Kubernetes

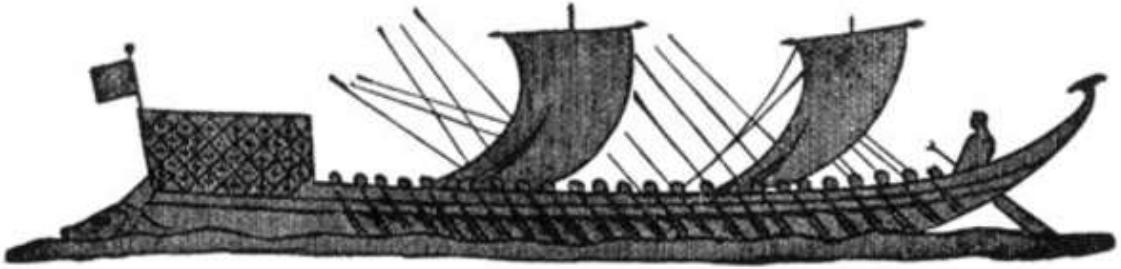
An Introduction

Why learn Kubernetes?

It's like landing on Pluto when people are still trying to figure out Mars (other tools) properly – Rishabh Indoria ☺

What Does “Kubernetes” Mean?

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



What is Kubernetes?

- A **Production-Grade Container Orchestration System** Google-grown, based on Borg and Omega, systems that run inside of Google right now and are proven to work at Google for over 10 years.
- Google spawns billions of containers per week with these systems.
- Created by three Google employees initially during the summer of 2014; grew exponentially and became the first project to get donated to the CNCF.
- Hit the first production-grade version v1.0.1 in July 2015. Has continually released a new minor version every three months since v1.2.0 in March 2016. Lately v1.13.0 was released in December 2018.

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.”

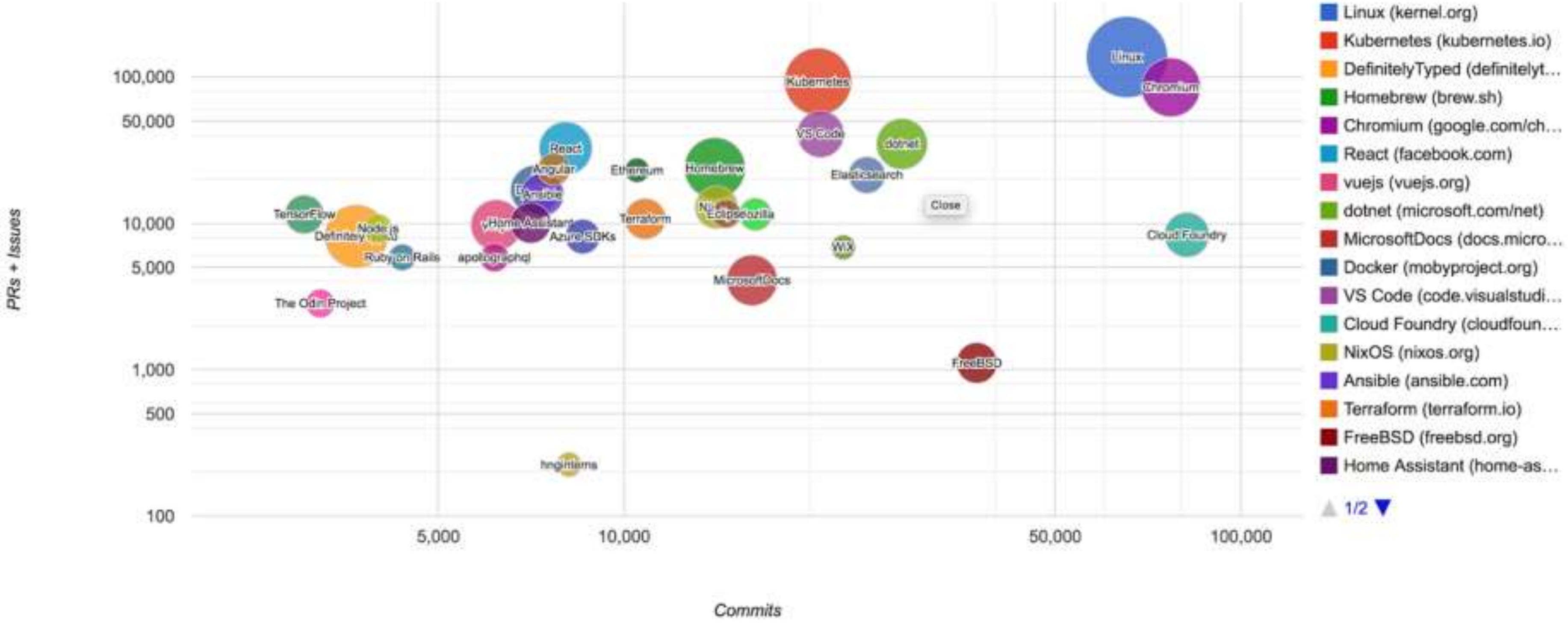
Project Stats

- Over 46,600 stars on Github
- 1800+ Contributors to K8s Core
- Most discussed Repository by a large margin
- **50,000+** users in Slack Team

10/2018

Project Stats

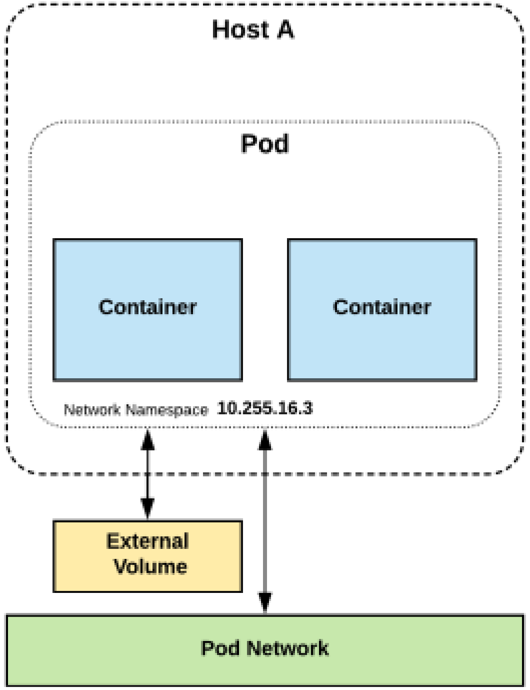
Top-30 projects in 2017



A Couple
Key Concepts...

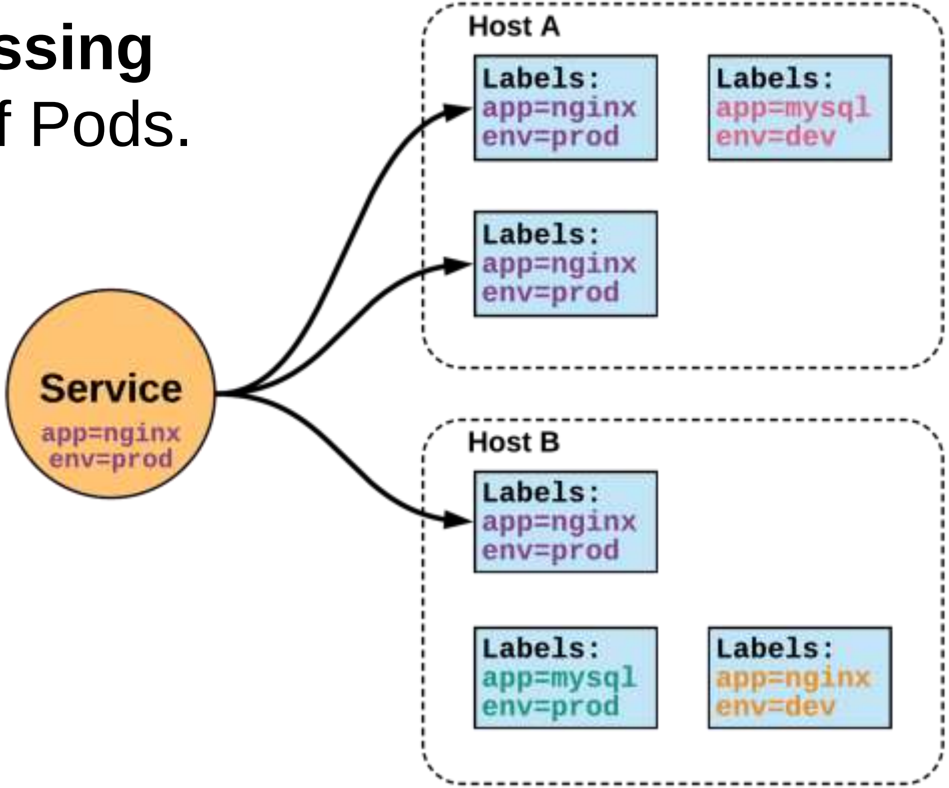
Pods

- **Atomic unit** or smallest "*unit of work*"of Kubernetes.
- Pods are **one or MORE containers** that share volumes and namespace.
- **They are also ephemeral!**



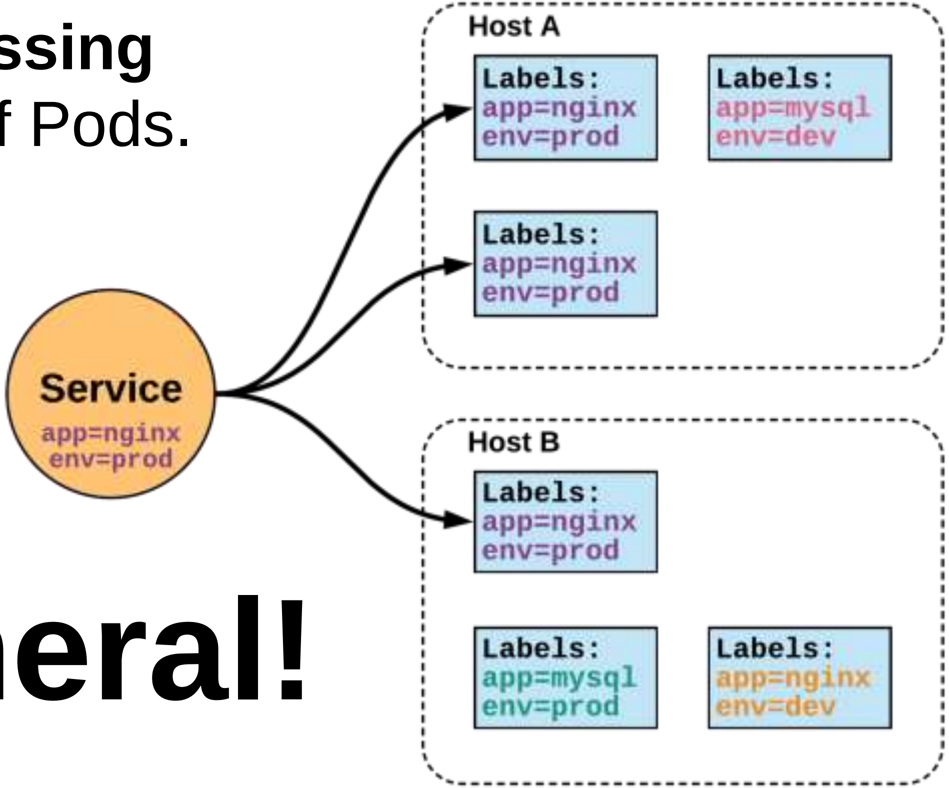
Services

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



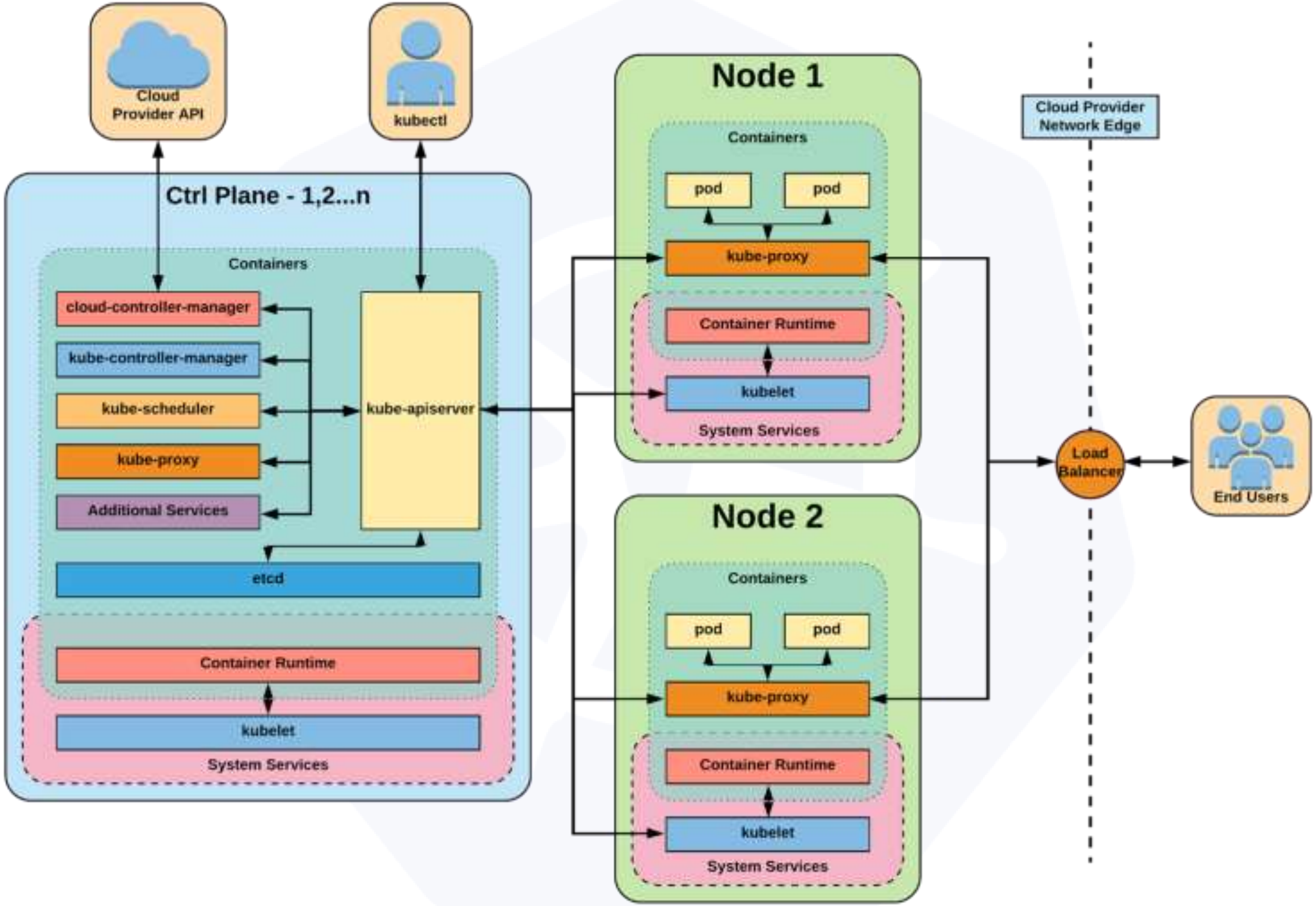
Services

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



NOT Ephemeral!

Architecture Overview

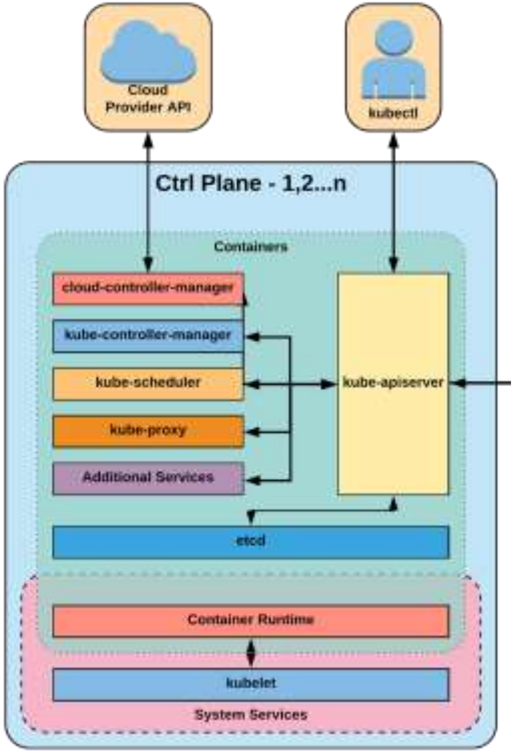

Cloud Provider API
kubectl
Ctrl Plane - 1,2...n
Containers
cloud-controller-manager
kube-controller-manager
kube-scheduler
kube-proxy
Additional Services
kube-apiserver
etcd
Container Runtime
kubelet
System Services
Node 1
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Node 2
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

Control Plane Components

Architecture Overview

Control Plane Components

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



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.

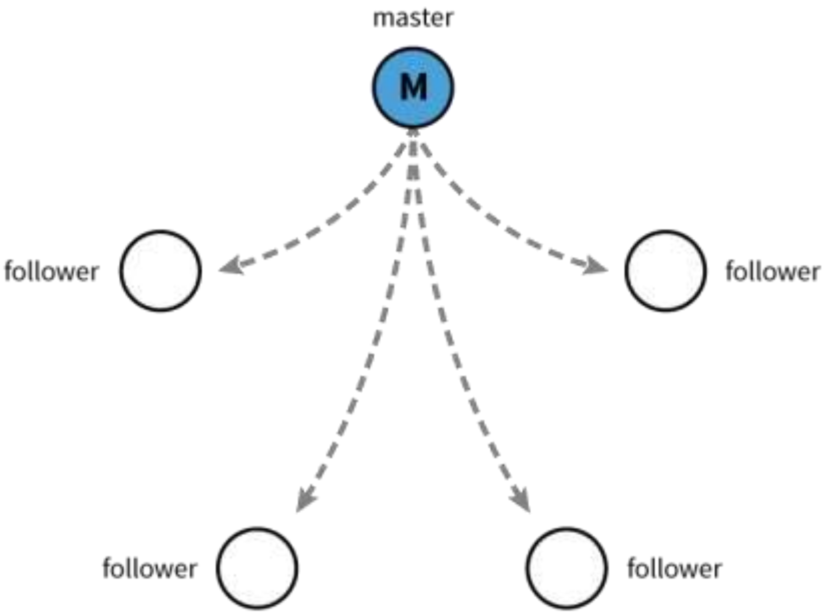


Image Source

https://raft.github.io/

kube-controller-manager

- Monitors the cluster state via the apiserver and **steers the cluster towards the desired state**.
- Node Controller: Responsible for noticing and responding when nodes go down.
- Replication Controller: Responsible for maintaining the correct number of pods for every replication controller object in the system.
- Endpoints Controller: Populates the Endpoints object (that is, joins Services & Pods).
- Service Account & Token Controllers: Create default accounts and API access tokens for new namespaces.

kube-scheduler

- Component on the master that watches newly created pods that have no node assigned, and selects a node for them to run on.
- Factors taken into account for scheduling decisions include individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference and deadlines.

cloud-controller-manager

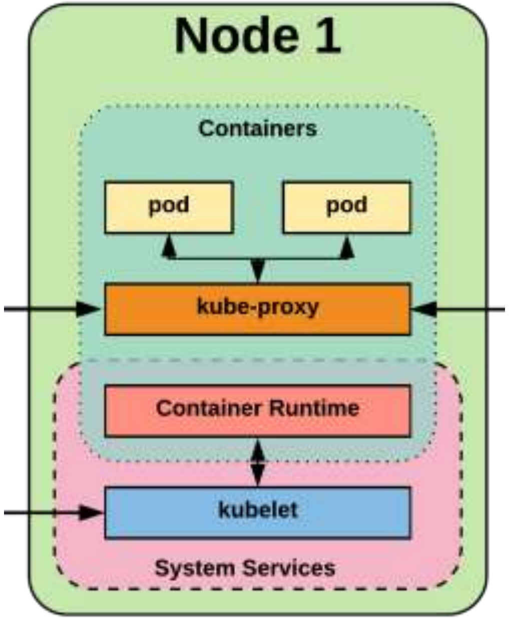
- Node Controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
- Route Controller: For setting up routes in the underlying cloud infrastructure
- Service Controller: For creating, updating and deleting cloud provider load balancers
- Volume Controller: For creating, attaching, and mounting volumes, and interacting with the cloud provider to orchestrate volumes

Node Components

Architecture Overview

Node Components

- kubelet
- kube-proxy
- Container Runtime Engine



kubelet

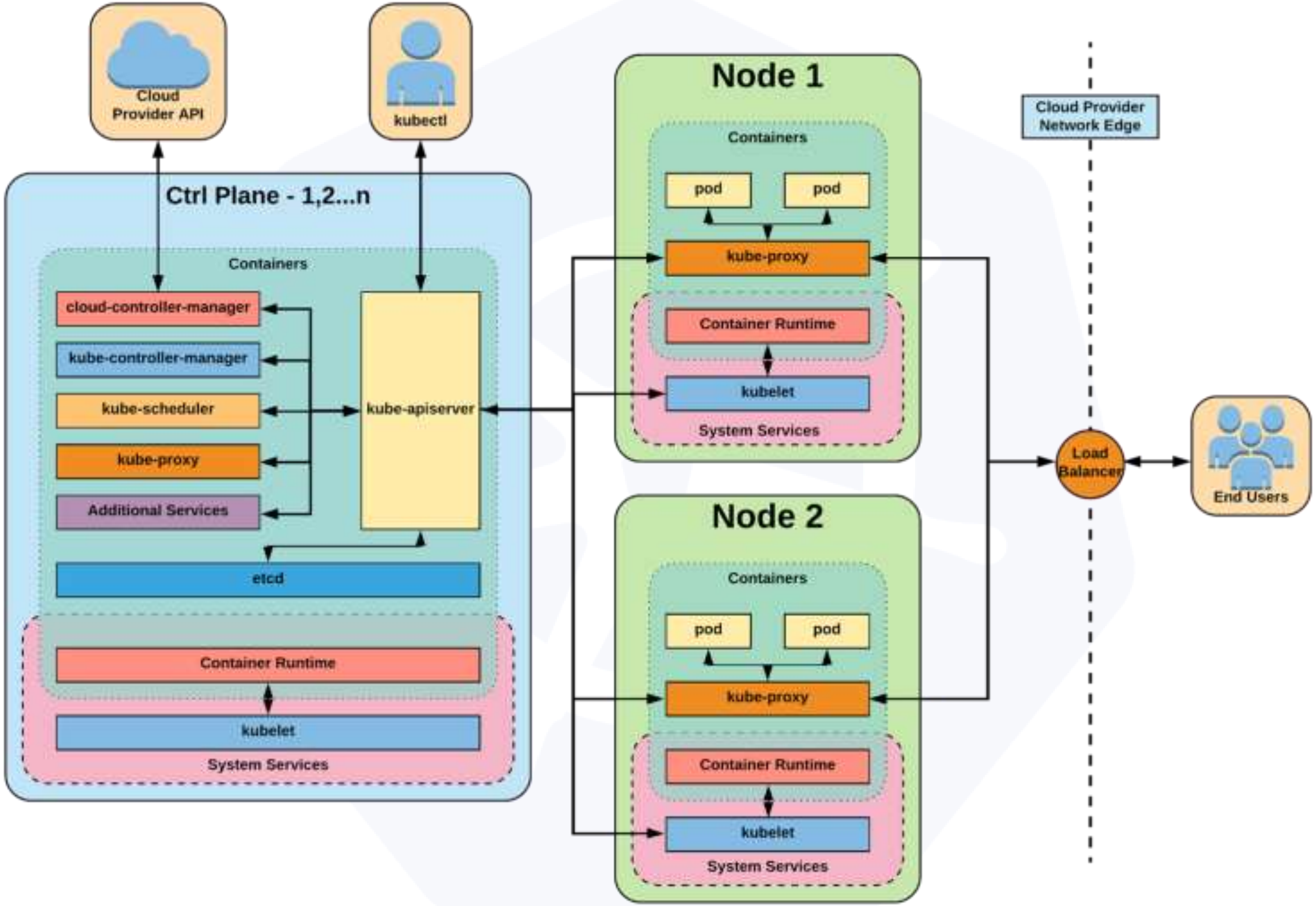
- An agent that runs on each node in the cluster. It makes sure that containers are running in a pod.
- The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy.

kube-proxy

- Manages the network rules on each node.
- Performs connection forwarding or load balancing for Kubernetes cluster services.

Container Runtime Engine

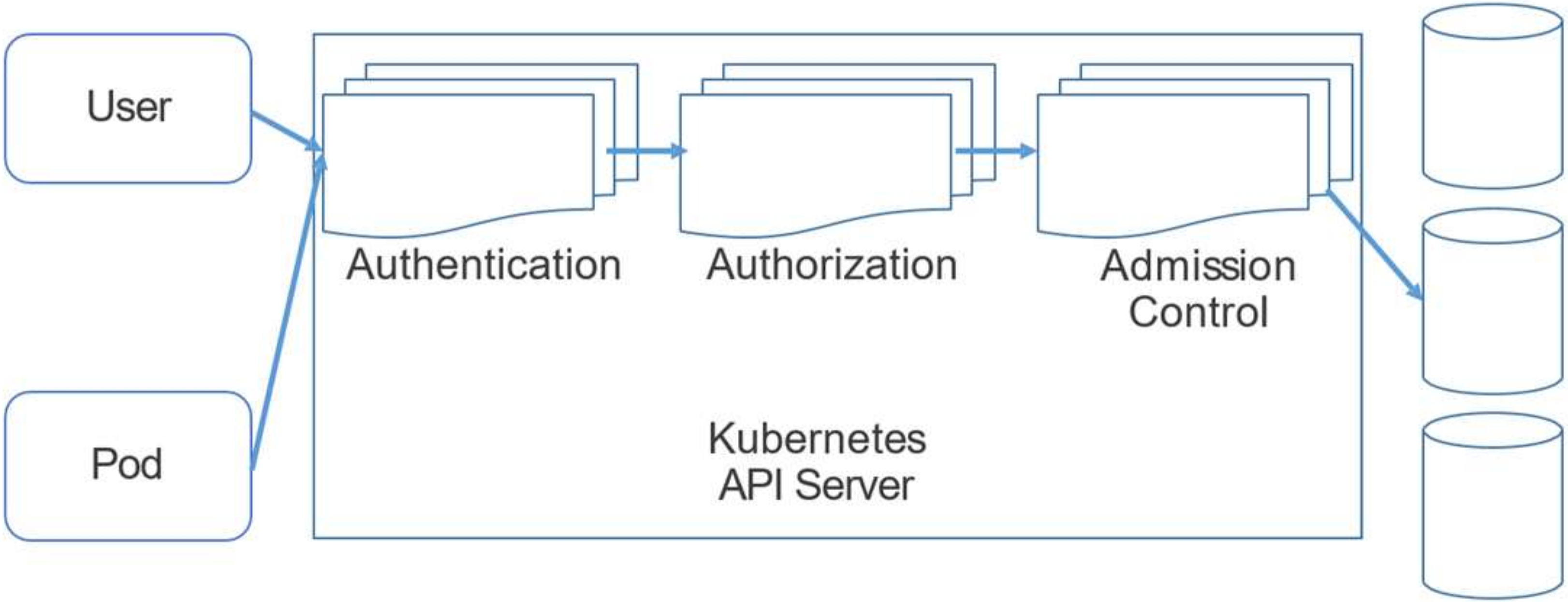
- A container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
    - Containerd (docker)
    - Cri-o
    - Rkt
    - Kata (formerly clear and hyper)
    - Virtlet (VM CRI compatible runtime)


Cloud Provider API
kubectl
Ctrl Plane - 1,2...n
Containers
cloud-controller-manager
kube-controller-manager
kube-scheduler
kube-proxy
Additional Services
kube-apiserver
etcd
Container Runtime
kubelet
System Services
Node 1
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Node 2
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

Security

Architecture Overview

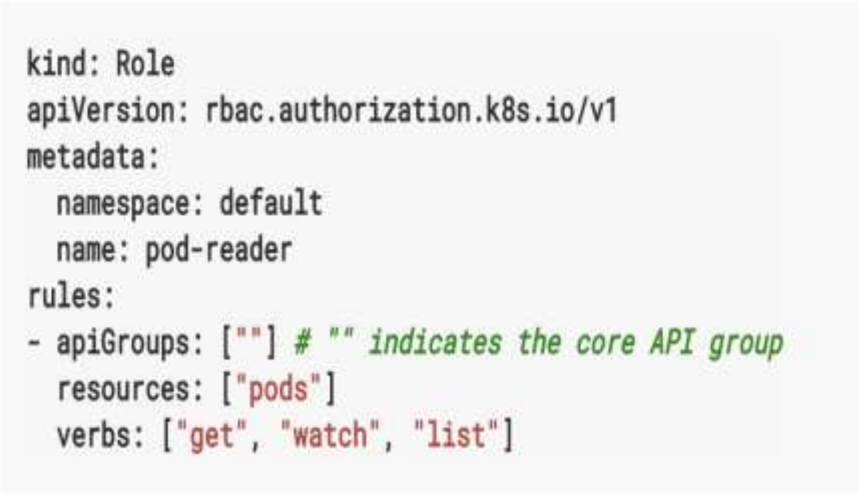
Access Control Diagram



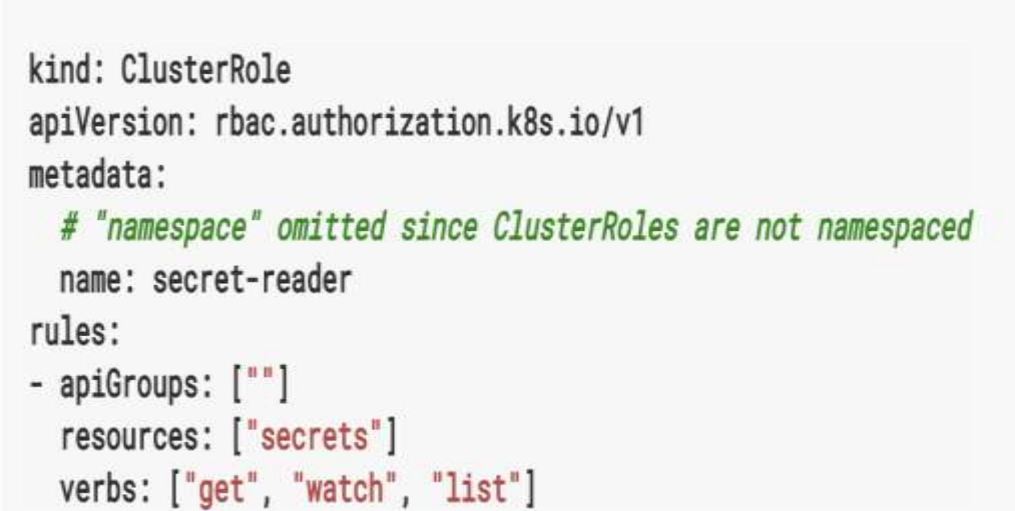
Authentication

- X509 Client Certs (CN used as user, Org fields as group) No way to revoke them!! – wip ☺
- Static Password File (password,user,uid,"group1,group2,group3")
- Static Token File (token,user,uid,"group1,group2,group3")
- Bearer Token (Authorization: Bearer 31ada4fd-ade)
- Bootstrap Tokens (Authorization: Bearer 781292.db7bc3a58fc5f07e)
- Service Account Tokens (signed by API server’s private TLS key or specified by file)

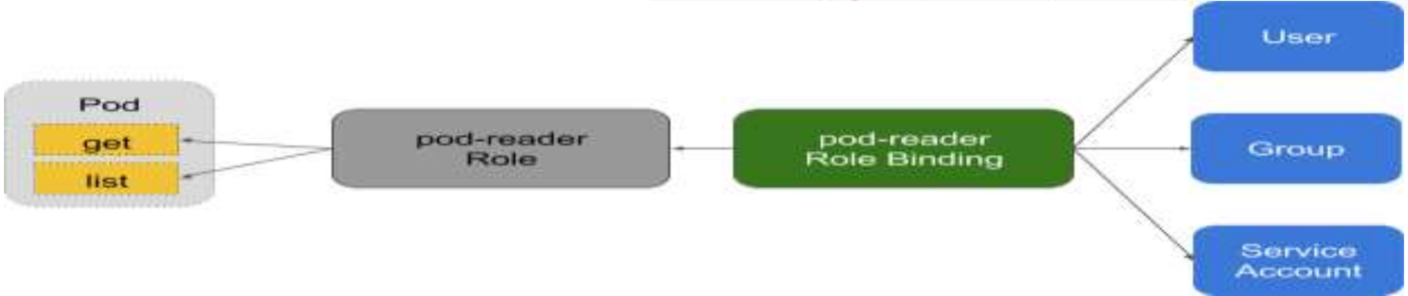
Role - Authorization



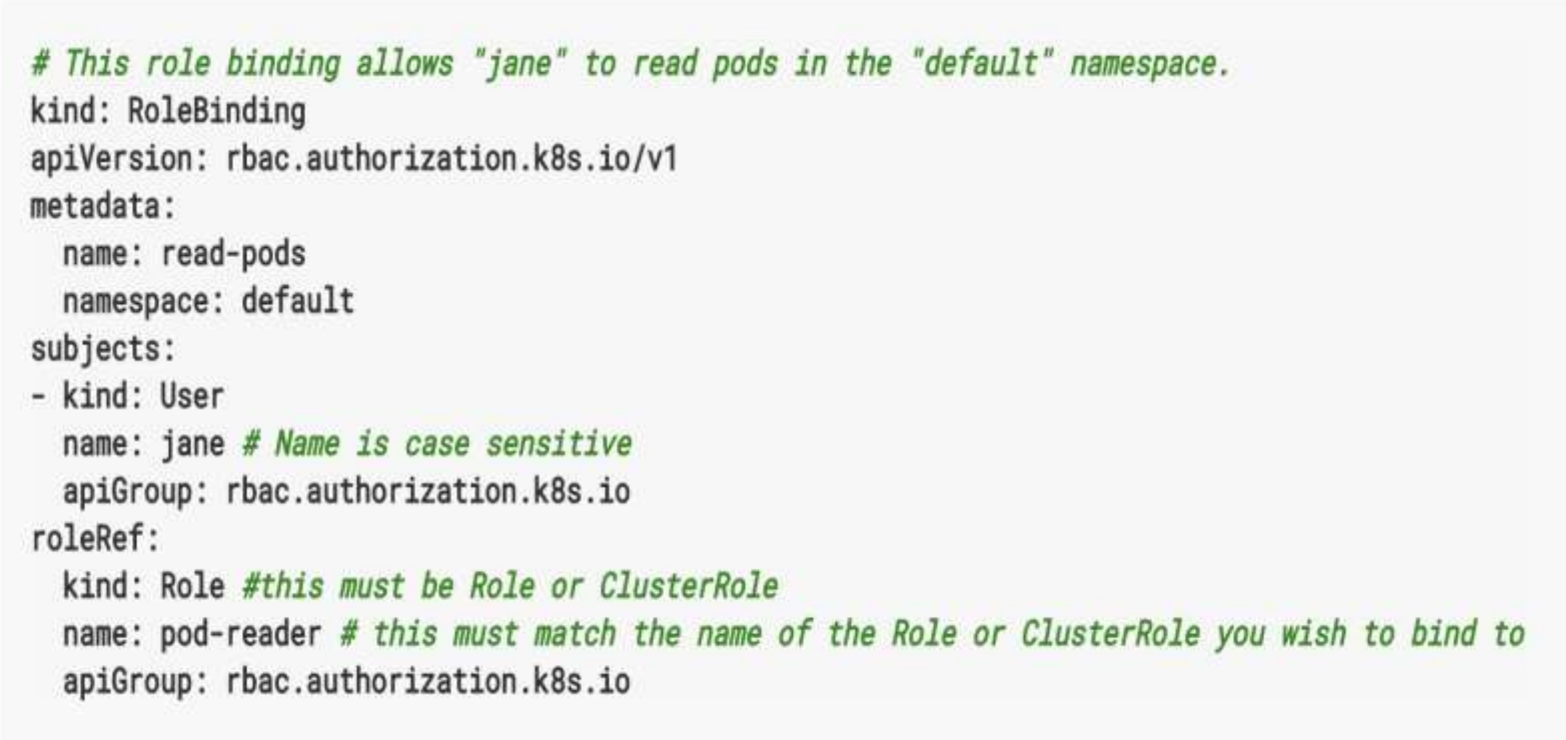
```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```



```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  # "namespace" omitted since ClusterRoles are not namespaced
  name: secret-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```

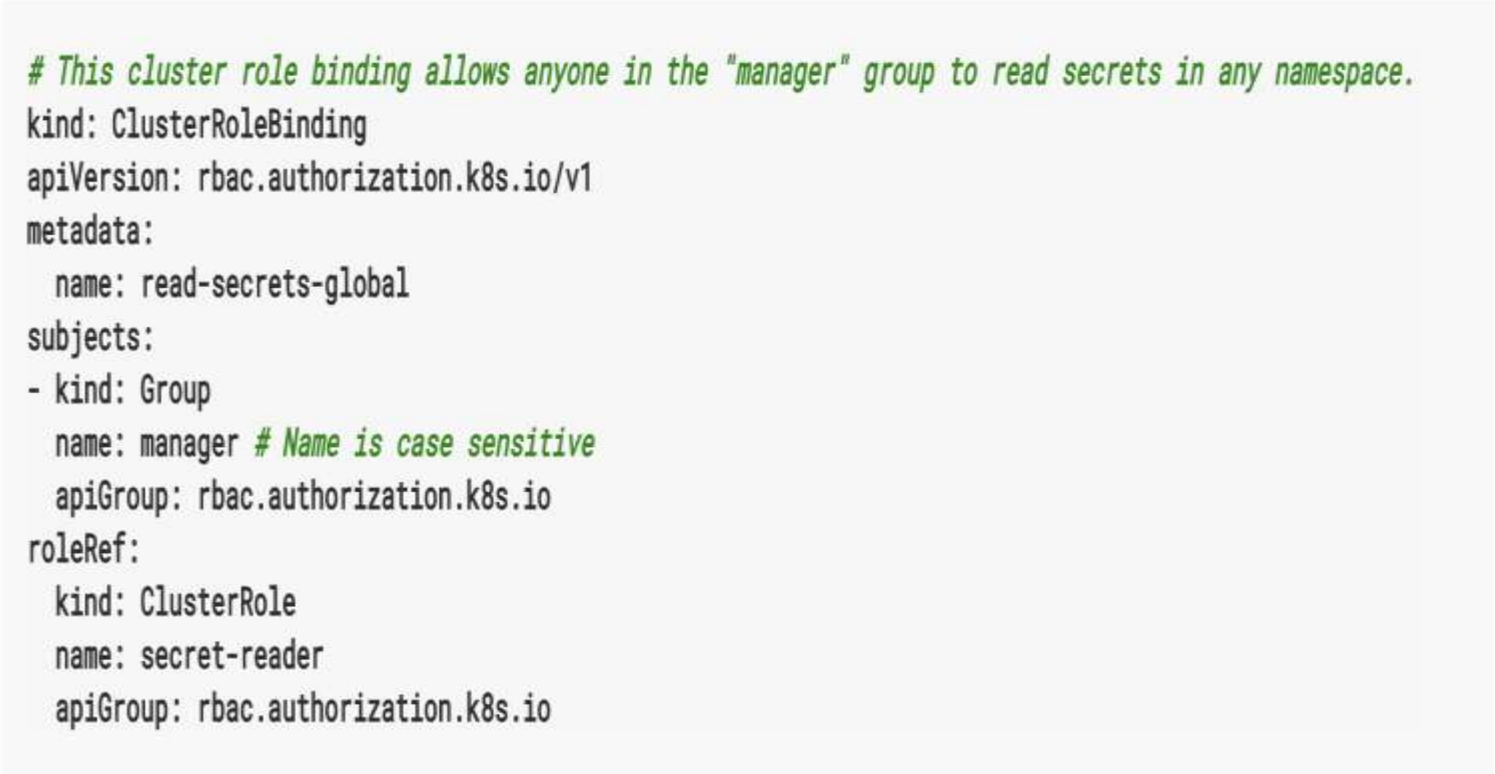


RoleBinding - Authorization



```yaml
# This role binding allows "jane" to read pods in the "default" namespace.
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role #this must be Role or ClusterRole
  name: pod-reader # this must match the name of the Role or ClusterRole you wish to bind to
  apiGroup: rbac.authorization.k8s.io
```

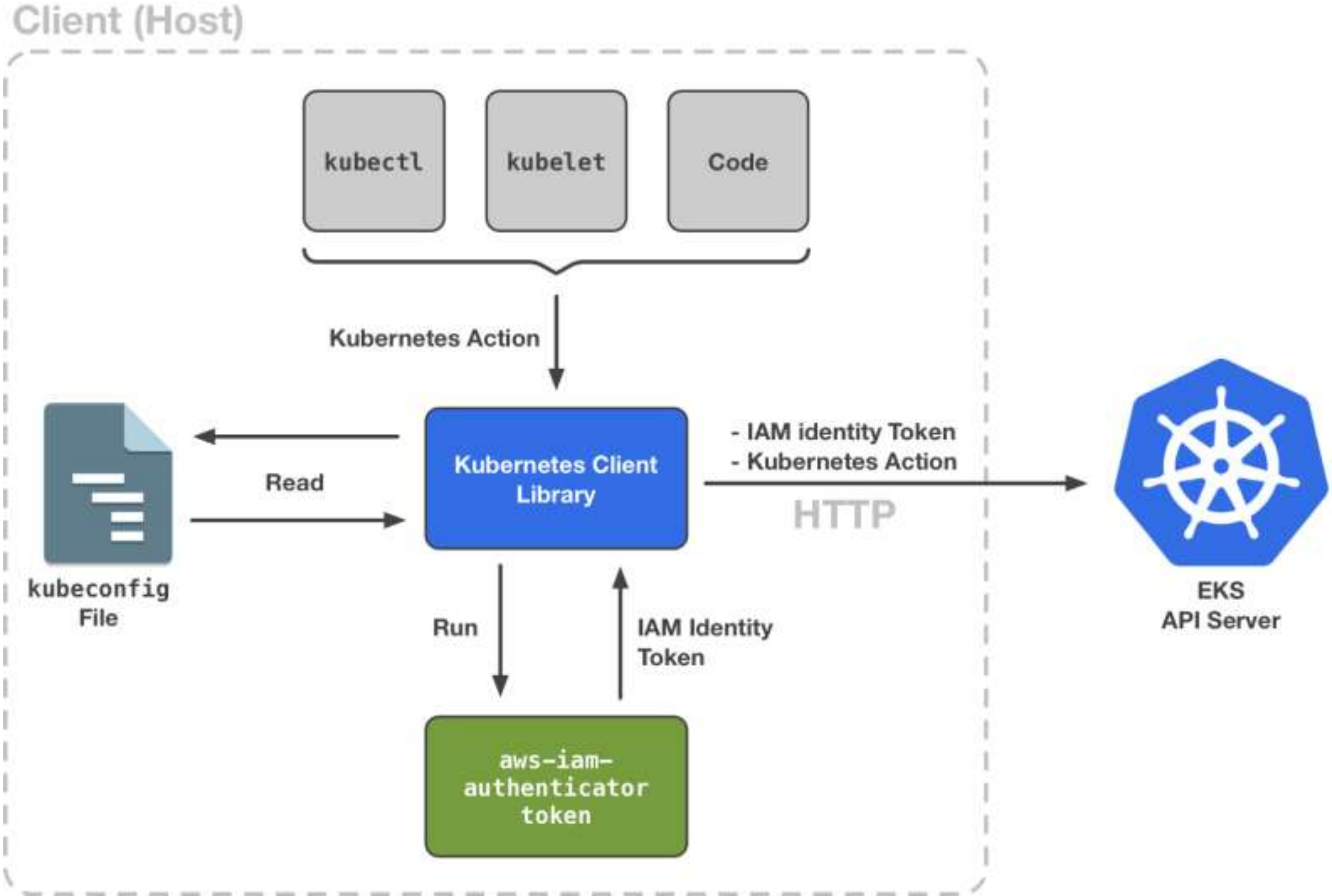
RoleBinding - Authorization



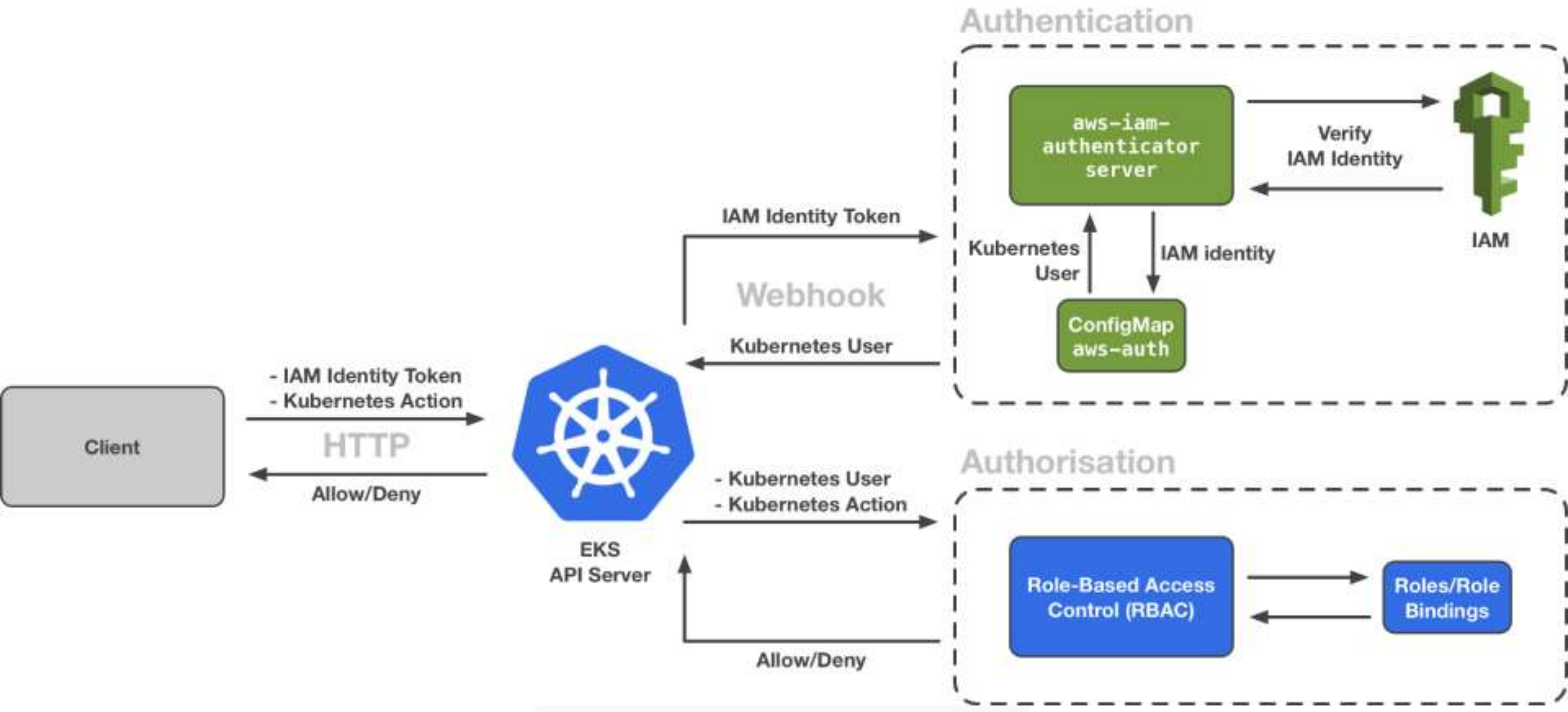
```
# This cluster role binding allows anyone in the "manager" group to read secrets in any namespace.
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: manager # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

Admission Control

- AlwaysPullImages
- DefaultStorageClass
- DefaultTolerationSeconds
- DenyEscalatingExec
- EventRateLimit
- ImagePolicyWebhook
- LimitRanger/ResourceQuota
- PersistentVolumeClaimResize
- PodSecurityPolicy



Client (Host)
kubectl
kubelet
Code
Kubernetes Action
kubeconfig
File
Read
Kubernetes Client Library
- IAM identity Token
- Kubernetes Action
HTTP
EKS
API Server
Run
IAM Identity
Token
aws-iam-authenticator token


Authentication
aws-iam-authenticator server
Verify IAM Identity
IAM
IAM Identity Token
Kubernetes User
IAM identity
Webhook
ConfigMap aws-auth
Kubernetes User
- IAM Identity Token
- Kubernetes Action
Client
HTTP
Allow/Deny
EKS API Server
Authorisation
- Kubernetes User
- Kubernetes Action
Role-Based Access Control (RBAC)
Roles/Role Bindings
Allow/Deny

Request/Response

```
{
  "apiVersion": "authentication.k8s.io/v1beta1",
  "kind": "TokenReview",
  "spec": {
    "token": "(BEARERTOKEN)"
  }
}
```

```
{
  "apiVersion": "authentication.k8s.io/v1beta1",
  "kind": "TokenReview",
  "status": {
    "authenticated": true,
    "user": {
      "username": "janedoe@example.com",
      "uid": "42",
      "groups": [
        "developers",
        "qa"
      ]
    }
  }
}
```

Networking

Architecture Overview

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).

Core
Objects and API

- Namespaces
- Pods
- Labels
- Selectors
- Services

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
```

Pod Examples

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

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

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}”]
```

Pod Template

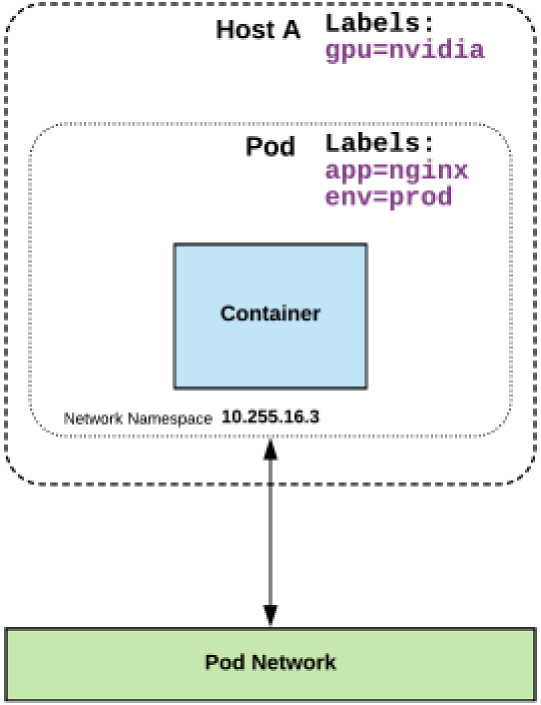
- Workload Controllers manage instances of Pods based off a provided template.
- Pod Templates are Pod specs with limited metadata.
- Controllers use Pod Templates to make actual pods.

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

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

Labels

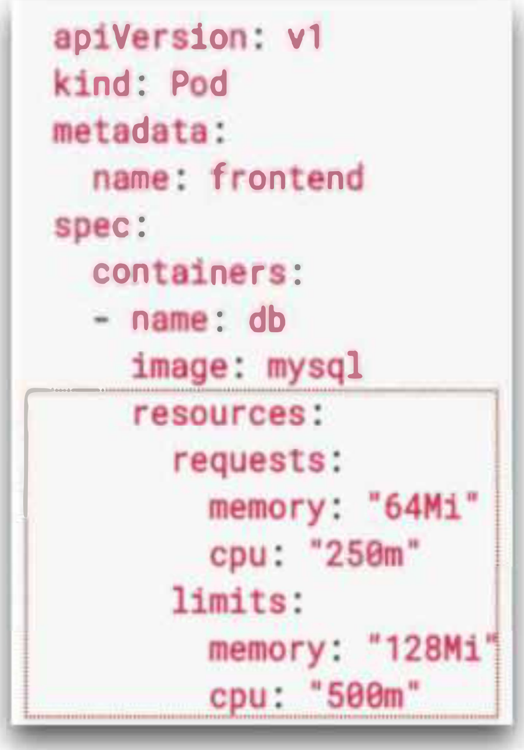
- key-value pairs that are used to identify, describe and group together related sets of objects or resources.
- **NOT** characteristic of uniqueness.
- Have a strict syntax with a slightly limited character set*.



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

Resource Model

- **Request**: amount of a resource allowed to be used, with a strong guarantee of availability
  - CPU (seconds/second), RAM (bytes)
  - Scheduler will not over-commit requests
- **Limit**: max amount of a resource that can be used, regardless of guarantees
  - scheduler **ignores** limits



```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: db
    image: mysql
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

- Mapping to Docker
  - —cpu-shares=requests.cpu
  - —cpu-quota=limits.cpu
  - —cpu-period=100ms
  - —memory=limits.memory

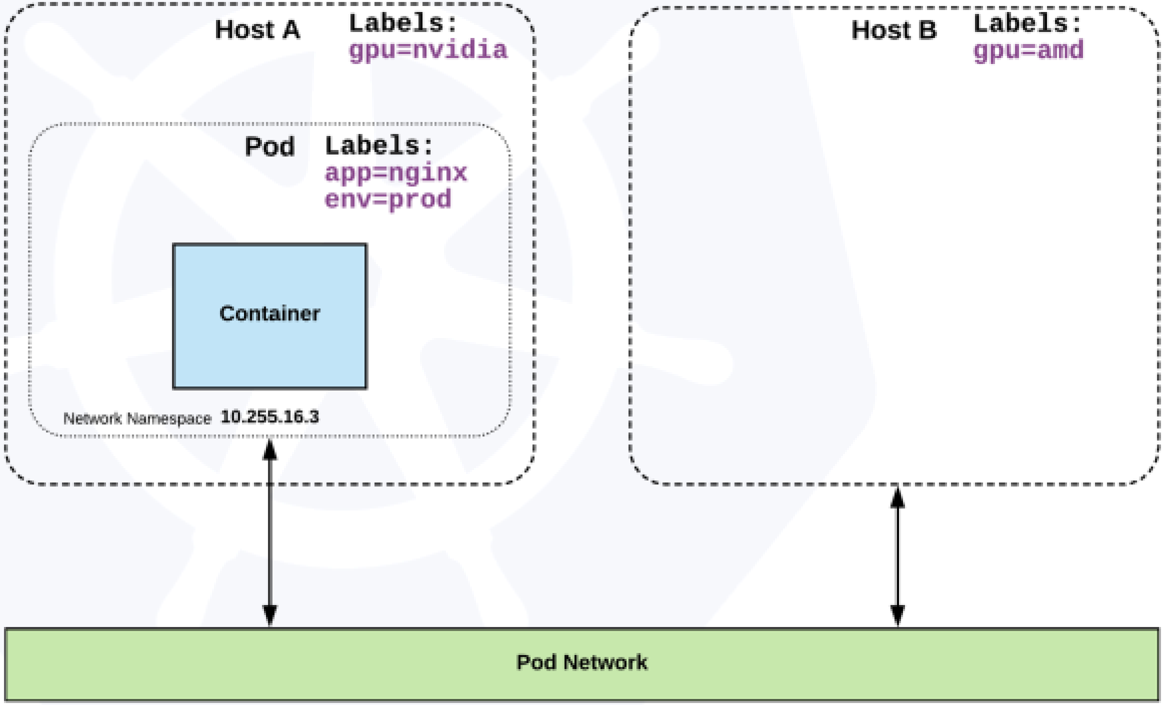
Selectors

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** (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

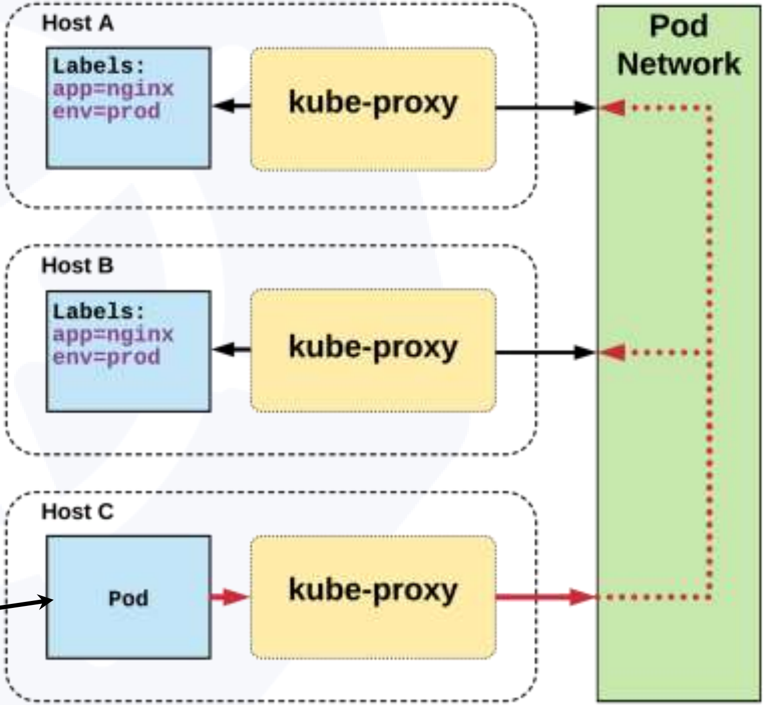
ClusterIP services exposes a service on a strictly cluster internal virtual IP.

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

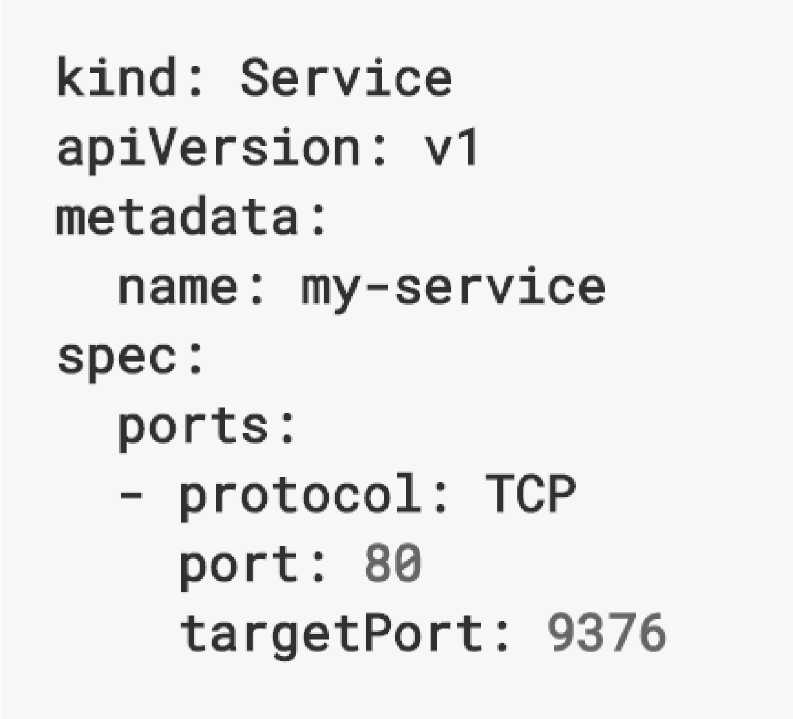
Cluster IP Service

```
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
```

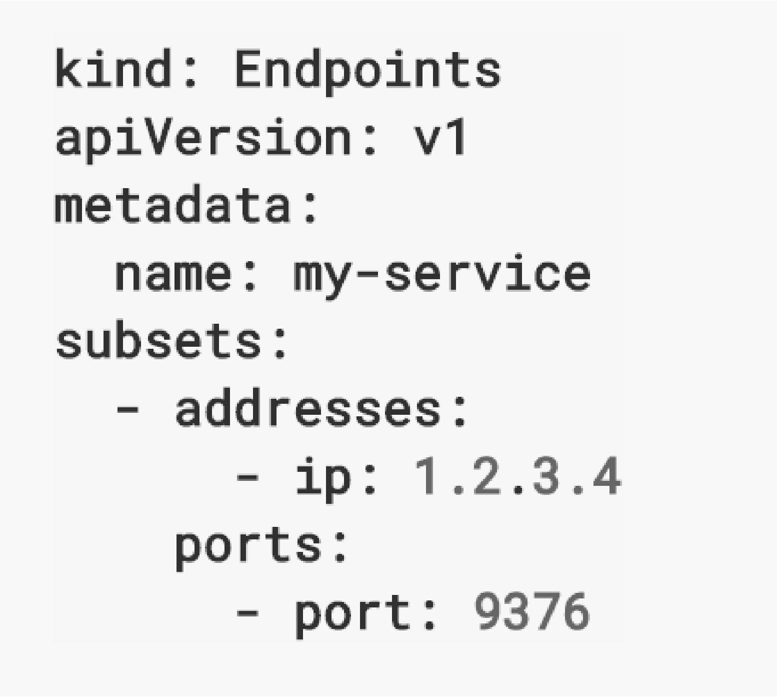
```
/ # nslookup example-prod.default.svc.cluster.local

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



ClusterIP Service Without Selector



```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```



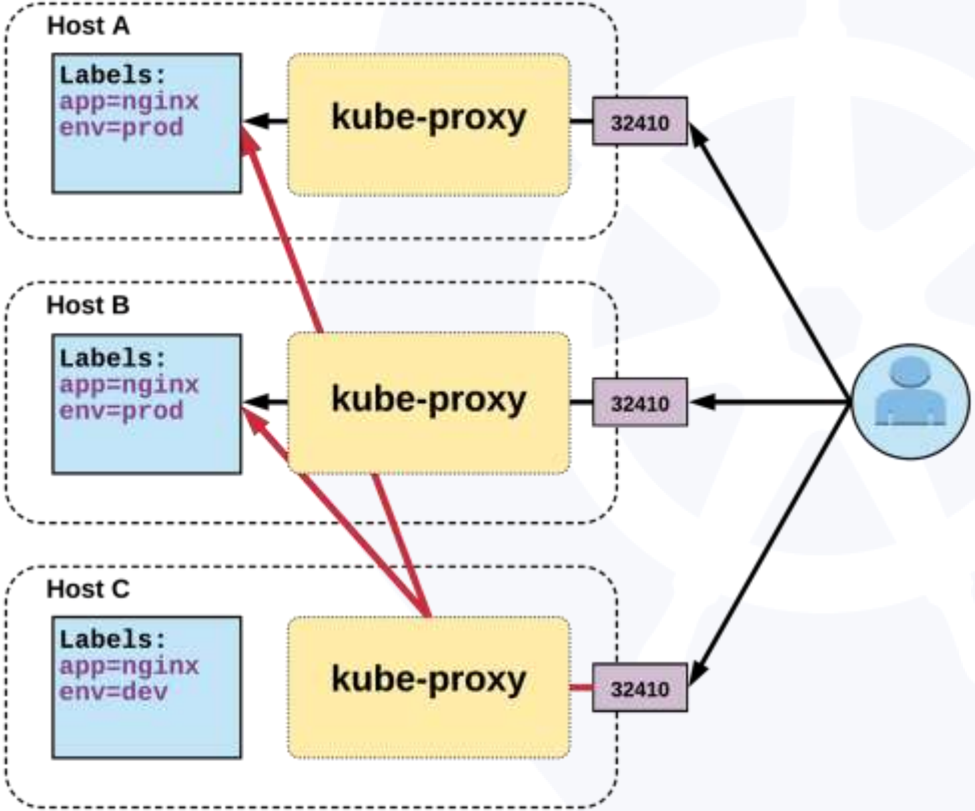
```
kind: Endpoints
apiVersion: v1
metadata:
  name: my-service
subsets:
  - addresses:
      - ip: 1.2.3.4
    ports:
      - port: 9376
```

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.

```yaml
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



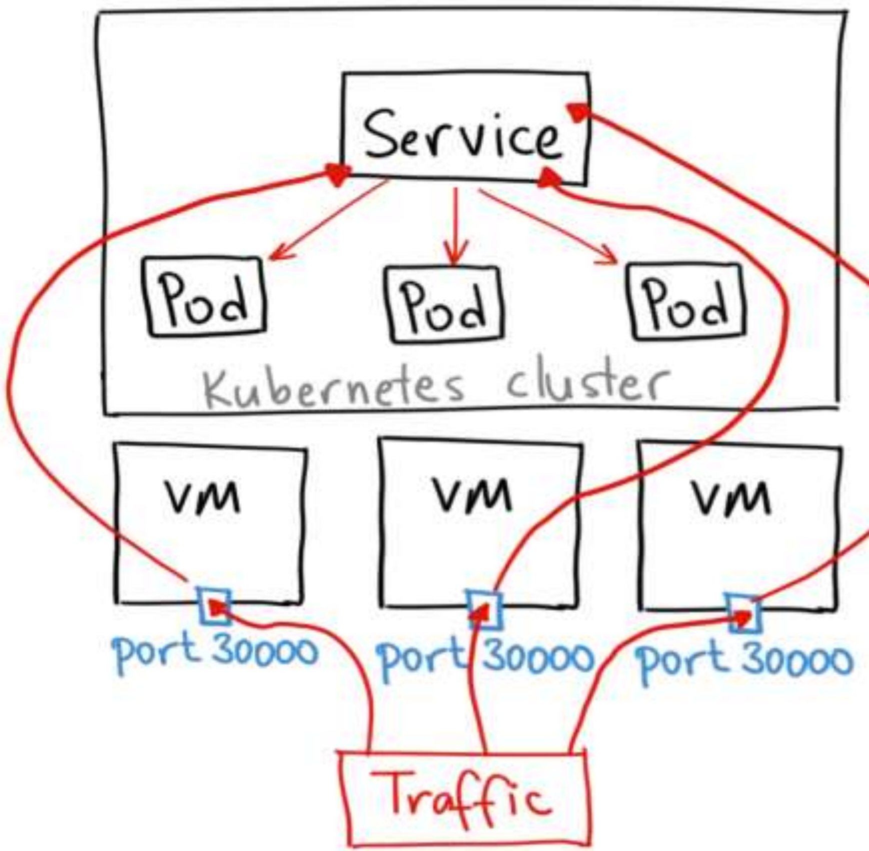
```
Name:         example-prod
Selector:     app=nginx,env=prod
Type:         NodePort
IP:           10.96.28.176
Port:         <unset>  80/TCP
TargetPort:   80/TCP
NodePort:     <unset>  32410/TCP
Endpoints:    10.255.16.3:80,
              10.255.16.4:80
```

LoadBalancer Service

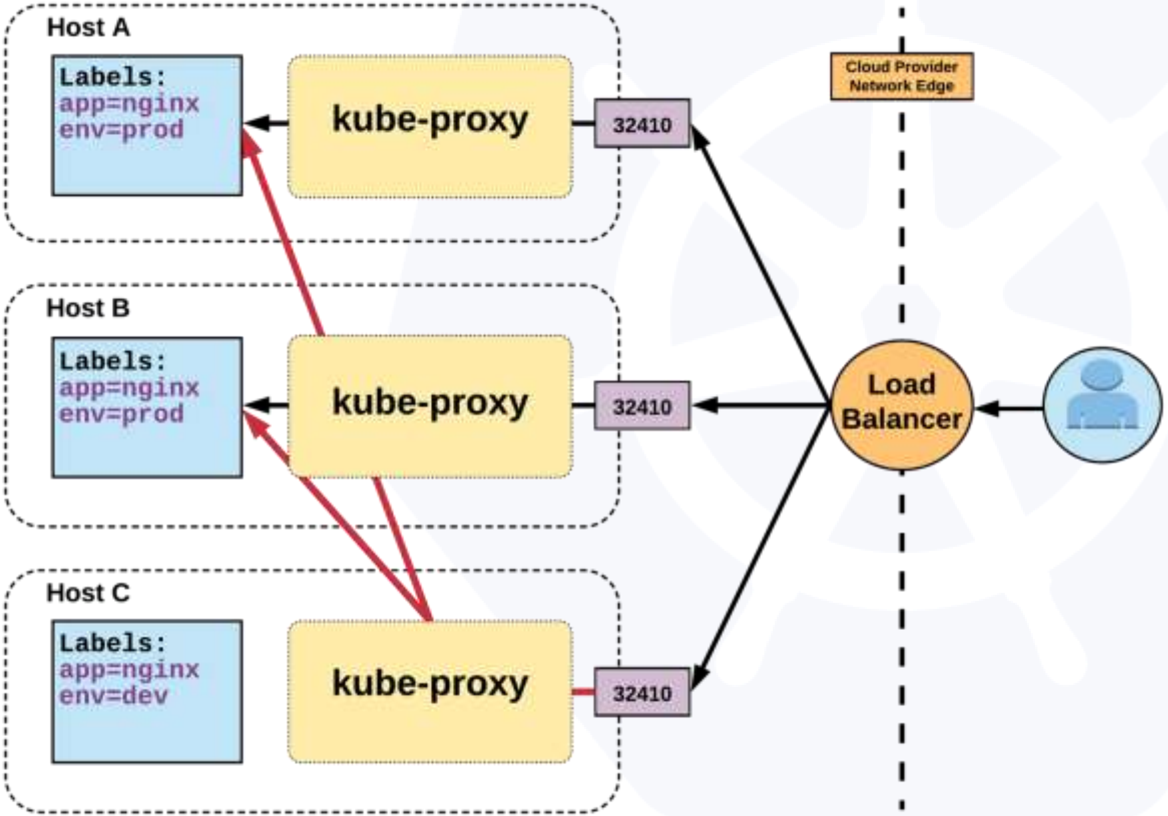
- **LoadBalancer** services extend **NodePort.**
- Works in conjunction with an external system to map a cluster external IP to the exposed service.

```
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



LoadBalancer Service



```
Name:         example-prod
Selector:     app=nginx,env=prod
Type:         LoadBalancer
IP:           10.96.28.176
LoadBalancer
Ingress:      172.17.18.43
Port:         <unset>  80/TCP
TargetPort:   80/TCP
NodePort:     <unset>  32410/TCP
Endpoints:    10.255.16.3:80,
              10.255.16.4:80
```

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
```

Ingress – Name Based Routing

- An API object that manages external access to the services in a cluster
- Provides load balancing, SSL termination and name/path-based virtual hosting
- Gives services externally-reachable URLs

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: name-virtual-host-ingress
spec:
 rules:
 - host: first.bar.com
  http:
   paths:
   - backend:
     serviceName: service1
     servicePort: 80
 - host: second.foo.com
  http:
   paths:
   - backend:
     serviceName: service2
     servicePort: 80
 - http:
   paths:
   - backend:
     serviceName: service3
     servicePort: 80
```

Ingress – Path Based Routing

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: service1
          servicePort: 4200
      - path: /bar
        backend:
          serviceName: service2
          servicePort: 8080
```

Lab - github.com/mrbobbytables/k8s-intro-tutorials/blob/master/core

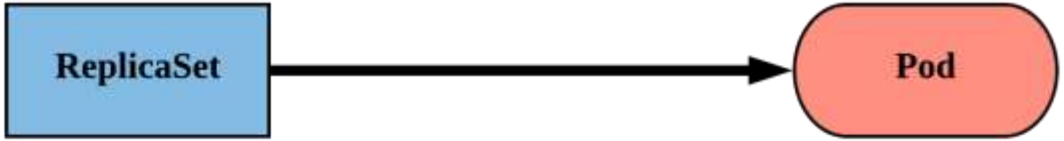
Exploring the Core

Workloads

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

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

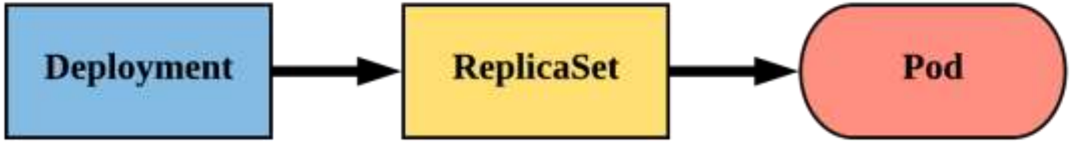
```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-example
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  template:
    metadata:
      labels:
        app: nginx
        env: prod
    spec:
      containers:
      - name: nginx
        image: nginx:stable-alpine
        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-mkll2   1/1      Running    0          1h
```

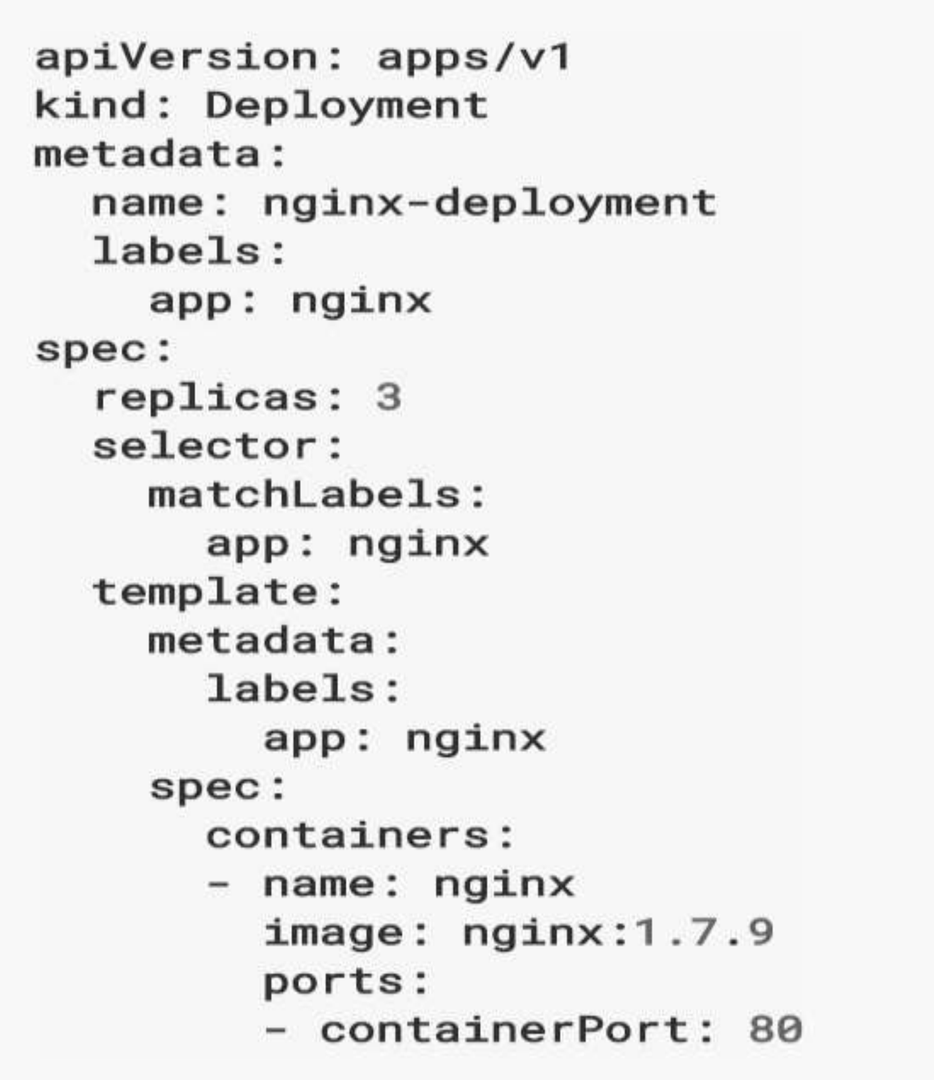
```
$ kubectl describe rs rs-example
Name:         rs-example
Namespace:    default
Selector:     app=nginx,env=prod
Labels:       app=nginx
              env=prod
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  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-mkll2
  Normal  SuccessfulCreate  16s   replicaset-controller  Created pod: rs-example-b7bcg
  Normal  SuccessfulCreate  16s   replicaset-controller  Created pod: rs-example-9l4dt
```

Deployment

- Way 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



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

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.

```yaml
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>
```

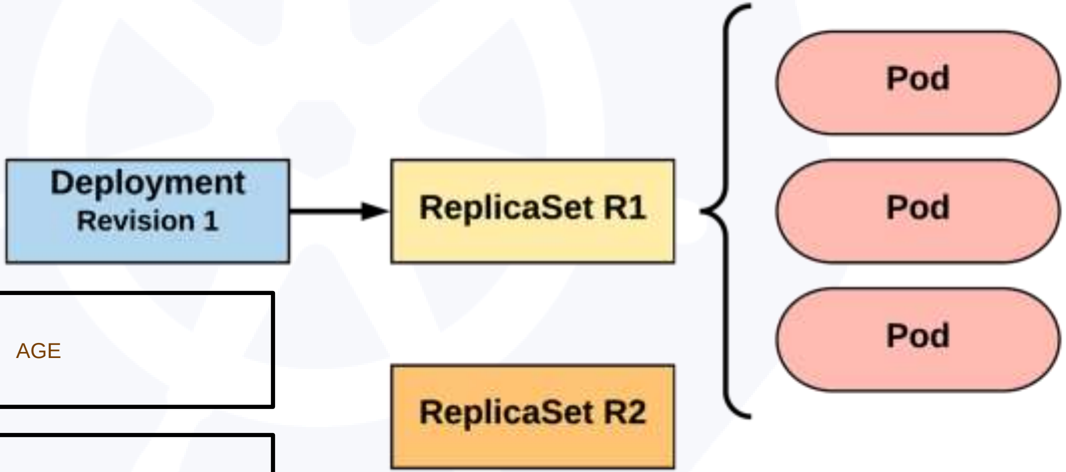
Deployment

```
$ kubectl create deployment test --image=nginx
$ kubectl set image deployment test nginx=nginx:1.9.1 --record

$ kubectl rollout history deployment test
deployments "test"
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl set image deployment test nginx=nginx:1.9.1 --record=true

$ kubectl annotate deployment test kubernetes.io/change-cause="image updated to 1.9.1"

$ kubectl rollout undo deployment test
$ kubectl rollout undo deployment test --to-revision=2

$ kubectl rollout history deployment test
deployments "test"
REVISION  CHANGE-CAUSE
2         kubectl set image deployment test nginx=nginx:1.9.1 --record=true
3         <none>

kubectl scale deployment test --replicas=10
kubectl rollout pause deployment test
kubectl rollout resume deployment test
```

RollingUpdate Deployment

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

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



```
$ 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
```

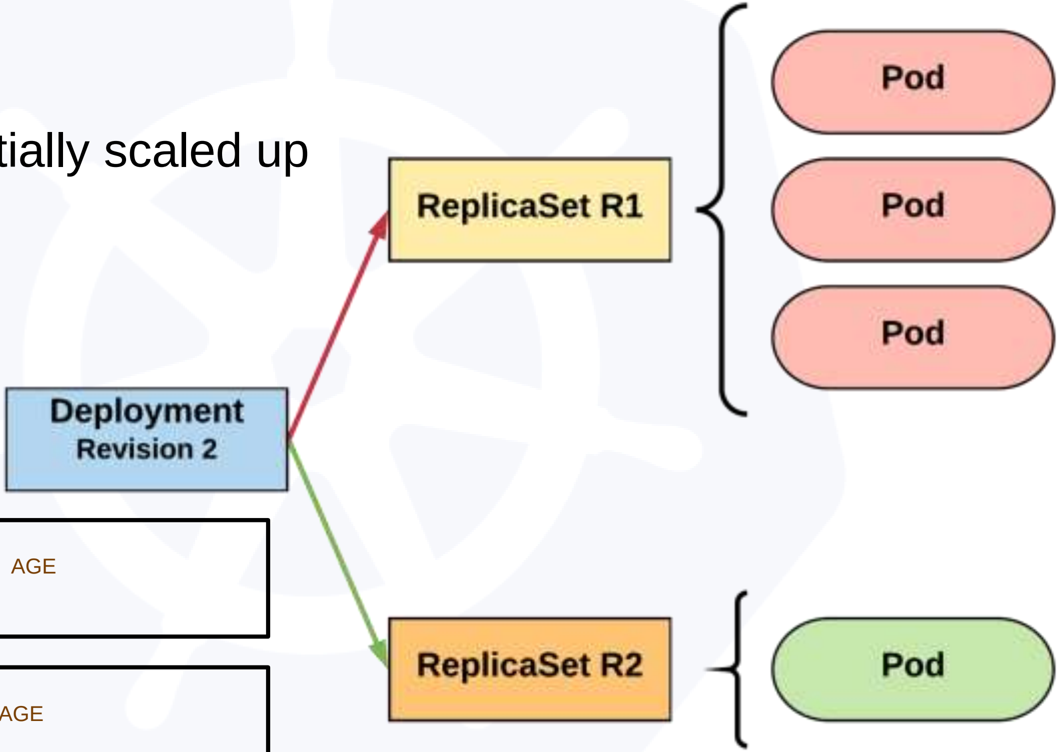
RollingUpdate Deployment

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

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

```
$ 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
```



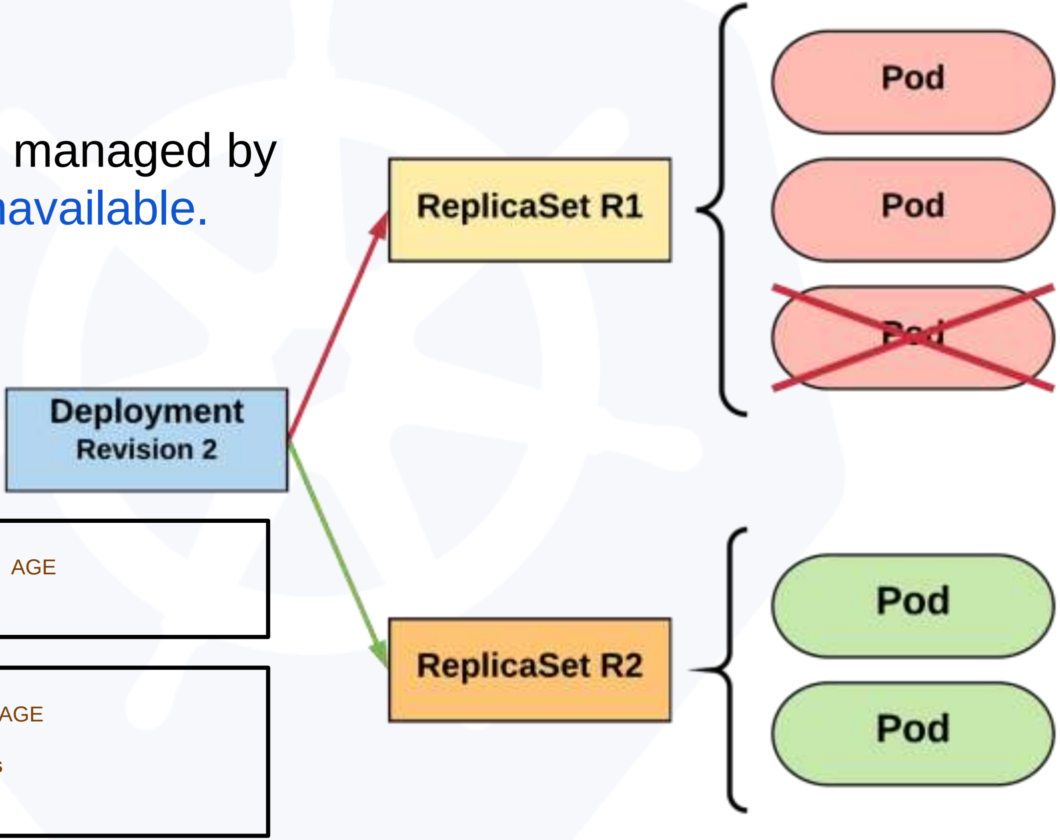
RollingUpdate Deployment

Phase out of old Pods managed by maxSurge and maxUnavailable.

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

```
$ kubectl get replicaset
NAME                 DESIRED   CURRENT   READY    AGE
mydep-54f7ff7d6d        2         2         2       8s
mydep-6766777fff        2         2         2       5h
```

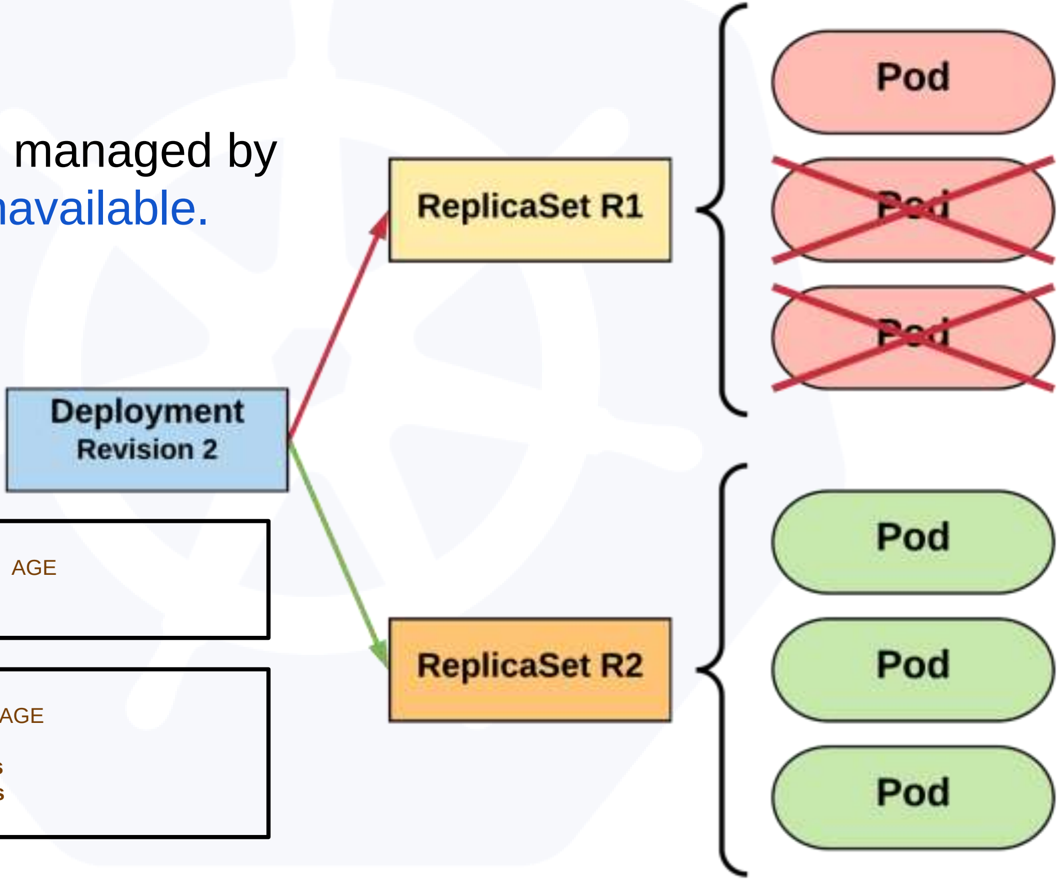
```
$ kubectl get pods
NAME                     READY    STATUS    RESTARTS   AGE
mydep-54f7ff7d6d-9gvll    1/1     Running   0          5s
mydep-54f7ff7d6d-cqvlq    1/1     Running   0          2s
mydep-6766777fff-9r2zn    1/1     Running   0          5h
mydep-6766777fff-hsfz9    1/1     Running   0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



RollingUpdate Deployment

Phase out of old Pods managed by maxSurge and maxUnavailable.

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



```
$ kubectl get replicaset
NAME                DESIRED   CURRENT   READY    AGE
mydep-54f7ff7d6d        3         3        3       10s
mydep-6766777fff        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-6766777fff-9r2zn   1/1      Running   0        5h
```

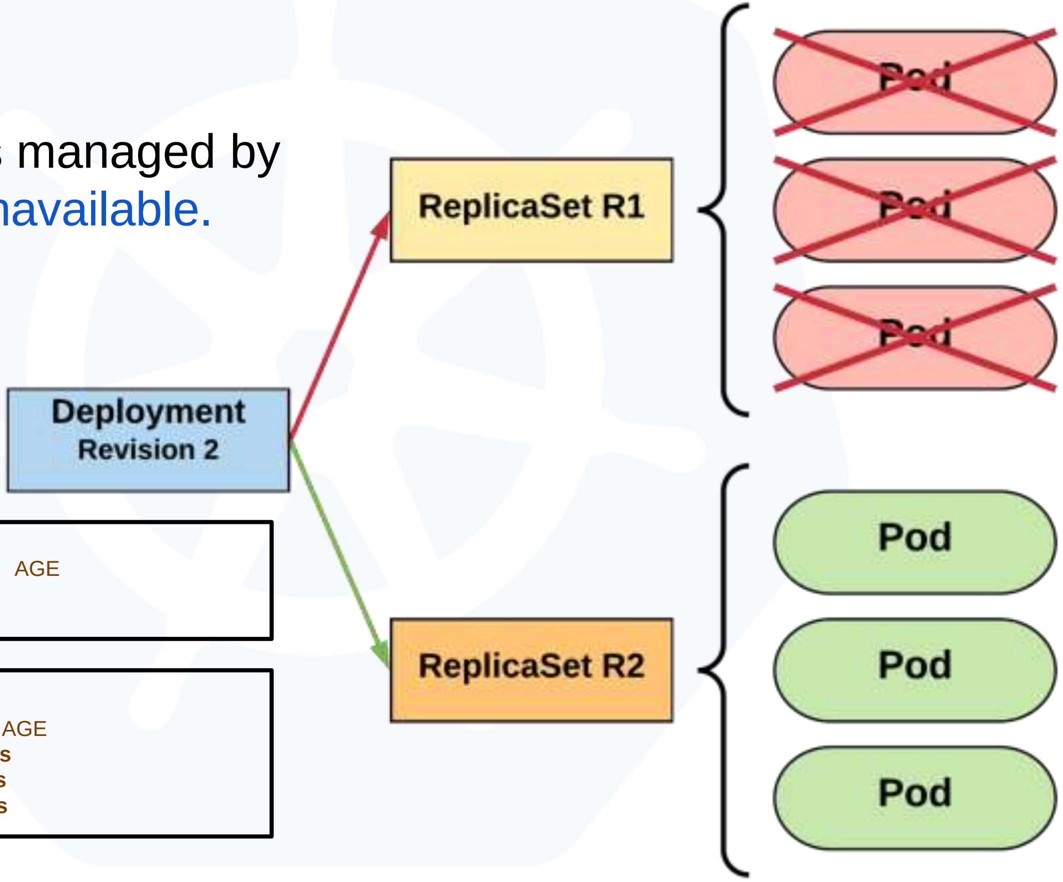
RollingUpdate Deployment

Phase out of old Pods managed by maxSurge and maxUnavailable.

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

```
$ kubectl get replicaset
NAME                    DESIRED   CURRENT   READY    AGE
mydep-54f7ff7d6d           3         3         3       13s
mydep-6766777fff           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
```



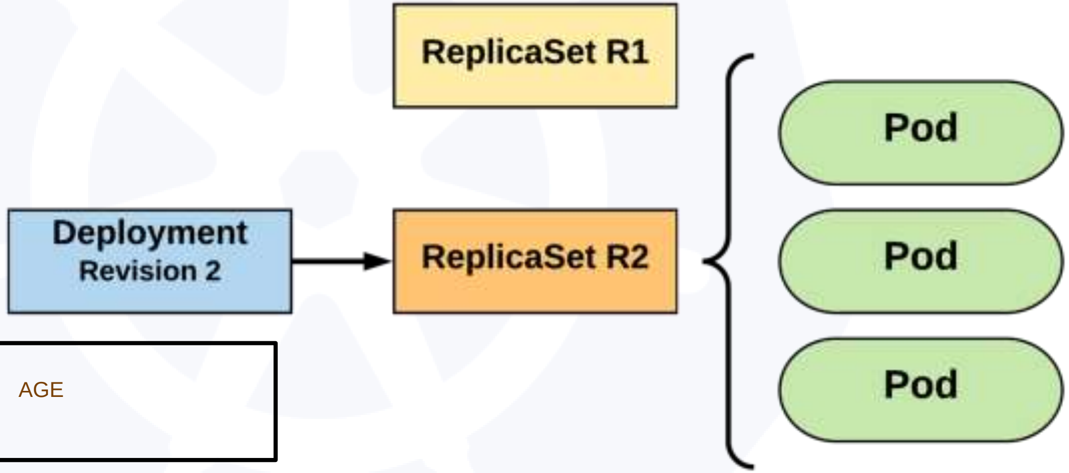
RollingUpdate Deployment

Updated to new deployment revision completed.

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

```
$ kubectl get replicaset
NAME                DESIRED   CURRENT   READY    AGE
mydep-54f7ff7d6d       3         3         3       15s
mydep-6766777fff       0         0         0       5h
```

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



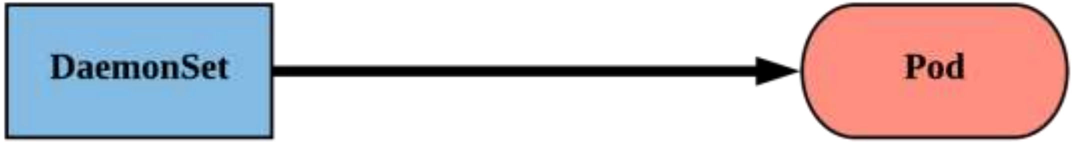
Taints and Tolerations

```
$ kubectl taint nodes node1 key=value:NoSchedule

tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule”

----------------------------------------------
tolerations:
- operator: "Exists"

tolerations:
- key: "key"
  operator: "Exists”

tolerations:
- key: "key1"
  operator: "Equal"
  value: "value1"
  effect: "NoExecute"
  tolerationSeconds: 3600
```

```
$ kubectl taint nodes node1 gpu=nvidia:NoSchedule

apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
  tolerations:
  - key: gpu
    value: nvidia
    effect: NoSchedule
```

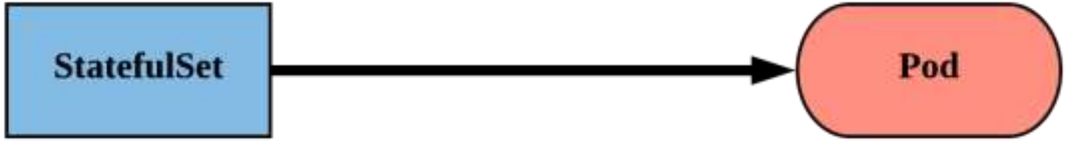
DaemonSet

- Ensure that all nodes matching certain criteria will run an instance of the supplied Pod.
- Are ideal for cluster wide services such as log forwarding or monitoring.



StatefulSet

- Tailored to managing Pods that must persist or maintain state.
- Pod lifecycle will be ordered and follow consistent patterns.
- Assigned a unique ordinal name following the convention of ‘<*statefulset name*>-<*ordinal index*>’.



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
```

```
/ # 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
```

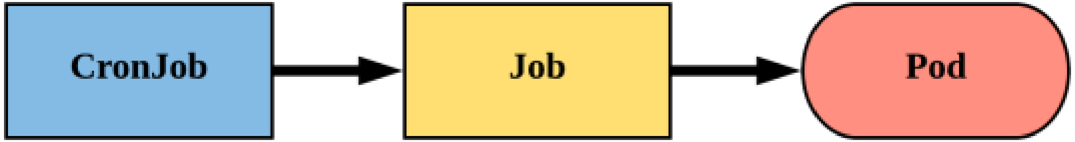
```
$ kubectl get pods
NAME            READY    STATUS    RESTARTS   AGE
sts-example-0   1/1      Running   0          11m
sts-example-1   1/1      Running   0          11m
```

```
/ # dig sts-example-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
```

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
```

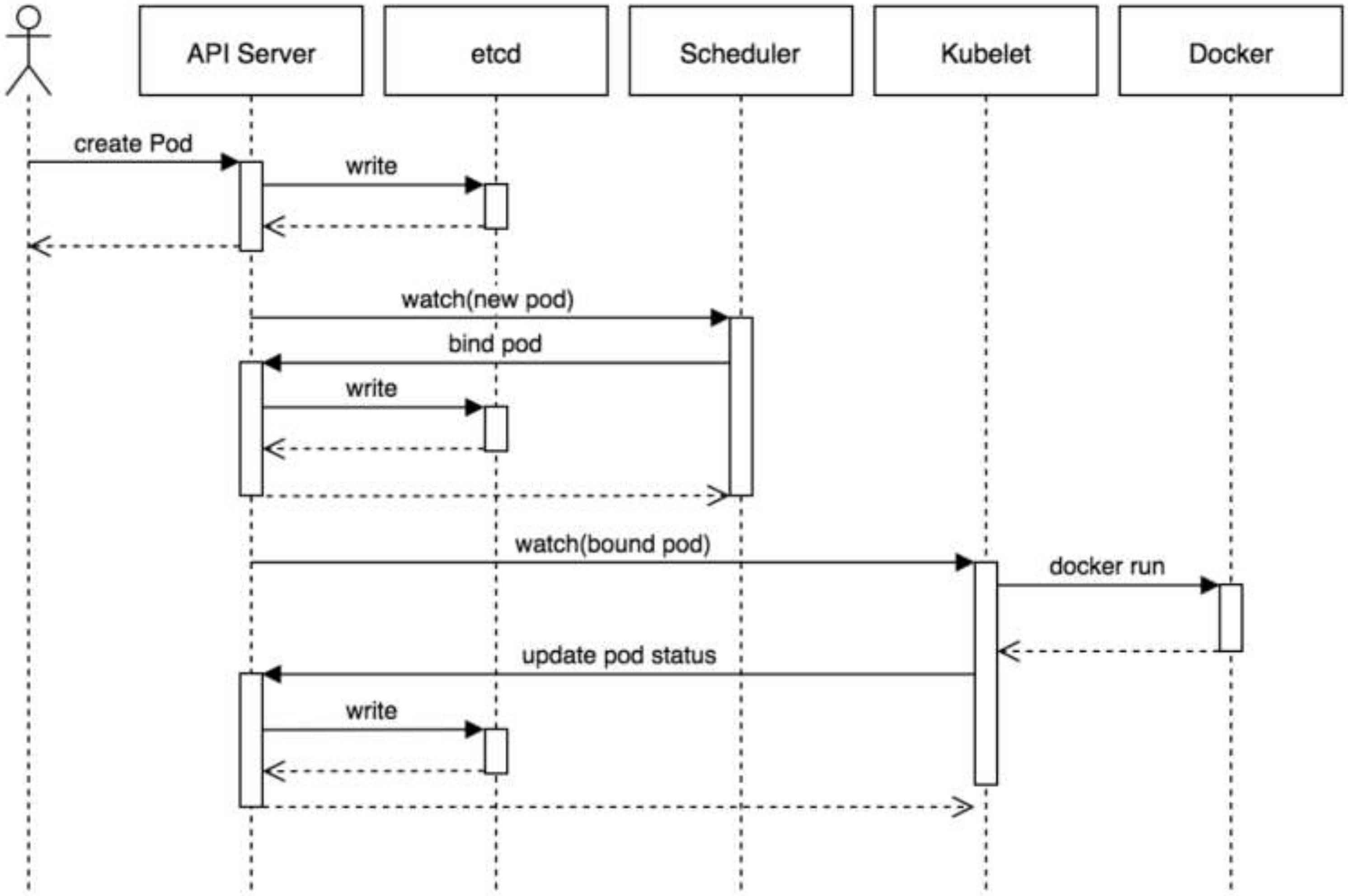
```
$ kubectl get jobs
NAME                        DESIRED   SUCCESSFUL   AGE
cronjob-example-1519053240   4         4            2m
cronjob-example-1519053300   4         4            1m
cronjob-example-1519053360   4         4            26s
```

```
$ 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
```

Health checks

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-readiness-http
spec:
  containers:
  - name: liveness-readiness-http
    image: k8s.gcr.io/ liveness-readiness-http
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 10
      timeoutSeconds: 4
      failureThreshold: 5
   readinessProbe:
      httpGet:
        path: /healthz
        port: 8080
      initialDelaySeconds: 100
      periodSeconds: 10
      timeoutSeconds: 4
      failureThreshold: 2
```

- initialDelaySeconds: Number of seconds after the container has started before liveness or readiness probes are initiated.
- periodSeconds: How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
- timeoutSeconds: Number of seconds after which the probe times out. Defaults to 1 second. Minimum value is 1.
- successThreshold: Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
- failureThreshold: When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up. Giving up in case of liveness probe means restarting the Pod. In case of readiness probe the Pod will be marked Unready. Defaults to 3. Minimum value is 1.


API Server
etcd
Scheduler
Kubelet
Docker
create Pod
write
watch(new pod)
bind pod
write
watch(bound pod)
docker run
update pod status
write

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**.

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 by eliminating the need for the cluster admin to pre-provision a PV

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

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
- persistentVolumeClaim
- 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 it’s 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 it’s 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 it’s 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.

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

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

```yaml
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

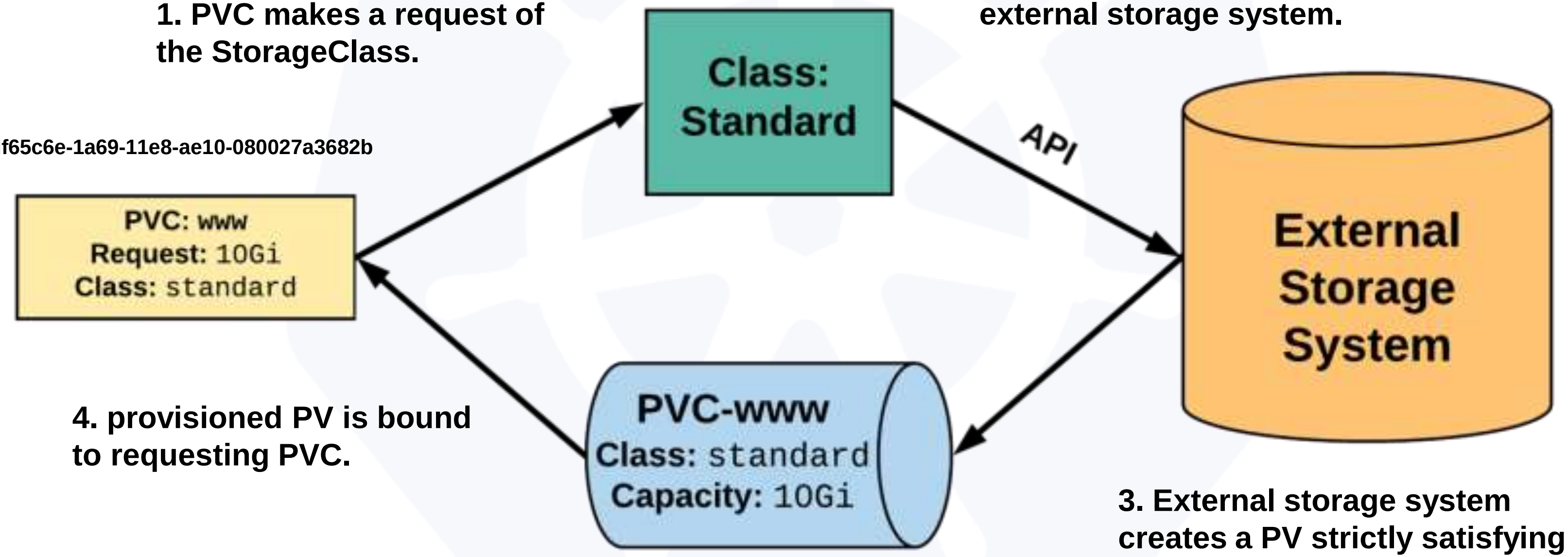
An error has been encountered.

StorageClass

1. PVC makes a request of the StorageClass.

2. StorageClass provisions request through API with external storage system.

uid: 9df65c6e-1a69-11e8-ae10-080027a3682b



PVC: www
Request: 10Gi
Class: standard

Class:
Standard

API

External
Storage
System

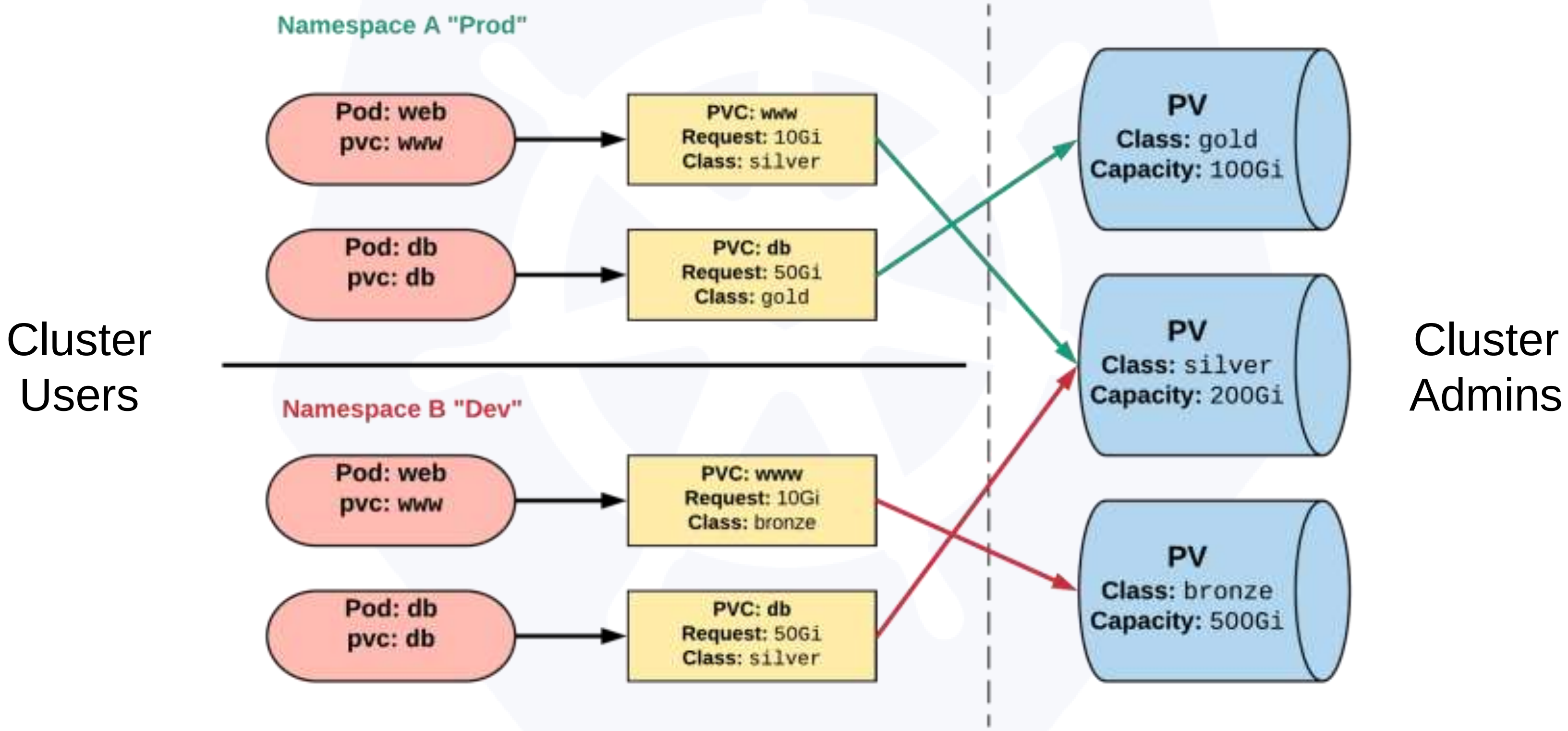
PVC-www
Class: standard
Capacity: 10Gi

4. provisioned PV is bound to requesting PVC.

3. External storage system creates a PV strictly satisfying the PVC request.

pv: pvc-9df65c6e-1a69-11e8-ae10-080027a3682b

Persistent Volumes and Claims



Working with Volumes

Configuration

- ConfigMap
- Secret

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.

```yaml
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 dir-example --from-file=cm/
configmap "dir-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 dir-example --from-file=cm/
configmap "dir-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 dir-example --from-file=cm/
configmap "dir-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 dir-example --from-file=cm/
configmap "dir-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
```

Secret

- Functionally identical to a ConfigMap.
- Stored as **base64 encoded content.**
- Encrypted at rest within etcd (**if configured!**).
- Stored on each worker node in tmpfs directory.
- Ideal for username/passwords, certificates or other sensitive information that should not be stored in a container.

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==
```

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!

```yaml
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

```yaml
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
```

```yaml
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

```yaml
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
```

```yaml
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
```

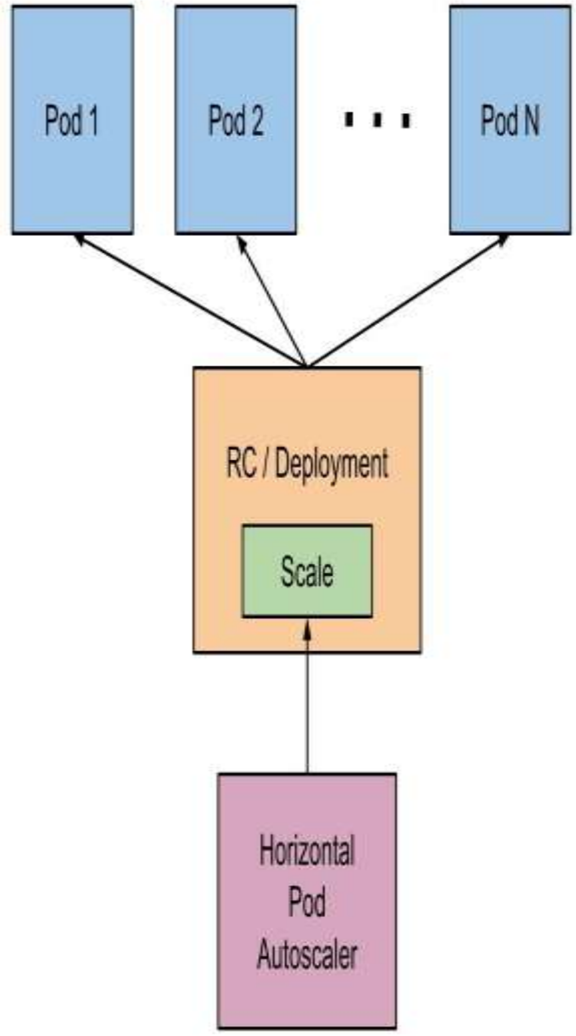
Metrics and Monitoring

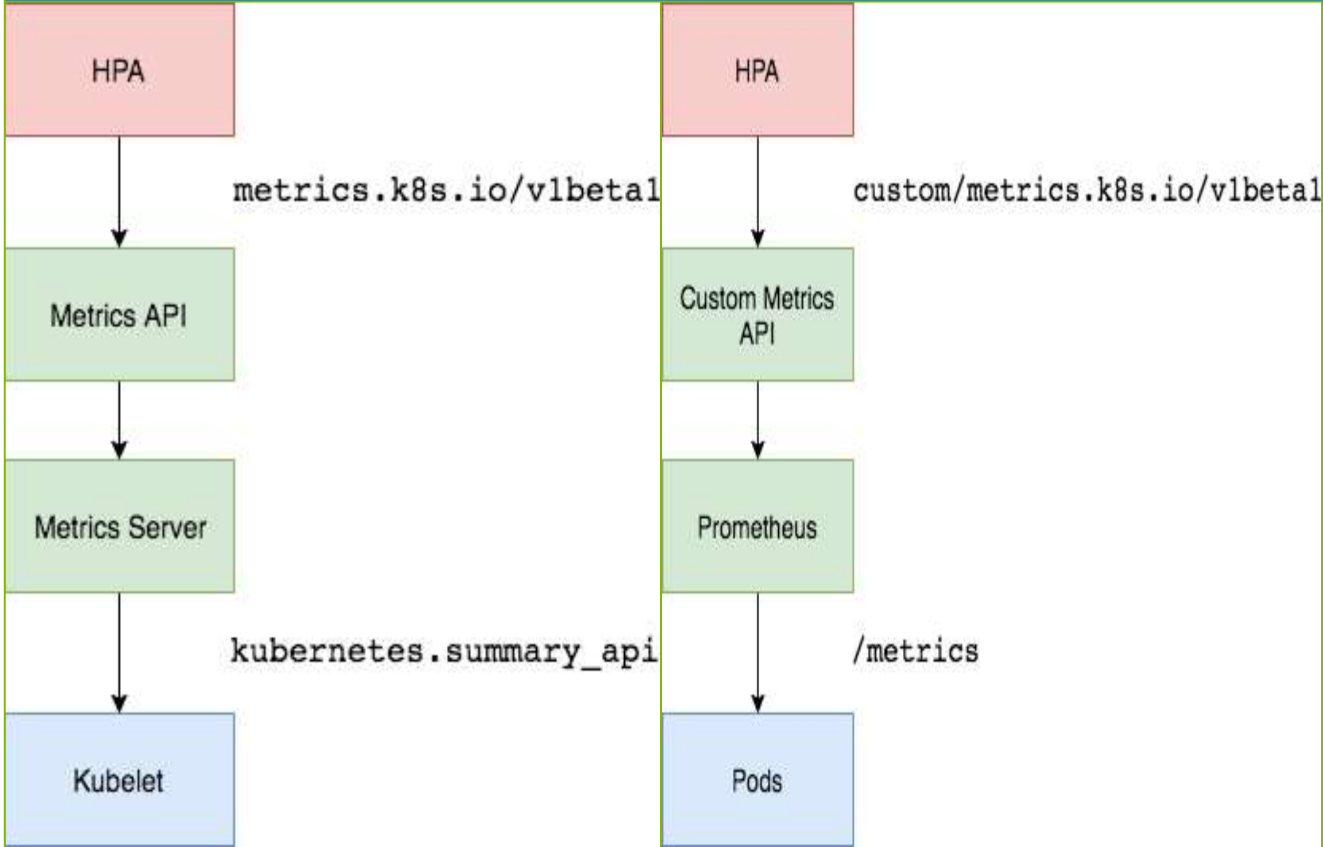
- Metrics server
- HPA (horizontal pod autoscaler)
- Prometheus
- Grafana (dashboards)
- Fluentd (log shipping)

Metrics API Server

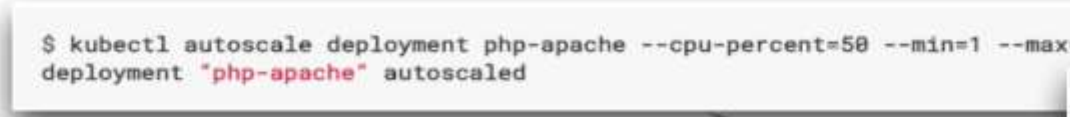
- Metric server collects metrics such as **CPU** and **Memory** by each pod and node from the Summary API, exposed by Kubelet on each node.
- Metrics Server registered in the main API server through Kubernetes aggregator, which was introduced in Kubernetes 1.7

HPA

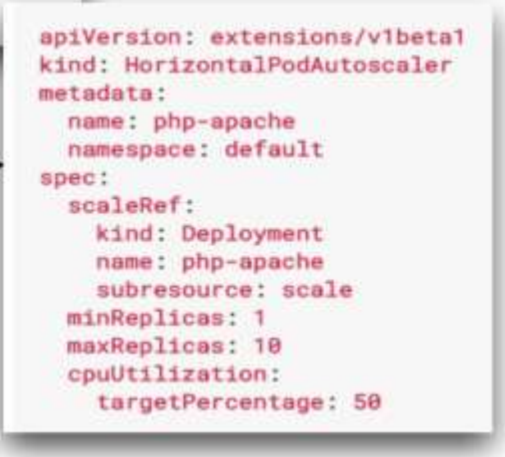

Pod 1
Pod 2
. . .
Pod N
RC / Deployment
Scale
Horizontal
Pod
Autoscaler


HPA
metrics.k8s.io/v1beta1
Metrics API
Metrics Server
kubernetes.summary_api
Kubelet
HPA
custom/metrics.k8s.io/v1beta1
Custom Metrics
API
Prometheus
/metrics
Pods

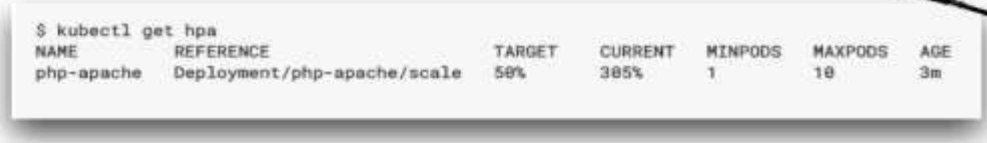
Horizontal Pod Autoscaling



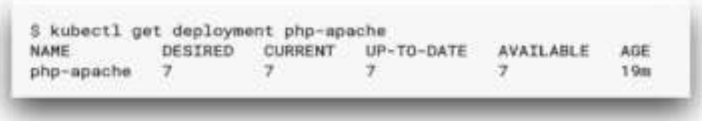
```
$ kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
deployment "php-apache" autoscaled
```



```
apiVersion: extensions/v1beta1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
  namespace: default
spec:
  scaleRef:
    kind: Deployment
    name: php-apache
    subresource: scale
  minReplicas: 1
  maxReplicas: 10
  cpuUtilization:
    targetPercentage: 50
```

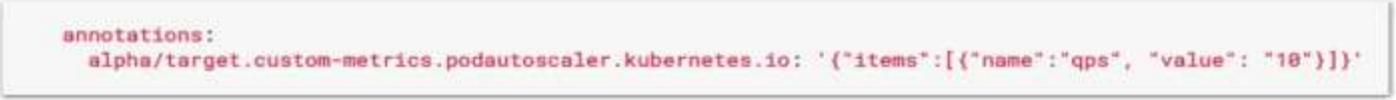


```
$ kubectl get hpa
NAME         REFERENCE                        TARGET    CURRENT   MINPODS   MAXPODS   AGE
php-apache   Deployment/php-apache/scale      50%       305%      1         10        3m
```

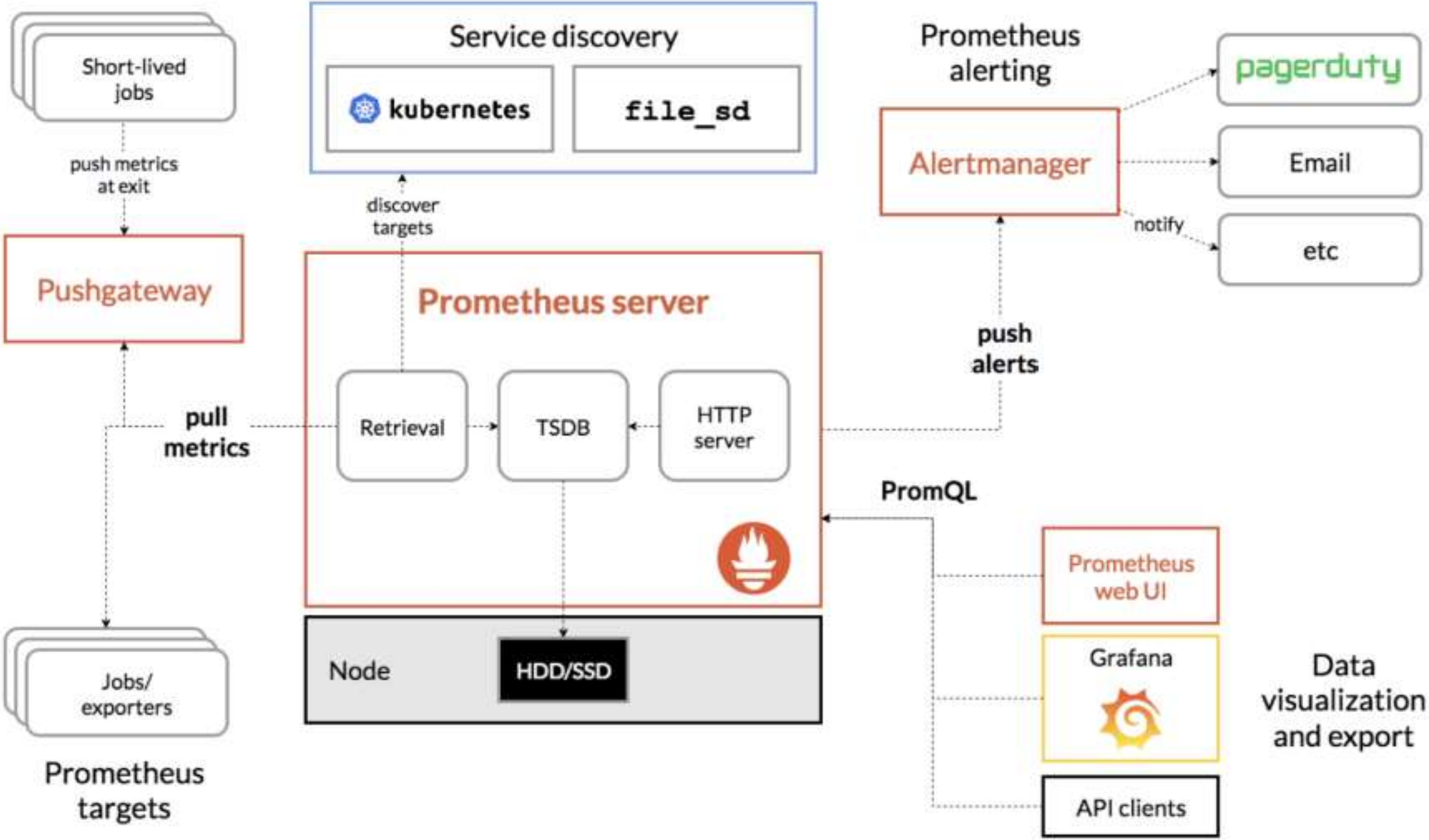


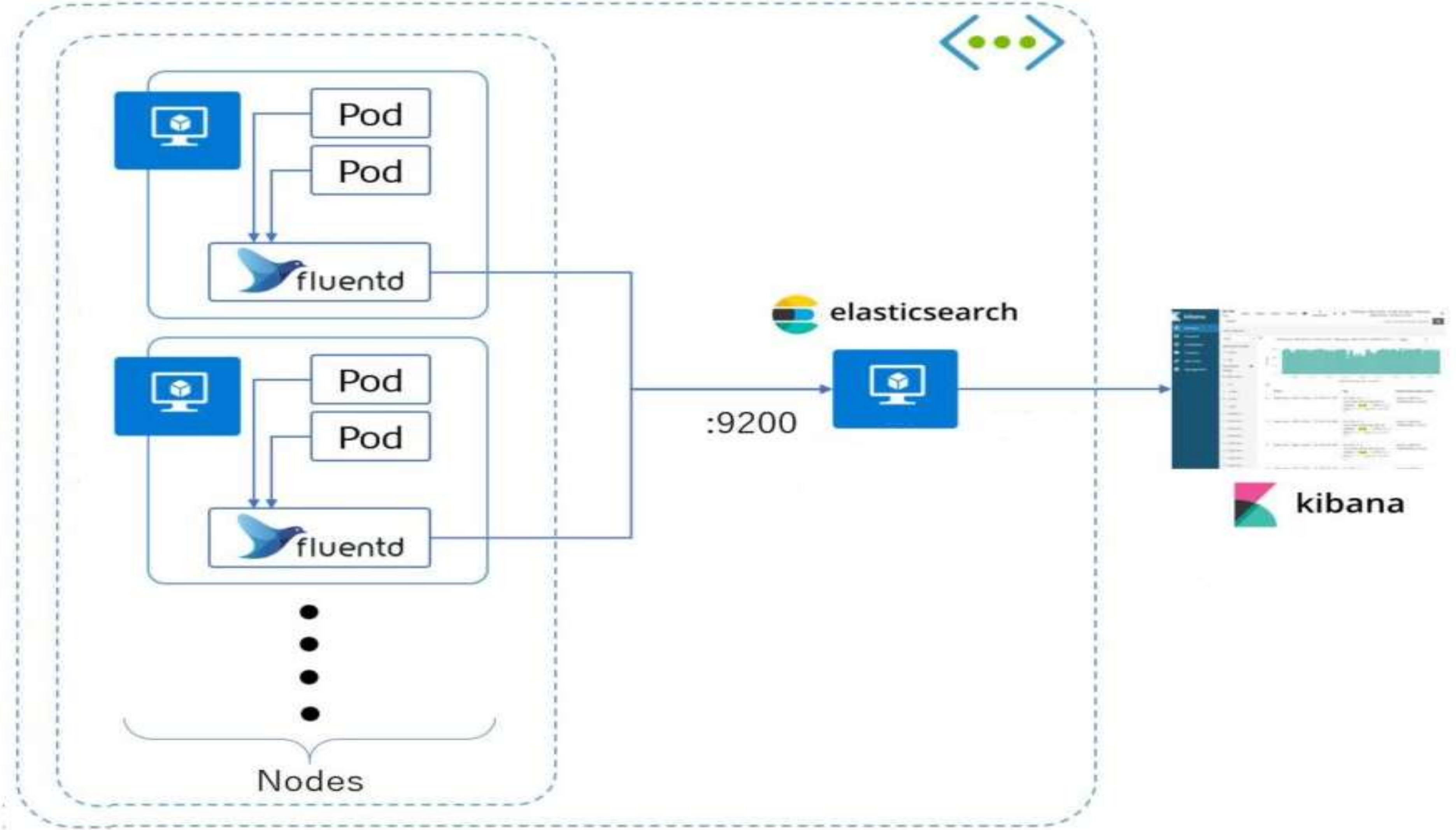
```
$ kubectl get deployment php-apache
NAME         DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
php-apache   7         7         7            7           19m
```

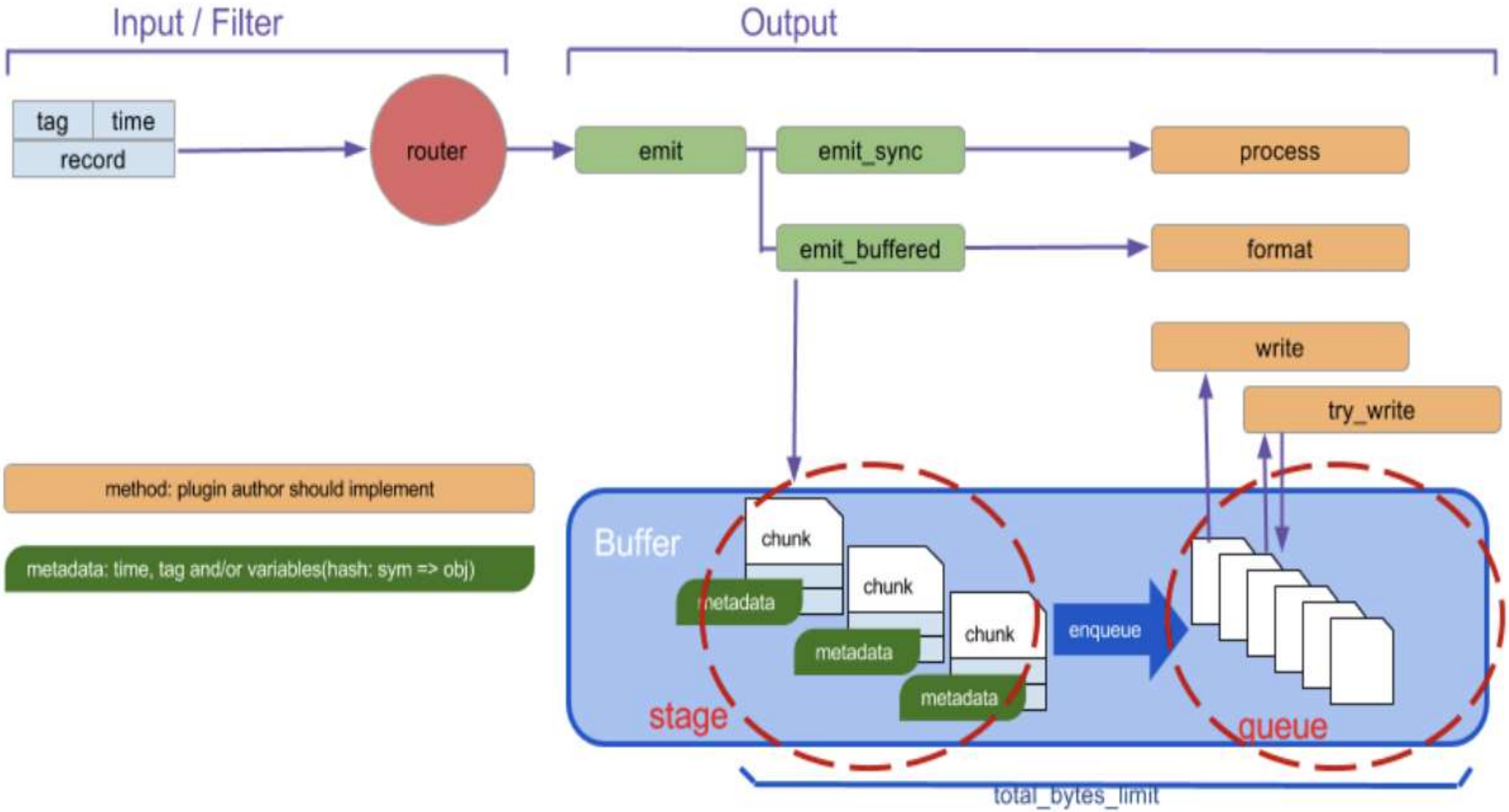
- Tips
  - Scale out/in
  - TriggeredScaleUp (GCE, AWS, will add more)
  - Support for **custom metrics**



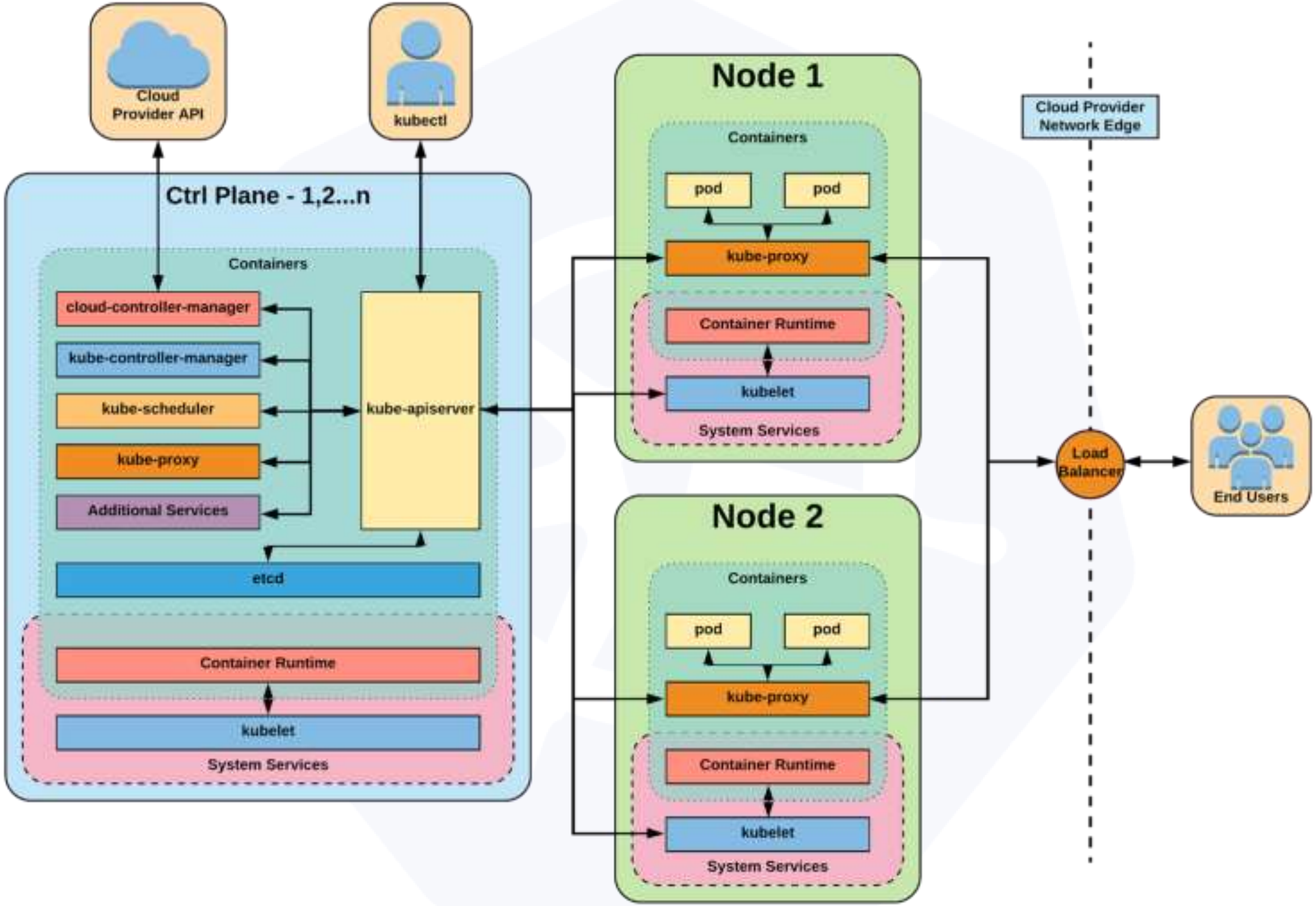
```
annotations:
  alpha/target.custom-metrics.podautoscaler.kubernetes.io: '{"items":[{"name":"qps", "value": "10"}]}'
```


Short-lived jobs
push metrics at exit
Pushgateway
pull metrics
Jobs/ exporters
Prometheus targets
Service discovery
kubernetes
file_sd
discover targets
Prometheus server
Retrieval
TSDB
HTTP server
Node
HDD/SSD
Prometheus alerting
Alertmanager
pagerduty
Email
etc
notify
push alerts
PromQL
Prometheus web UI
Grafana
API clients
Data visualization and export


Pod
Pod
fluentd
Pod
Pod
fluentd
Nodes
elasticsearch
:9200
kibana


Input / Filter
Output
tag
time
record
router
emit
emit_sync
emit_buffered
process
format
write
try_write
method: plugin author should implement
metadata: time, tag and/or variables(hash: sym => obj)
Buffer
chunk
metadata
chunk
metadata
chunk
metadata
enqueue
stage
queue
total_bytes_limit

Summary


Cloud Provider API
kubectl
Ctrl Plane - 1,2...n
Containers
cloud-controller-manager
kube-controller-manager
kube-scheduler
kube-proxy
Additional Services
kube-apiserver
etcd
Container Runtime
kubelet
System Services
Node 1
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Node 2
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

Връзки към Документация

- **Официална документация за K8s**
  https://kubernetes.io/docs/home/
- **Официална документация за Kubespray -**
  https://github.com/kubernetes-sigs/kubespray
- **Официална страница на CloudNative**
- **https://www.cncf.io/**
- **Официален Youtube канал на k8s**
  https://www.youtube.com/c/KubernetesCommunity
- **Официален Youtube канал на k8s**
  https://www.youtube.com/c/cloudnativefdn
-

Въпроси?

Изплозвани са материали и картинки от - Joe Beda