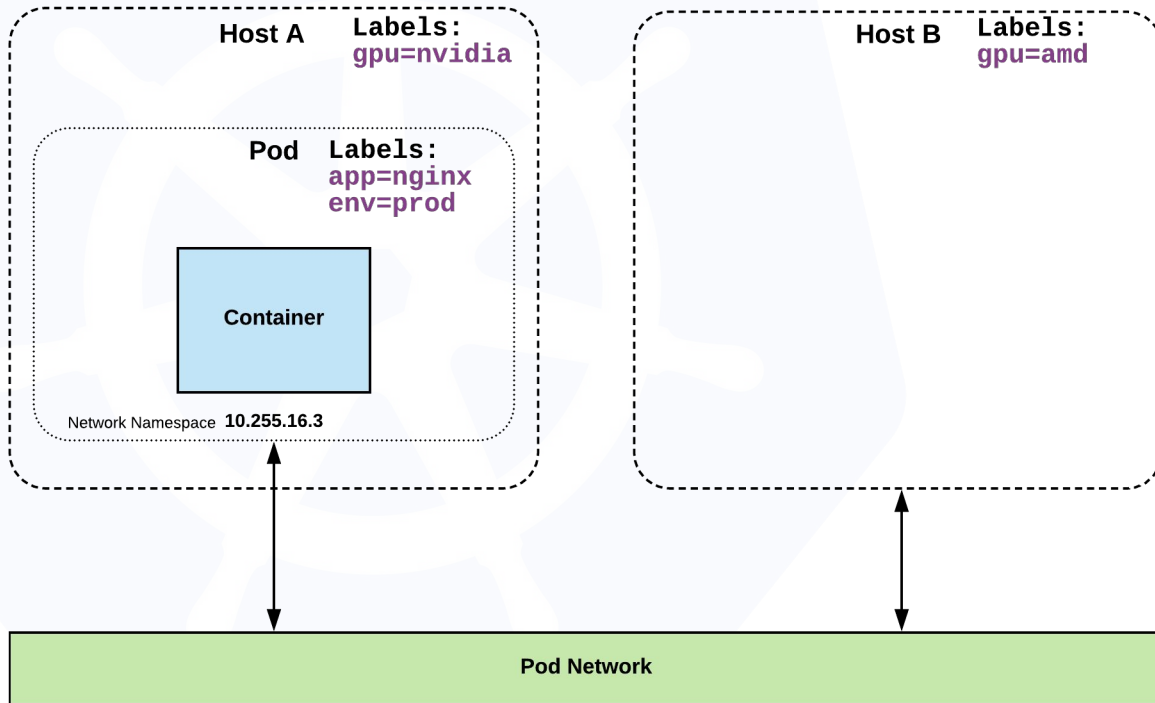


Selectors use labels to filter or select objects, and are used throughout Kubernetes.

```
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"]
```



Service Types

There are 4 major service types:

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

ClusterIP Service



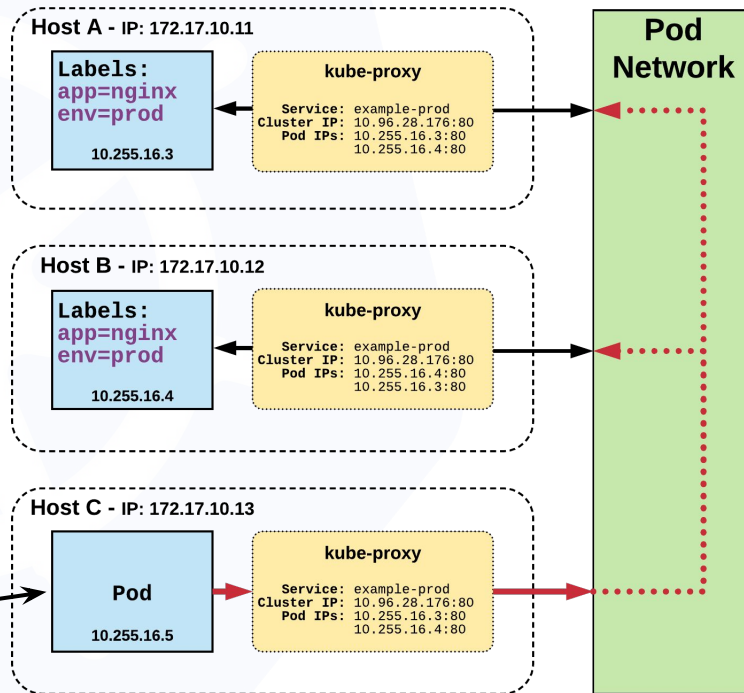
- **ClusterIP** services exposes a service on a strictly cluster-internal virtual IP.

```
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
```



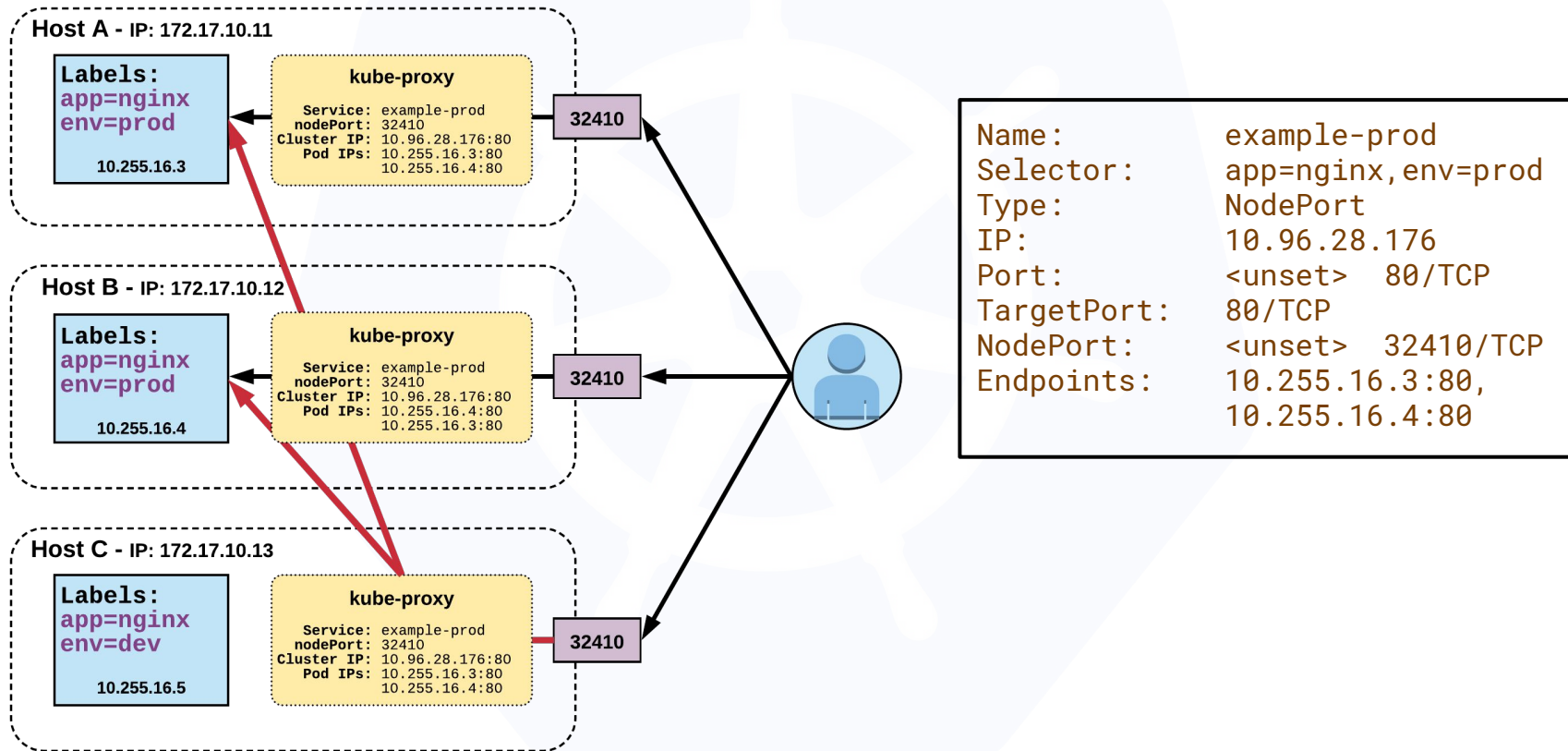


NodePort Service

- **NodePort** services extend the **ClusterIP** service and additionally exposes a port on every node.

```
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



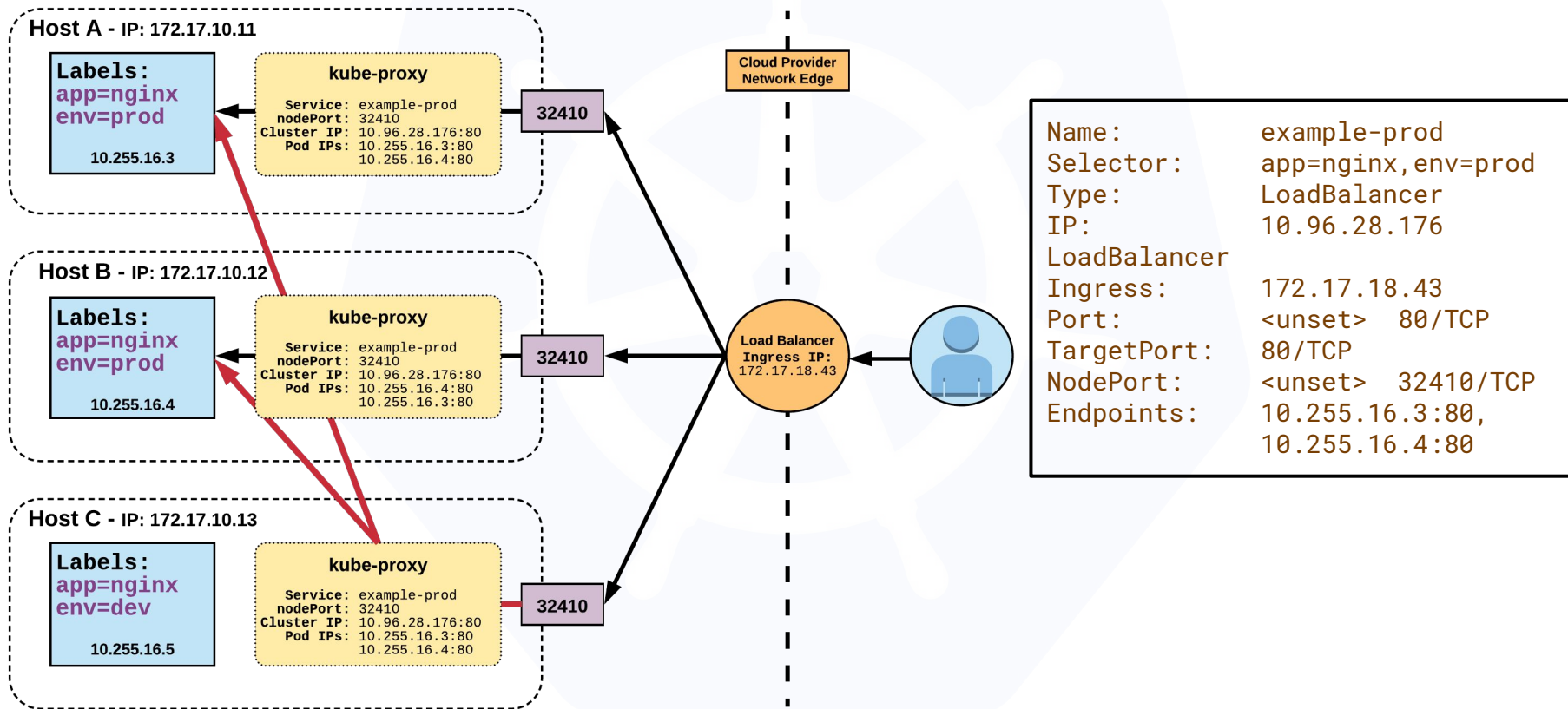


LoadBalancer Service

- **LoadBalancer** services extend **NodePort** and 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

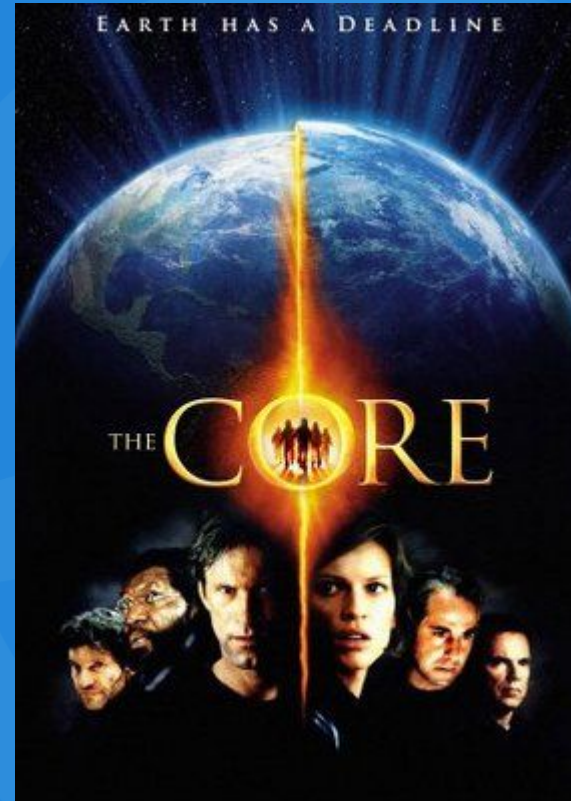
- **ExternalName** is used to reference endpoints OUTSIDE the cluster.
- It creates an internal **CNAME** DNS entry that aliases another.

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

Exploring the Core



Exploring the Core



Workloads

- ReplicaSet
- Deployment

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-mk1l2    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-mk1l2
  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

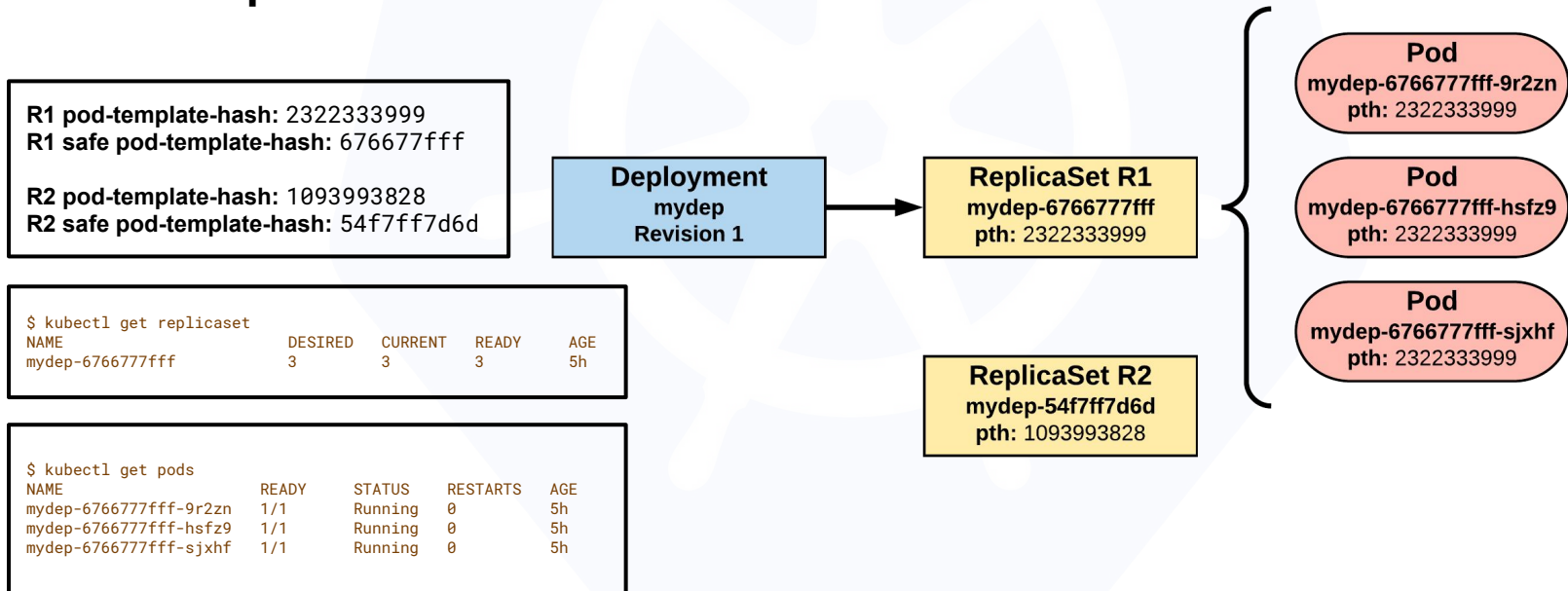


- **revisionHistoryLimit**: The number of previous iterations of the Deployment to retain.
- **strategy**: Describes the method of updating the Pods based on the **type**. Valid options are **RollingUpdate** or **Recreate**.
 - **RollingUpdate**: Cycles through updating the Pods according to the parameters: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

```
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.



RollingUpdate Deployment

New **ReplicaSet** is initially scaled up based on **maxSurge**.

R1 pod-template-hash: 2322333999
R1 safe pod-template-hash: 676677ffff

R2 pod-template-hash: 1093993828
R2 safe pod-template-hash: 54f7ff7d6d

Deployment
mydep
Revision 2

ReplicaSet R1
mydep-6766777fff
pth: 2322333999

Pod
mydep-6766777fff-9r2zn
pth: 2322333999

Pod
mydep-6766777fff-hsfz9
pth: 2322333999

Pod
mydep-6766777fff-sjxhf
pth: 2322333999

ReplicaSet R2
mydep-54f7ff7d6d
pth: 1093993828

Pod
mydep-54f7ff7d6d-9gvll
pth: 1093993828

```
$ 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-9gvll	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: 2322333999
R1 safe pod-template-hash: 676677fff

R2 pod-template-hash: 1093993828
R2 safe pod-template-hash: 54f7ff7d6d

Deployment
mydep
Revision 2

ReplicaSet R1
mydep-6766777fff
pth: 2322333999

Pod
mydep-6766777fff-9r2zn
pth: 2322333999

Pod
mydep-6766777fff-hsfz9
pth: 2322333999

~~**Pod**
mydep-6766777fff-sjxhf
pth: 2322333999~~

```
$ kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
mydep-54f7ff7d6d	2	2	2	8s
mydep-6766777fff	2	2	2	5h

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mydep-54f7ff7d6d-9gvll	1/1	Running	0	5s
mydep-54f7ff7d6d-cqvlq	1/1	Running	0	2s
mydep-6766777fff-9r2zn	1/1	Running	0	5h
mydep-6766777fff-hsfz9	1/1	Running	0	5h

ReplicaSet R2
mydep-54f7ff7d6d
pth: 1093993828

Pod
mydep-54f7ff7d6d-9gvll
pth: 1093993828

Pod
mydep-54f7ff7d6d-cqvlq
pth: 1093993828

RollingUpdate Deployment

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

R1 pod-template-hash: 2322333999
R1 safe pod-template-hash: 676677ffff

R2 pod-template-hash: 1093993828
R2 safe pod-template-hash: 54f7ff7d6d

Deployment
mydep
Revision 2

ReplicaSet R1
mydep-676677ffff
pth: 2322333999

ReplicaSet R2
mydep-54f7ff7d6d
pth: 1093993828

```
$ kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
mydep-54f7ff7d6d	3	3	3	10s
mydep-676677ffff	0	1	1	5h

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mydep-54f7ff7d6d-9gvll	1/1	Running	0	7s
mydep-54f7ff7d6d-cqvlq	1/1	Running	0	5s
mydep-54f7ff7d6d-gccr6	1/1	Running	0	2s
mydep-676677ffff-9r2zn	1/1	Running	0	5h

Pod

mydep-676677ffff-9r2zn
pth: 2322333999

~~**Pod**~~

~~**mydep-676677ffff-hsfz9**
pth: 2322333999~~

~~**Pod**~~

~~**mydep-676677ffff-sjxhf**
pth: 2322333999~~

Pod

mydep-54f7ff7d6d-9gvll
pth: 1093993828

Pod

mydep-54f7ff7d6d-cqvlq
pth: 1093993828

Pod

mydep-54f7ff7d6d-gccr6
pth: 1093993828

RollingUpdate Deployment

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

R1 pod-template-hash: 2322333999
R1 safe pod-template-hash: 676677ffff

R2 pod-template-hash: 1093993828
R2 safe pod-template-hash: 54f7ff7d6d

Deployment
mydep
Revision 2

ReplicaSet R1
mydep-67667777ff
pth: 2322333999

ReplicaSet R2
mydep-54f7ff7d6d
pth: 1093993828

```
$ kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
mydep-54f7ff7d6d	3	3	3	13s
mydep-67667777ff	0	0	0	5h

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mydep-54f7ff7d6d-9gvll	1/1	Running	0	10s
mydep-54f7ff7d6d-cqvlq	1/1	Running	0	8s
mydep-54f7ff7d6d-gccr6	1/1	Running	0	5s

~~Pod~~

~~mydep-67667777ff-9r2zn
pth: 2322333999~~

~~Pod~~

~~mydep-67667777ff-hsfz9
pth: 2322333999~~

~~Pod~~

~~mydep-67667777ff-sjxhf
pth: 2322333999~~

Pod

mydep-54f7ff7d6d-9gvll
pth: 1093993828

Pod

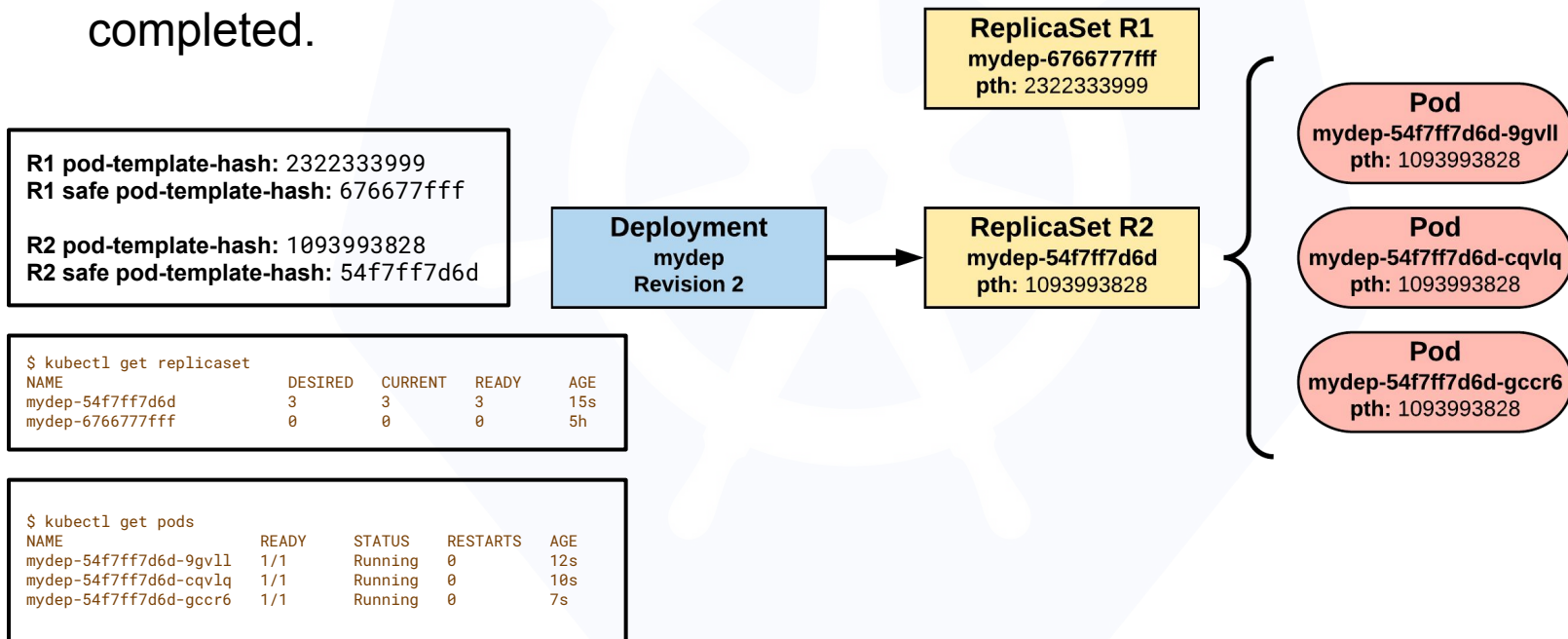
mydep-54f7ff7d6d-cqvlq
pth: 1093993828

Pod

mydep-54f7ff7d6d-gccr6
pth: 1093993828

RollingUpdate Deployment

Updated to new deployment revision completed.



Using Workloads





**Where to go
From Here**

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/channel/UCZ2bu0qutTOM0tHYa_jklwg
- Official CNCF Youtube Channel
<https://www.youtube.com/channel/UCvqbFHwN-nwalWPjPUKpvTA>
- Track to becoming a CKA/CKAD (Certified Kubernetes Administrator/Application Developer)
<https://www.cncf.io/certification/expert/>
- Awesome Kubernetes
<https://www.gitbook.com/book/ramitsurana/awesome-kubernetes/details>

