O'REILLY®

Certified Kubernetes Administrator (CKA) Crash Course

Kubernetes 1.19+ Edition

About the trainer

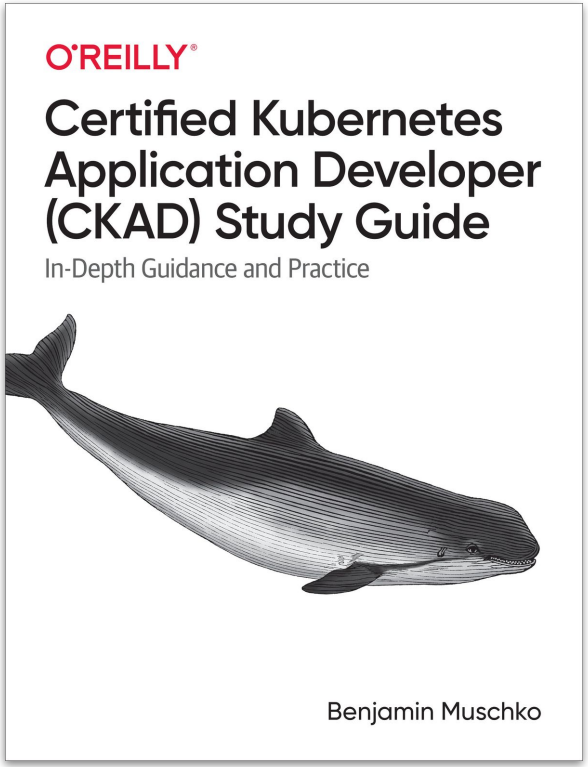
bmuschko

bmuschko

bmuschko.com

AUTOMATED ASCENT

automatedascent.com



Companion study guide with practice questions

Released in February 2021

Online access on O'Reilly learning platform:

https://learning.oreilly.com/library/view/certified-kubernetes-application/9781492083726/

Exam Details and Resources

Objectives, Environment, Time Management

Exam Objectives

“Perform typical responsibilities of a Kubernetes administrator.”

The certification program allows users to demonstrate their competence in a hands-on, command-line environment.

https://www.cncf.io/certification/cka/

Exam Domains & Weights

Domain	Weight
Cluster Architecture, Installation & Configuration	25%
Workloads & Scheduling	15%
Services & Networking	20%
Storage	10%
Troubleshooting	30%

The Curriculum

25% - Cluster Architecture, Installation & Configuration

- Manage role based access control (RBAC)
- Use Kubeadm to install a basic cluster
- Manage a highly-available Kubernetes cluster
- Provision underlying infrastructure to deploy a Kubernetes cluster
- Perform a version upgrade on a Kubernetes cluster using Kubeadm
- Implement etcd backup and restore

20% - Services & Networking

- Understand host networking configuration on the cluster nodes
- Understand connectivity between Pods
- Understand ClusterIP, NodePort, LoadBalancer service types and endpoints
- Know how to use Ingress controllers and Ingress resources
- Know how to configure and use CoreDNS
- Choose an appropriate container network interface plugin

30% - Troubleshooting

- Evaluate cluster and node logging
- Understand how to monitor applications
- Manage container stdout & stderr logs
- Troubleshoot application failure
- Troubleshoot cluster component failure
- Troubleshoot networking

15% - Workloads & Scheduling

- Understand deployments and how to perform rolling update and rollbacks
- Use ConfigMaps and Secrets to configure applications
- Know how to scale applications
- Understand the primitives used to create robust, self-healing, application deployments
- Understand how resource limits can affect Pod scheduling
- Awareness of manifest management and common templating tools

10% - Storage

- Understand storage classes, persistent volumes
- Understand volume mode, access modes and reclaim policies for volumes
- Understand persistent volume claims primitive
- Know how to configure applications with persistent storage

Candidate Skills

kubernetes

Architecture & Concepts

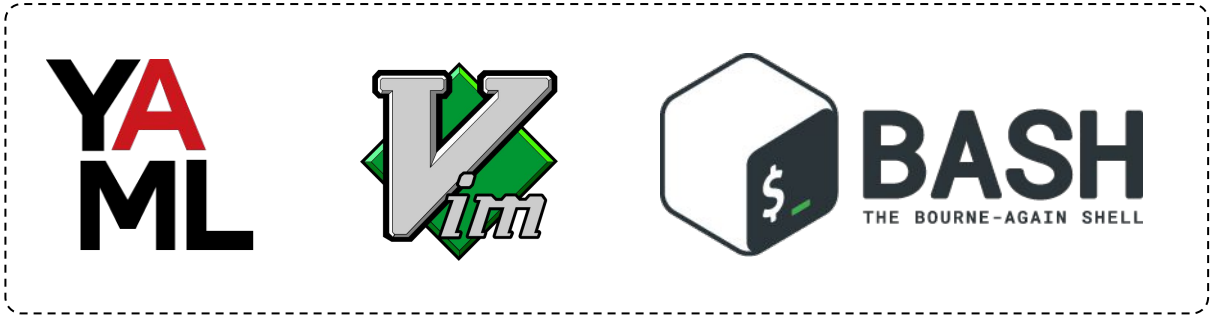
`kubectl`

Running Commands

Underlying Concepts

Exam Environment

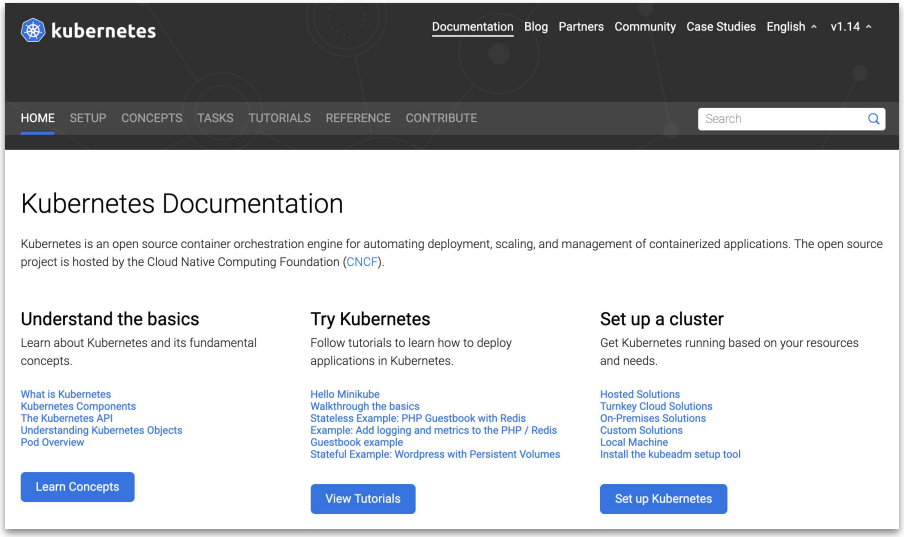
Online and proctored exam



The trinity of tooling you need to be familiar with

Using Documentation

Know where and how to find relevant documentation



https://kubernetes.io/docs

Getting Help on a Command

Render subcommands and options with `--help`

```
$ kubectl create --help
Create a resource from a file or from stdin.

JSON and YAML formats are accepted.

...
Available Commands:
...
  configmap           Create a configmap from a local file, directory or literal
value
  deployment          Create a deployment with the specified name.
...

Options:
...
```

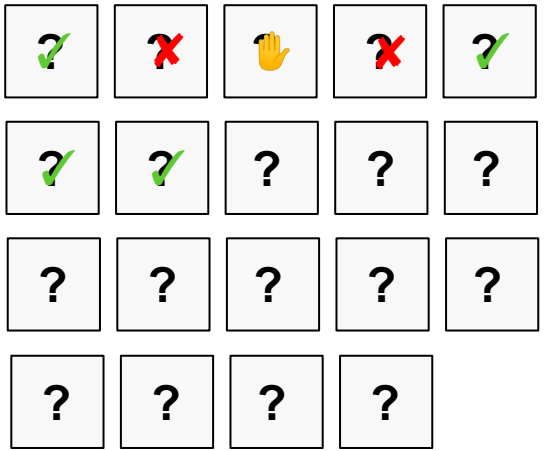
Zeroing in on Command Details

Drill into object details with the `explain` command

```
$ kubectl explain pods.spec
KIND:     Pod
VERSION:  v1

RESOURCE: spec <Object>

DESCRIPTION:
...

FIELDS:
...
```

Time Management

# of problems in 2 hours, use your time wisely!



Using an Alias for kubectl

Your first action at the beginning of the exam


```
$ alias k=kubectl
$ k version
...
```

Setting a Context & Namespace

Questions will ask you to run a command on a specific cluster - Make sure to execute it!

```
$ kubectl config set-context <context-of-question>↵
--namespace=<namespace-of-question>
```

Internalize Resource Short Names

Some API resources provide a shortcut

```
$ kubectl get ns
```

Usage of `ns` instead of `namespaces`

```
$ kubectl describe pvc claim
```

Usage of `pvc` instead of `persistentvolumeclaim`

Deleting Kubernetes Objects

Don’t wait for a graceful deletion of objects...

```
$ kubectl delete pod nginx --grace-period=0 --force
```

Understand and Practice bash

Practice relevant syntax and language constructs

```
$ if [ ! -d ~/tmp ]; then mkdir -p ~/tmp; fi; while true;↲
do echo $(date) >> ~/tmp/date.txt; sleep 5; done;
```

Finding Object Information

Filter configuration with context from a set of objects

```
$ kubectl describe pods | grep -C 10 "author=John Doe"
$ kubectl get pods -o yaml | grep -C 5 labels:
```

`grep` is your friend!

How to Prepare

Practice, practice, practice!

The key to cracking the exam

Q & A

BREAK

Cluster Architecture, Installation & Configuration

RBAC, Kubeadm, HA, etcd Backup and Restore

RBAC High-Level Overview

Three key elements for understanding concept

Subject

Groups
Users
Service-Accounts

API Resources

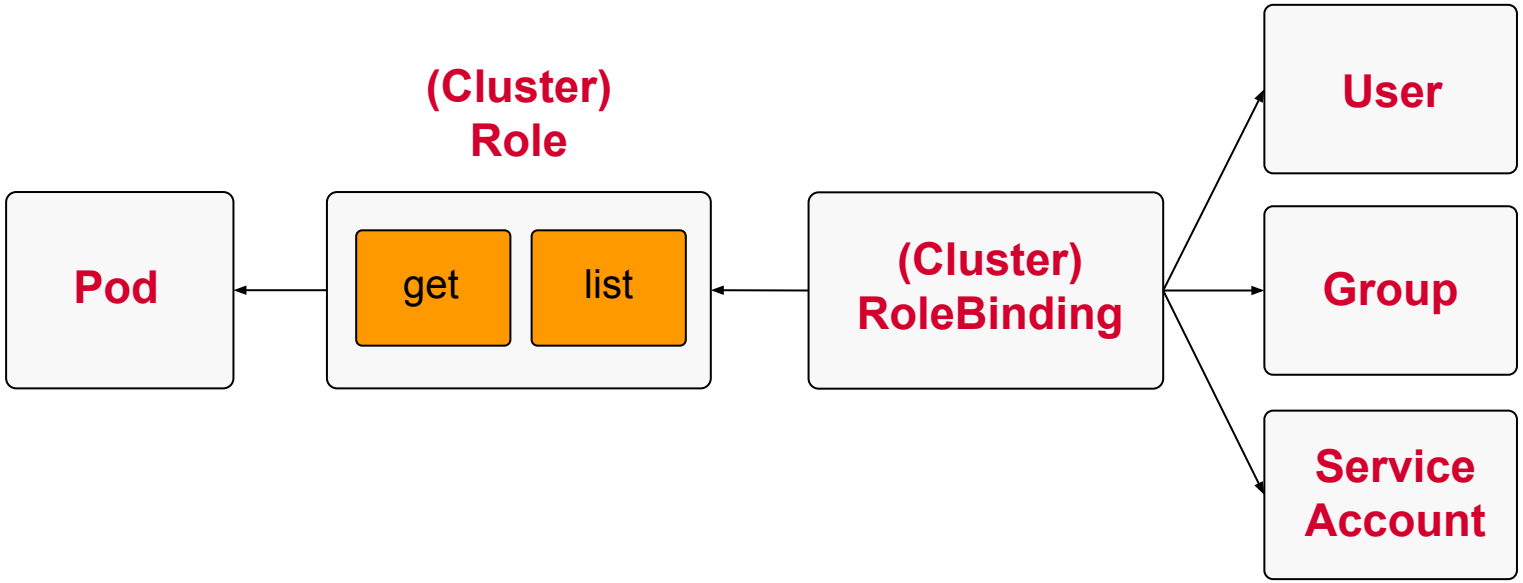
ConfigMap
Pod
Deployment
Node
...

Operations (Verbs)

create
list
watch
delete
...

Involved RBAC Primitives

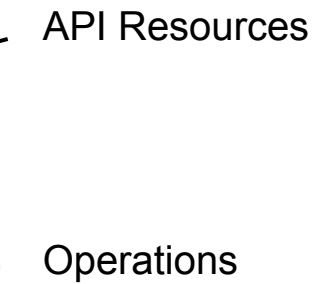
Restrict access to API resources based on user roles



Defining a Role

Connects API resources and verbs

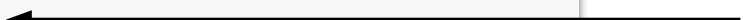
```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```



Defining a RoleBinding

Grants the permissions defined in a role to subject

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
subjects:
- kind: User
  name: jane
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```



ClusterRole + ClusterRoleBinding

Same as Role and RoleBinding but on cluster-level

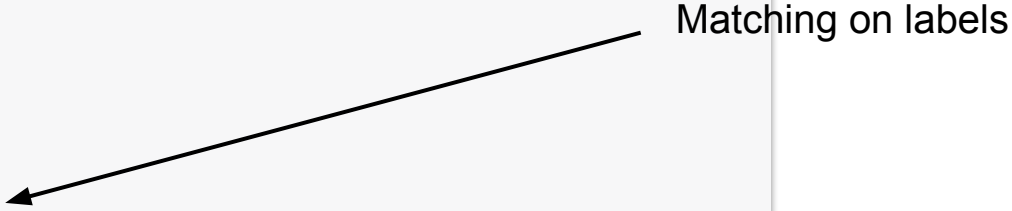
- ClusterRole
  - Can grant same permissions as Role.
  - Can grant access to nodes, non-resource endpoints and resources across all namespaces.

- ClusterRoleBinding
  - Can grant same access as RoleBinding.
  - Bind a ClusterRole to all the namespaces in the cluster.

Aggregated ClusterRoles

Combine multiple ClusterRoles into one

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring
aggregationRule:
  clusterRoleSelectors:
  - matchLabels:
      rbac.example.com/aggregate-to-monitoring: "true"
rules: []
```



EXERCISE

Regulating Access
to API Resources
with RBAC

What is Kubeadm?

Tool for creating and managing Kubernetes clusters

- Needs to be installed separately from other tools like `kubectl`.
- Deals with cluster bootstrapping but not provisioning.
- Representative use cases
  - Bootstrap a control-plane node.
  - Bootstrap worker nodes and join them to the cluster.
  - Upgrade a cluster to a newer version.

Installing a Cluster

Start with master, join nodes

- Initialize the control plane on master node using `kubeadm init`.
- Install a Pod network add-on.
- Join worker nodes using `kubeadm join`.

Detailed installation instructions

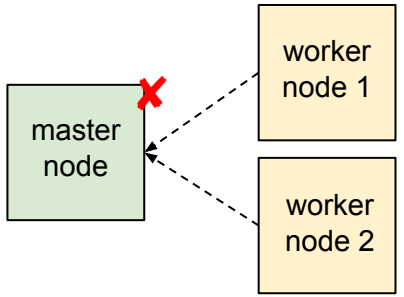
DEMO

Creating a Cluster
with Kubeadm

Single-Master Cluster Setup

Losing the master node causes issues

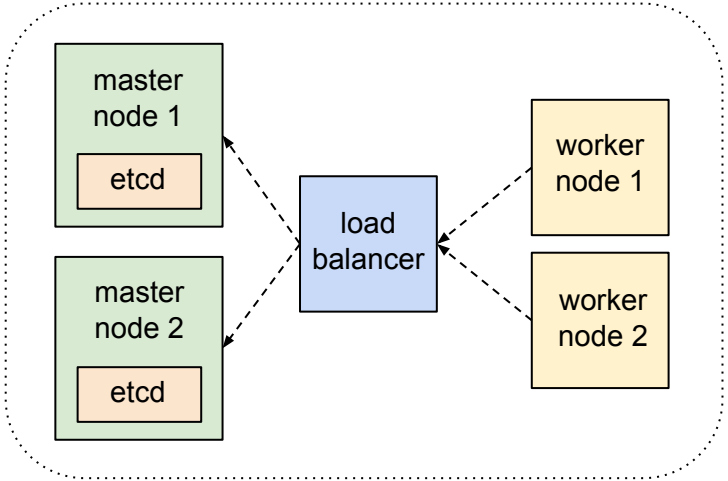
- A ReplicaSet cannot recreate failing Pod as the worker node can’t talk back to scheduler on master node.
- Cluster cannot be accessed externally as API server is not available anymore.



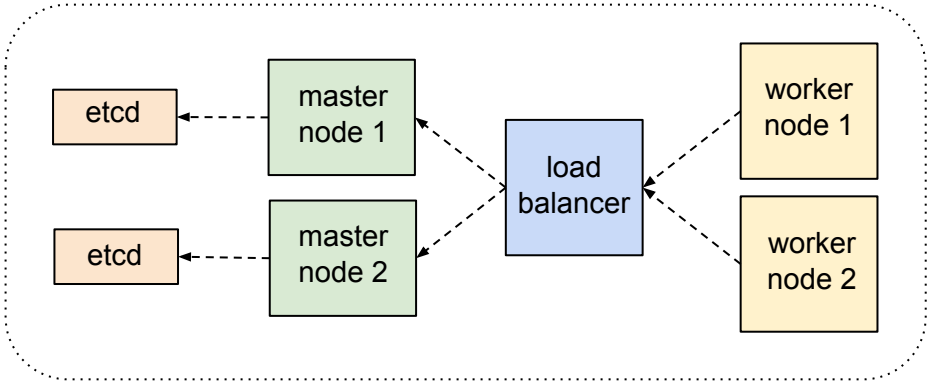
High-Availability Cluster Setup

Two configuration options available

Stacked etcd topology



External etcd topology



Detailed installation instructions

Upgrading a Cluster Version

Upgrading should be done in version increments

- Determine which version to upgrade to.
- Upgrade control plane nodes.
- Upgrade worker nodes.
- Verify the status of the cluster.

Detailed installation instructions

DEMO

Upgrading a
Cluster Version with
Kubeadm

Backing up & Restoring etcd

Get etcdctl utility if it's not already present

- Create backup with `etcdctl snapshot save` command. The options `--cert`, `--cacert` and `--key` are mandatory.
- Restore backup with `etcdctl snapshot restore` command. The option `--data-dir` is mandatory. Modify the `hostPath.path` in `/etc/kubernetes/manifests/etcd.yaml` to point to directory.

Detailed installation instructions

DEMO

Backing Up and
Restoring etcd

Q & A

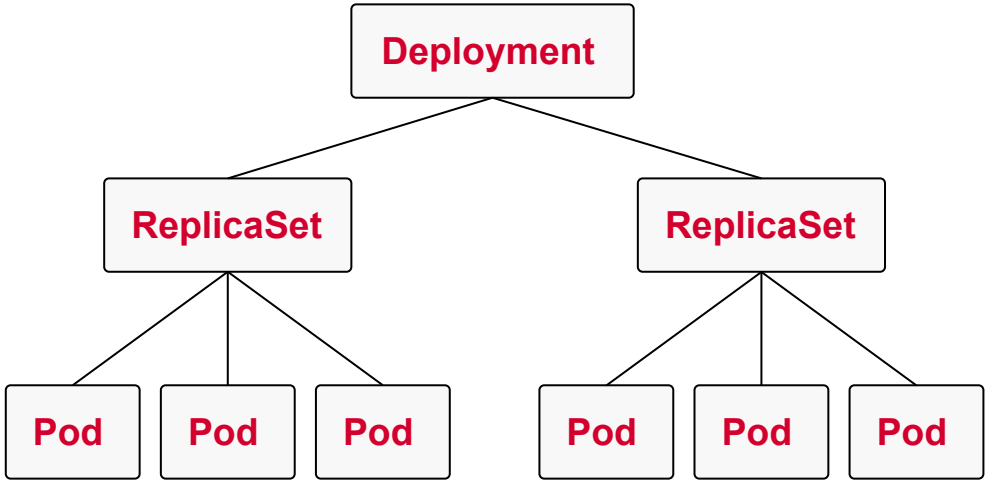
BREAK

Workloads & Scheduling

Deployments, ConfigMaps & Secrets, Health Probes, Pod Resource Limits, Node Affinity, Taints & Tolerations

Understanding Deployments

Scaling and replication features for a set of Pods



Creating a Deployment

The `create` command supports `replicas` option with 1.19+

```
$ kubectl create deployment my-deploy --image=nginx --replicas=3↵
  --dry-run=client -o yaml > deploy.yaml
$ vim deploy.yaml
$ kubectl create -f deploy.yaml
deployment.apps/my-deploy created
```

Creating a Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: my-deploy
  name: my-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-deploy
  template:
    metadata:
      labels:
        app: my-deploy
    spec:
      containers:
      - image: nginx
        name: nginx
```

The number of Pods running a specific set of containers

Selects the Pods for this deployment

The labels of the Pods

Inspecting Deployment State

Indicator between desired state and actual state

```
$ kubectl get deployments
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-deploy     3       3            3           25m
```

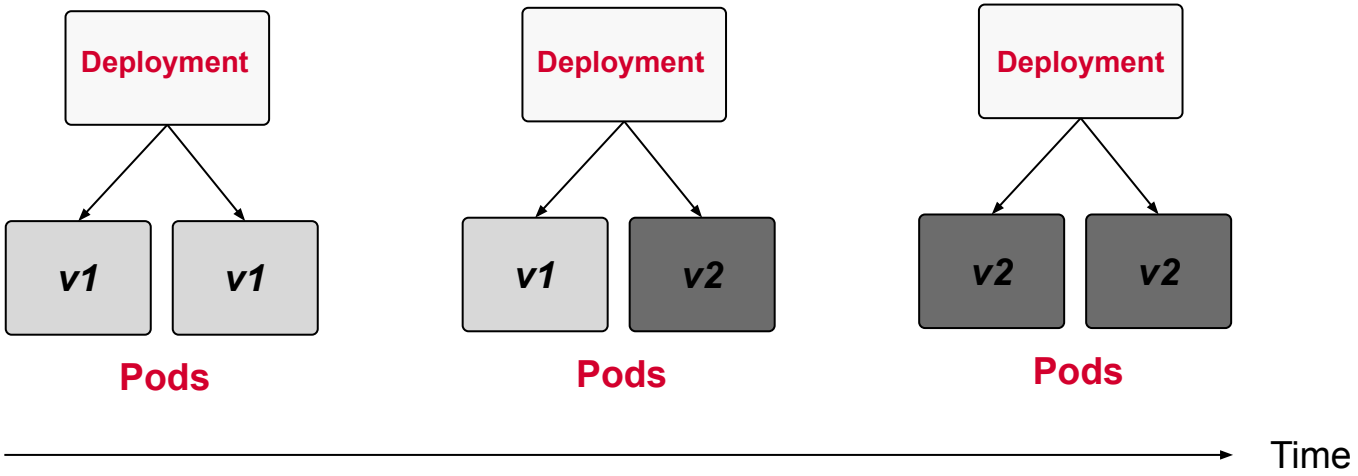
Underlying Replication Feature

Automatically created by Deployment, not meant to be modified

```
$ kubectl get replicasets
NAME                    DESIRED   CURRENT   READY   AGE
my-deploy-7786f96d67    3         3         3       6h

$ kubectl describe deploy my-deploy
...
OldReplicaSets:  <none>
NewReplicaSet:   my-deploy-7786f96d67 (3/3 replicas created)
...

$ kubectl describe replicasets my-deploy-7786f96d67
...
Controlled By:  Deployment/my-deploy
...
```

Rolling Updates

“Look ma, shiny new features. Let’s deploy them to production!”



Rollout Revision Log

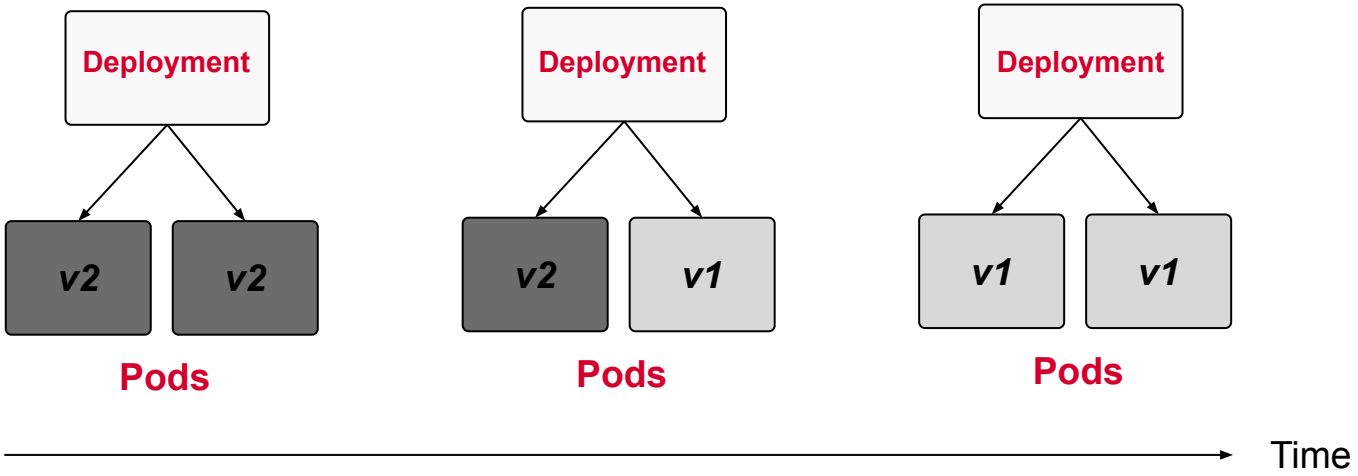
```
# Check initial deployment revisions
$ kubectl rollout history deployments my-deploy
deployment.extensions/my-deploy
REVISION  CHANGE-CAUSE
1         <none>

# Make a change to the deployment
$ kubectl edit deployments my-deploy

# Revision history indicates changed version
$ kubectl rollout history deployments my-deploy
deployment.extensions/my-deploy
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

Rendering Revision Details

```
$ kubectl rollout history deployments my-deploy --revision=2
deployment.extensions/my-deploy with revision #2
Pod Template:
  Labels:    app=my-deploy
    pod-template-hash=1365642048
  Containers:
   nginx:
    Image:   nginx:latest
    Port:    <none>
    Host Port:   <none>
    Environment: <none>
    Mounts:  <none>
  Volumes:   <none>
```

Rolling Back

“Bug in the application. Let’s revert to the previous version!”

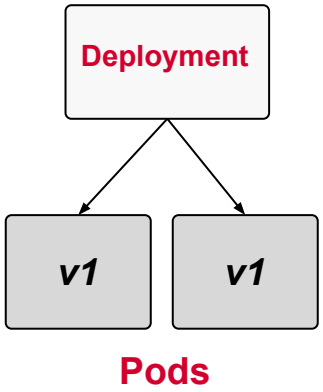


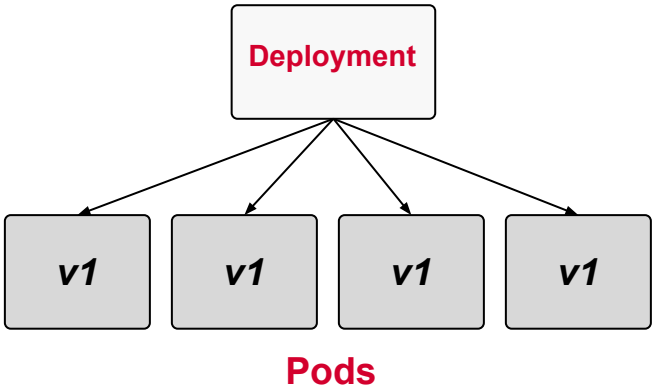
Rolling Back to a Revision

```
# Roll back to previous revision
$ kubectl rollout undo deployments my-deploy
deployment.extensions/my-deploy

# Check rollout status
$ kubectl rollout status deployments my-deploy
deployment "my-deploy" successfully rolled out

# Revision history indicates changed version
$ kubectl rollout history deployments my-deploy
deployment.extensions/my-deploy
REVISION  CHANGE-CAUSE
2         <none>
3         <none>
```

Manually Scaling a Deployment

“Load is increasing. We need to scale up the application.”



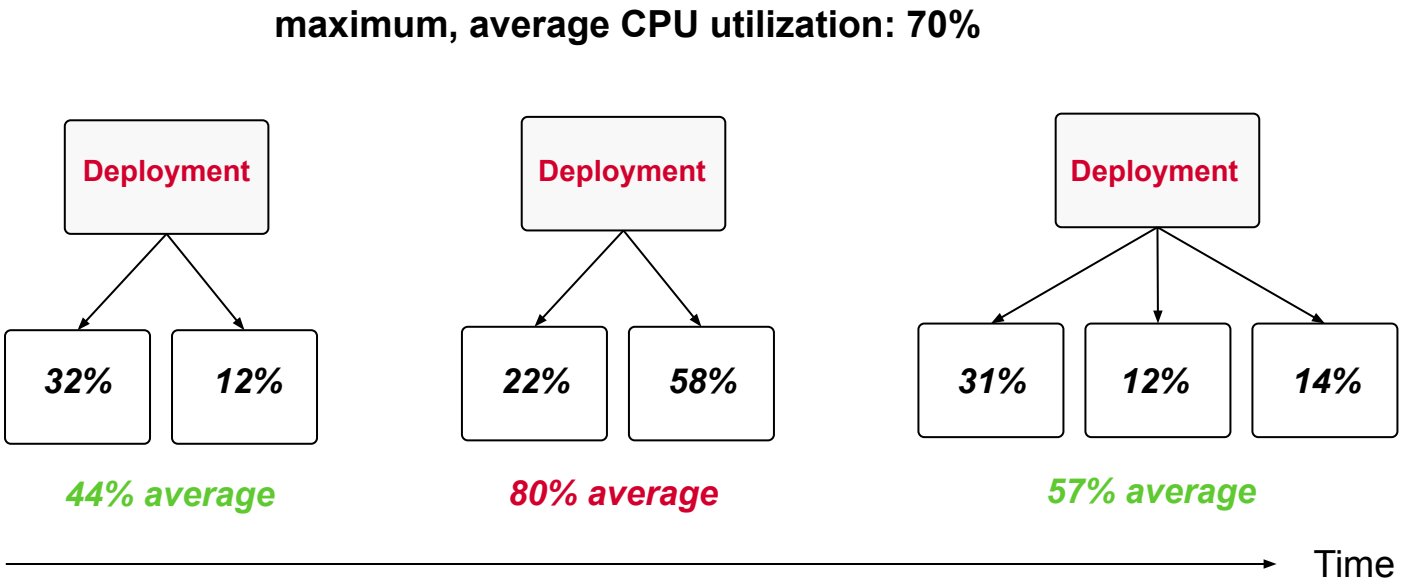


Providing a Specific # of Replicas

```
# Check current deployment replicas
$ kubectl get deployments my-deploy
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
my-deploy    2       2            2           9h

# Scaling from 2 to 4 replicas
$ kubectl scale deployment my-deploy --replicas=4
deployment.extensions/my-deploy scaled

# Check the changed deployment replicas
$ kubectl get deployment my-deploy
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
my-deploy    4       4            4           9h
```

Autoscaling a Deployment

“Don’t make me think. Autoscale based on CPU utilization.”



Create Horizontal Pod Autoscaler

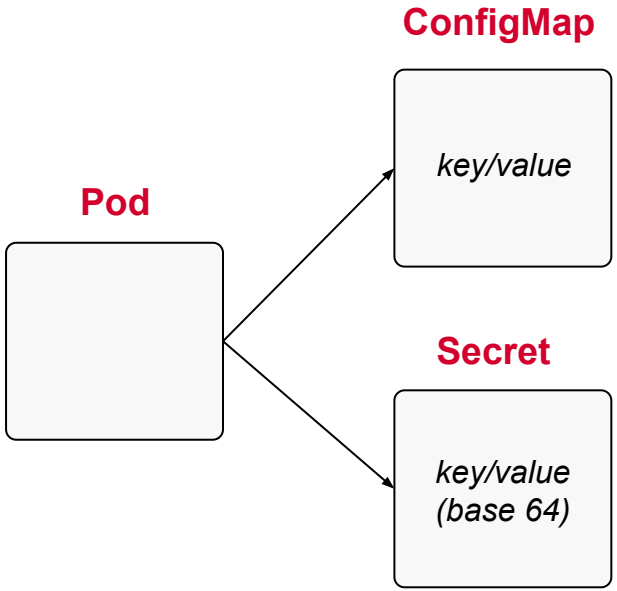
```
# Maintain average CPU utilization across all Pods of 70%
$ kubectl autoscale deployments my-deploy --cpu-percent=70 ↲
  --min=1 --max=10
horizontalpodautoscaler.autoscaling/my-deploy autoscaled

# Check the current status of autoscaler
$ kubectl get hpa my-deploy
NAME         REFERENCE               TARGETS          MINPODS ↲
MAXPODS   REPLICAS   AGE
my-deploy   Deployment/my-deploy   0%/70%           1 ↲
10        4          23s
```

EXERCISE

Performing Rolling Updates and Scaling a Deployment

Centralized Configuration Data

Injects runtime configuration through object references



Creating ConfigMaps (imperative)

Fast, easy and flexible, can point to different sources

```
# Literal values
$ kubectl create configmap db-config --from-literal=db=staging

# Single file with environment variables
$ kubectl create configmap db-config --from-env-file=config.env

# File or directory
$ kubectl create configmap db-config --from-file=config.txt
```

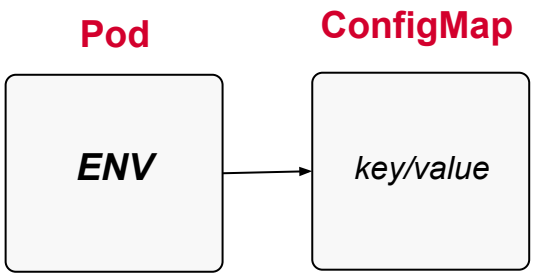
Creating ConfigMaps (declarative)

Definition of a ConfigMap is fairly short and on point

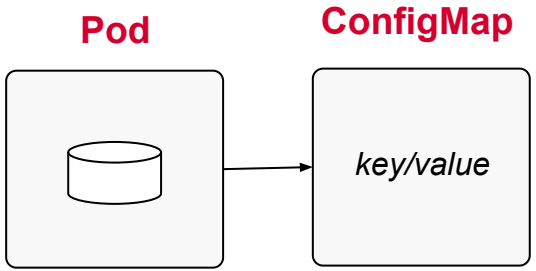
```
apiVersion: v1
data:
  db: staging
  username: jdoe
kind: ConfigMap
metadata:
  name: db-config
```

Mounting a ConfigMap

Two options for consuming data



Injected as environment variables



Mounted as volume

ConfigMap Env. Variables in Pod

Convenient if ConfigMap reflects the desired syntax

```
apiVersion: v1
kind: Pod
metadata:
  name: backend
spec:
  containers:
  - image: nginx
    name: backend
    envFrom:
      - configMapRef:
          name: db-config
```

```
$ kubectl exec -it nginx -- env
DB=staging
USERNAME=jdoe
...
```

ConfigMap in Pod as Volume

Each key becomes file in mounted directory

```
apiVersion: v1
kind: Pod
metadata:
  name: backend
spec:
  containers:
    - name: backend
      image: nginx
      volumeMounts:
      - name: config-volume
        mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        name: db-config
```

```
$ kubectl exec -it backend -- /bin/sh
# ls /etc/config
db
username
# cat /etc/config/db
staging
```

Creating Secrets (imperative)

Similar usage to creation of ConfigMap

```
# Literal values
$ kubectl create secret generic db-creds ↲
  --from-literal=pwd=s3cre!

# File containing environment variables
$ kubectl create secret generic db-creds ↲
  --from-env-file=secret.env

# SSH key file
$ kubectl create secret generic db-creds ↲
  --from-file=ssh-privatekey=~/.ssh/id_rsa
```

Creating Secrets (declarative)

Value has to be base64-encoded manually

```
$ echo -n 's3cre!' | base64
czNjcmUh
```

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  pwd: czNjcmUh
```

Secret in Pod as Volume

Value has to be base64-encoded manually

```
apiVersion: v1
kind: Pod
metadata:
  name: backend
spec:
  containers:
    - name: backend
      image: nginx
      volumeMounts:
      - name: secret-volume
        mountPath: /etc/secret
  volumes:
    - name: secret-volume
      secret:
        secretName: mysecret
```

```
$ kubectl exec -it backend -- /bin/sh
# ls /etc/secret
pwd
# cat /etc/secret/pwd
s3cre!
```

EXERCISE

Creating and
Mounting a
ConfigMap

Container Health

“How does Kubernetes know if a container is up and running?”

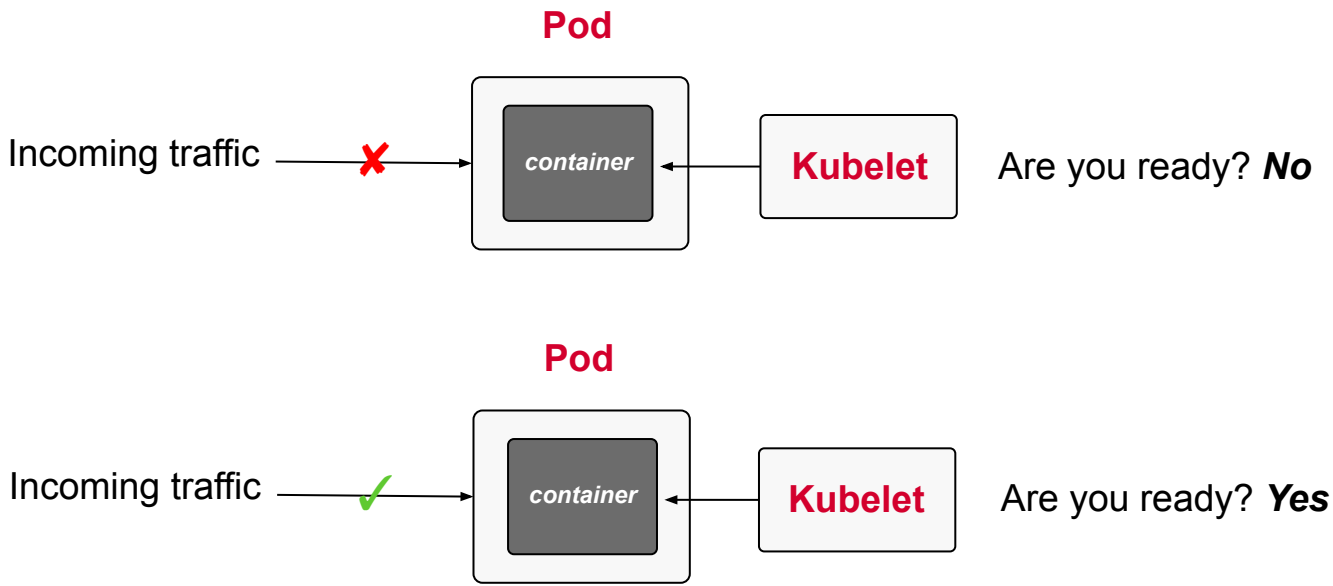
Probes can detect
and correct failures

Health Verification Methods

Method	Option	Description
Custom Command	`exec.command`	Executes a command inside of the container e.g. a cat command and checks its exit code. Kubernetes considers an zero exit code to be successful. A non-zero exit code indicates an error.
HTTP GET Request	`httpGet`	Sends an HTTP GET request to an endpoint exposed by the application. A HTTP response code in the range of 200 and 399 indicates success. Any other response code is regarded as an error.
TCP socket connection	`tcpSocket`	Tries to open a TCP socket connection to a port. If the connection could be established, the probing attempt was successful. The inability to connect is accounted for as an error.

Understanding Readiness Probes

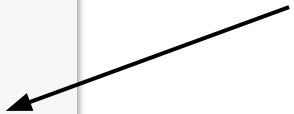
“Is application ready to serve requests?”



Defining a Readiness Probe

HTTP probes are very helpful for web applications

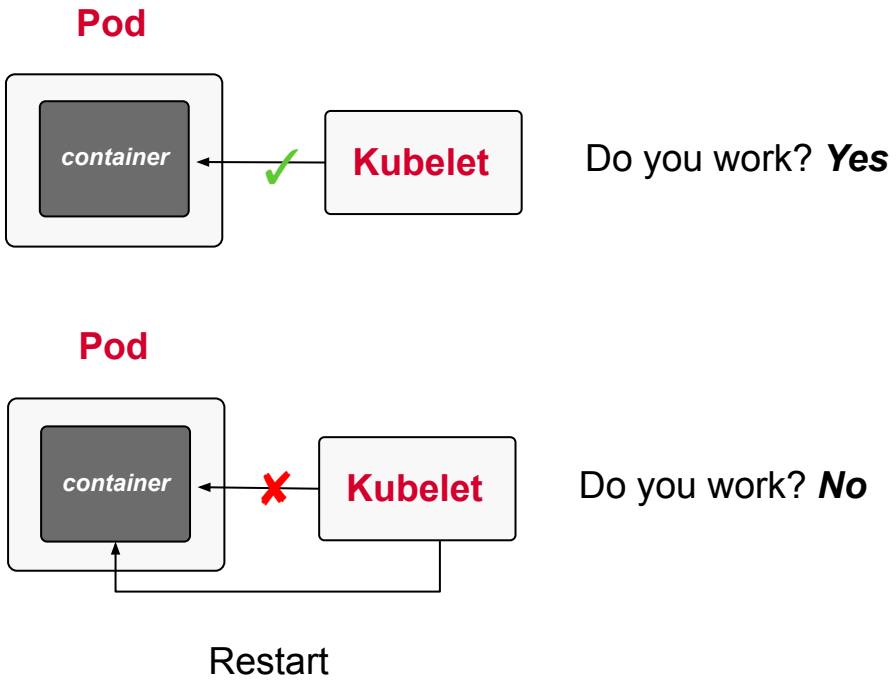
```
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  containers:
  - name: web-app
    image: eshop:4.6.3
    readinessProbe:
      httpGet:
        path: /
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 2
```



Successful if HTTP status code is between 200 and 399

Understanding Liveness Probes

“Does the application still function without errors?”



Defining a Liveness Probe

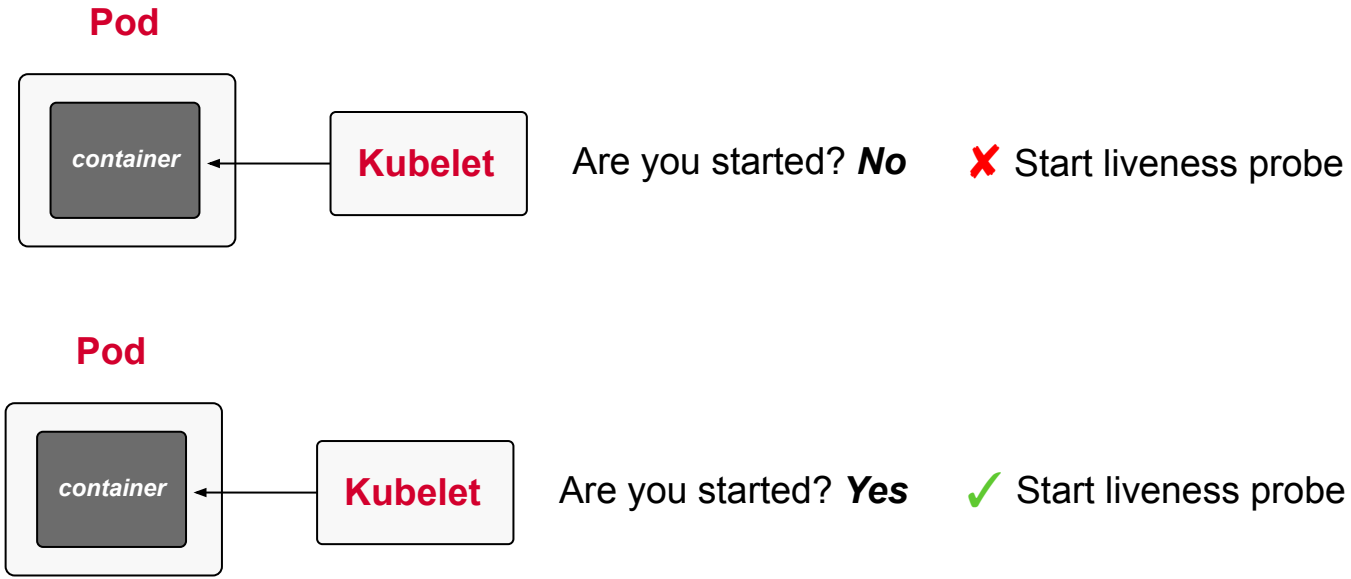
An event log can be queried with a custom command

```
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  containers:
  - name: web-app
    image: eshop:4.6.3
    livenessProbe:
      exec:
        command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 10
      periodSeconds: 5
```

It makes sense to delay the initial check as the application to fully start up first

Understanding Startup Probes

“Legacy application may need longer to start. Hold off on probing.”



Defining a Startup Probe

TCP socket connection if exposed by application

```
apiVersion: v1
kind: Pod
metadata:
  name: startup-pod
spec:
  containers:
  - image: httpd:2.4.46
    name: http-server
    startupProbe:
      tcpSocket:
        port: 80
      initialDelaySeconds: 3
      periodSeconds: 15
```

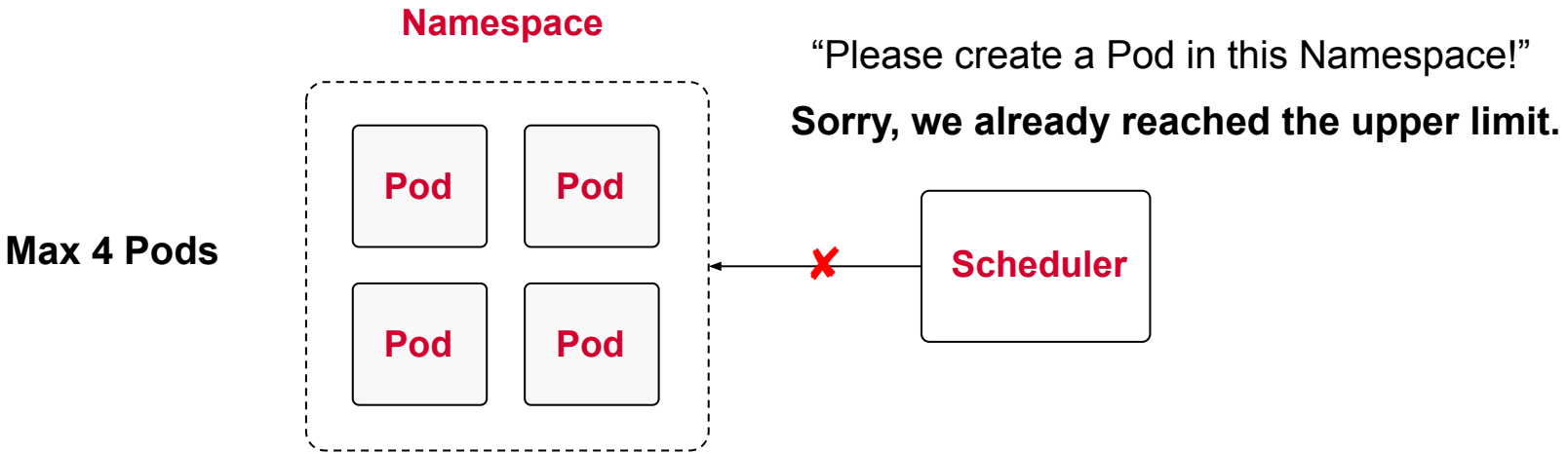
Tries to open a TCP socket connection to a port

EXERCISE

Configuring Health
Probes for a Pod

Defining Resource Boundaries

Defines # of Pods, CPU and memory usage per Namespace



Creating a Resource Quota

Definition on the Namespace-level

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: app
spec:
  hard:
    pods: "2"
    requests.cpu: "2"
    requests.memory: 500m
```

```
$ kubectl create namespace rq-demo
$ kubectl create -f rq.yaml
--namespace=rq-demo
resourcequota/app created
$ kubectl describe quota --namespace=rq-demo
Name:            app
Namespace:       rq-demo
Resource         Used  Hard
--------         ----  ----
pods             0     2
requests.cpu     0     2
requests.memory  0     500m
```

Defining Container Constraints

Required if Namespace defines Resource Quota

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - image: nginx
    name: mypod
    resources:
      requests:
        cpu: "0.5"
        memory: "200m"
```

Requires at least 0.5 CPU resources and 200m of memory

EXERCISE

Defining a Pod’s Resource Requirements

Node Affinity & Taints/Tolerations

Concepts with different purposes, but can go hand in hand

- **Node Affinity:** Attract Pods to a node as soft or hard requirement.
- **Taint:** Allow a node to repel a set of Pods.
- **Tolerations:** Applied to Pods to allow scheduling them to nodes with a specific taint.

Kubernetes Scheduler

`kube-scheduler` is the default scheduler for Kubernetes

```
$ kubectl get pods -n kube-system
NAME                       READY   STATUS      RESTARTS   AGE
kube-scheduler-minikube    1/1     Running     2          76d
...
```

Detailed information on node selection

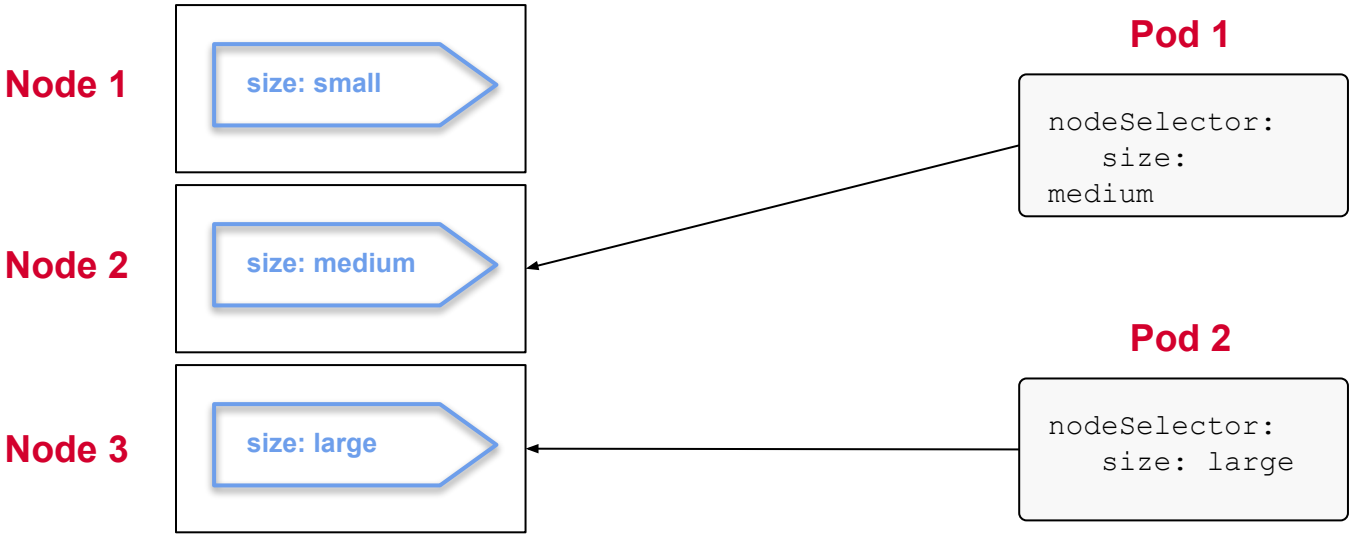
Pod to Node Assignment

Once Pod is scheduled, the node is assigned automatically

```
$ kubectl get nodes
NAME       STATUS   ROLES    AGE    VERSION
minikube   Ready    master   204d   v1.19.2

$ kubectl get pods -o=wide
NAME   READY   STATUS    RESTARTS   AGE   IP           NODE       ...
app    1/1     Running   0          22h   10.0.0.102   minikube   ...

$ kubectl get pod app -o yaml | grep nodeName:
  nodeName: minikube
```

Node Selection Constraint

Define a label selector in Pod's spec that matches node label



Node Selection Constraint

Add labels to nodes

```
$ kubectl label nodes minikube size=medium
node/minikube labeled

$ kubectl get nodes --show-labels
NAME           STATUS   ROLES    AGE    VERSION   LABELS
...
minikube-m02   Ready    master   204d   v1.19.2   ...,size=medium
minikube-m03   Ready    master   204d   v1.19.2   ...
minikube-m04   Ready    master   204d   v1.19.2   ...
```

Node Selection Constraint

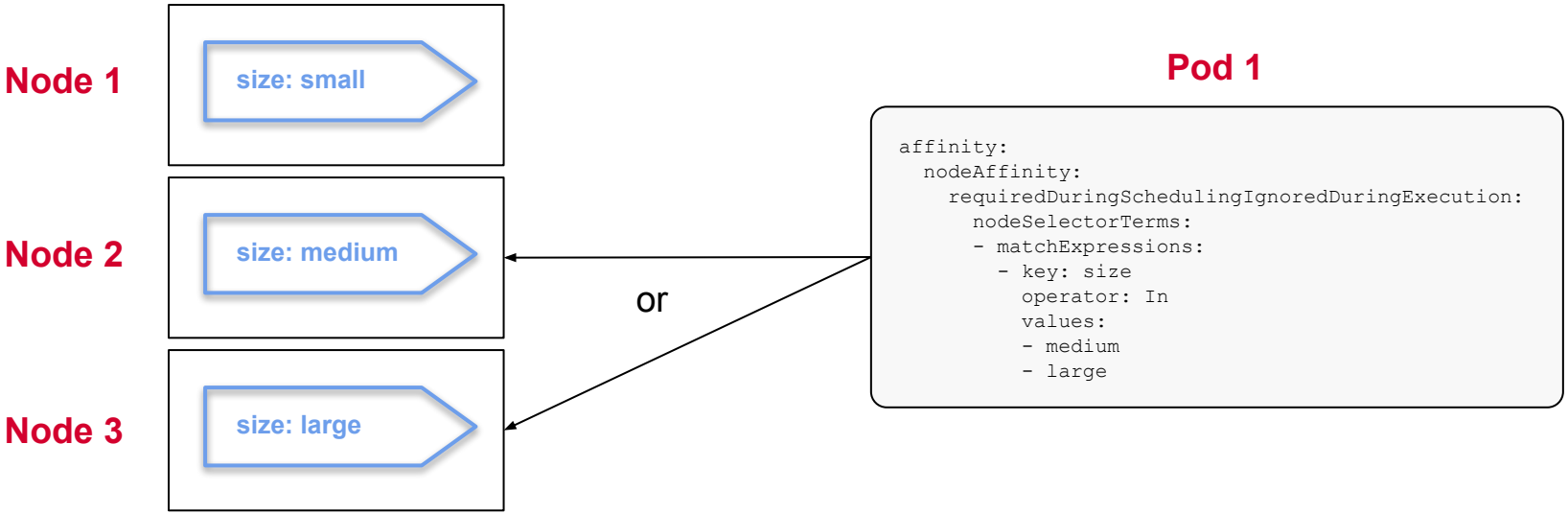
Define the `nodeSelector` attribute upon Pod creation

```
apiVersion: v1
kind: Pod
metadata:
  name: app
spec:
  containers:
  - image: nginx
    name: nginx
  restartPolicy: Never
  nodeSelector:
    size: medium
```

Can't have multiple keys with the same value as the underlying data structure is a map

Node Affinity

Similar to `nodeSelector` but more flexible and powerful



Setting a Pod's Node Affinity

Requires a lot of configuration in various shapes and forms

```
apiVersion: v1
kind: Pod
metadata:
  name: app
spec:
  containers:
  - image: nginx
    name: nginx
  restartPolicy: Never
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: size
            operator: In
            values:
            - medium
```


Available operators:
In, NotIn, Exists, DoesNotExist, Gt, Lt

Node Affinity Types

Currently two types, potentially more in the future

Type	Description
`requiredDuringSchedulingIgnoredDuringExecution`	Rules that must be met for a Pod to be scheduled onto a node.
`preferredDuringSchedulingIgnoredDuringExecution`	Rules that specify preferences that the scheduler will try to enforce but will not guarantee.

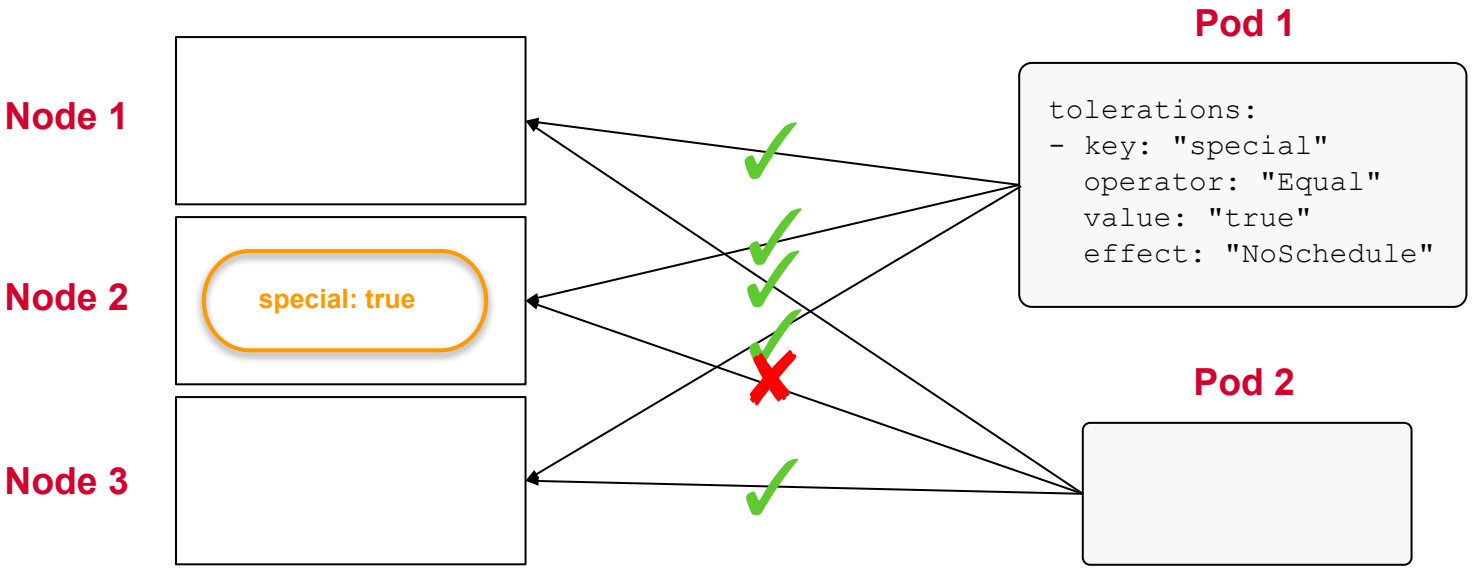
`*IgnoredDuringException` means that changes to the affinity of a running Pod does not have an effect

EXERCISE

Scheduling a Pod
on Specific Nodes

Taints and Tolerations

A Pod that doesn't have specific toleration is repelled



Setting a Node Taint

Add taint to nodes

```
$ kubectl taint nodes minikube-m02 special=true:NoSchedule
node/minikube-m02 tainted

$ kubectl get nodes minikube-m02 -o yaml | grep -C 3 taints:
...
spec:
  taints:
  - effect: NoSchedule
    key: special
    value: "true
```

key=value:effect

Taint Effects

Needs to be provided to node and Pod

Effect	Description
`NoSchedule`	Unless a Pod has matching toleration, it won't be be scheduled on the node.
`PreferNoSchedule`	Try not to place a Pod that does not tolerate the taint on the node, but it is not required.
`NoExecute`	Evict Pod from node if already running on it. No future scheduling on the node.

Setting a Pod's Toleration

Requires a lot of configuration in various shapes and forms

```
apiVersion: v1
kind: Pod
metadata:
  name: app
spec:
  containers:
  - image: nginx
    name: nginx
  restartPolicy: Never
  tolerations:
  - key: "special"
    operator: "Equal"
    value: "true"
    effect: "NoSchedule"
```

Available operators:
`Equal`, `Exists`

EXERCISE

Configuring a Node
to Only Accept
Specific Pods

Q & A

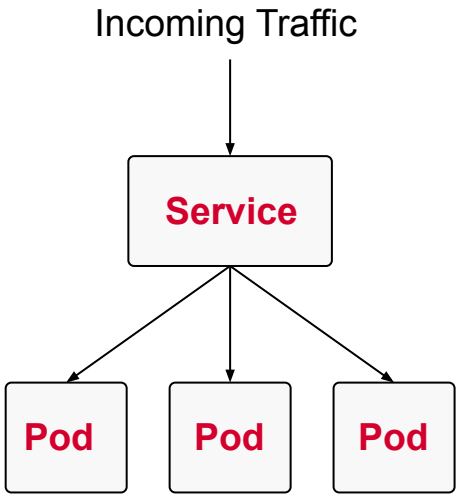
BREAK

Services & Networking

Inter-Pod Communication, Service Types, Ingress, CoreDNS, CNI plugins

Understanding Services

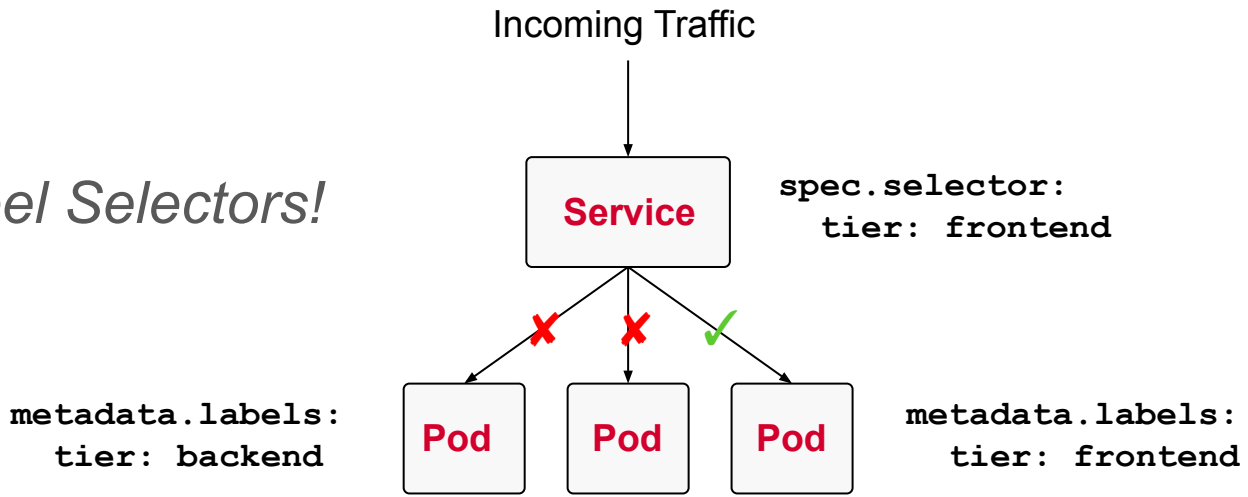
Enables network access for a set of Pods



Request Routing

“How does a service decide which Pod to forward the request to?”

Label Selectors!



Creating a Service (imperative)

“Create a Service with explicit type”

```
$ kubectl create service clusterip nginx --tcp=80:80
service/nginx created
```

Creating a Service (imperative)

“Create a Pod and expose it with a Service”

```
$ kubectl run nginx --image=nginx --port=80 --expose
service/nginx created
pod/nginx created
```

Creating a Service (declarative)

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    tier: frontend
  ports:
  - port: 3000
    protocol: TCP
    targetPort: 80
  type: ClusterIP
```

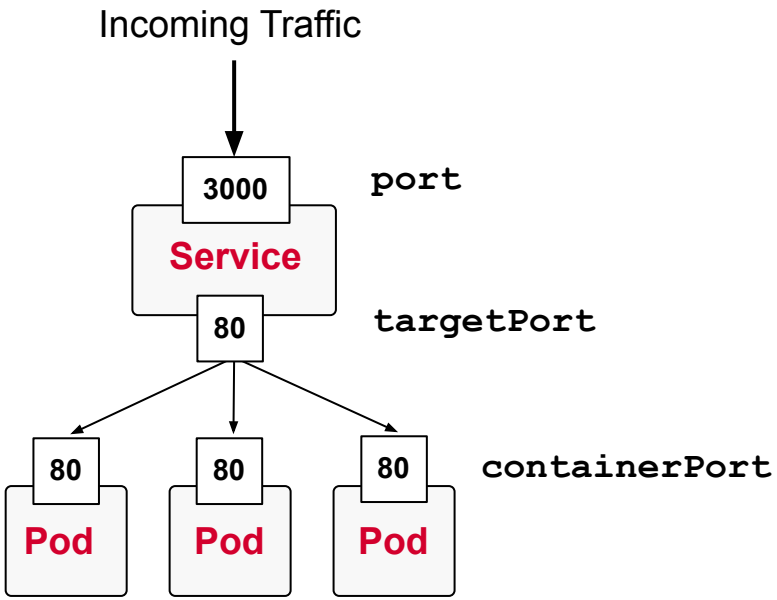
Determines the Pod(s) for routing traffic

Maps incoming port to port of the Pod

Specifies how to expose the Service (inside/outside of cluster or LoadBalancer)

Port Mapping

“How to map the service port to the container port in Pod?”



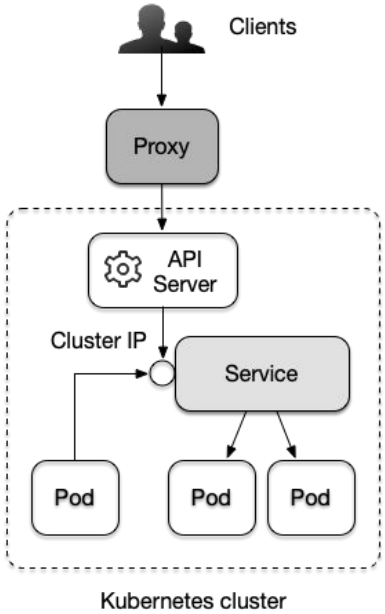
Different Types of Services

Type	Behavior
`ClusterIP`	Exposes the service on a cluster-internal IP. Only reachable from within the cluster.
`NodePort`	Exposes the service on each node's IP at a static port. Accessible from outside of the cluster.
`LoadBalancer`	Exposes the service externally using a cloud provider's load balancer.
`ExternalName`	Map a Service to a DNS name.

```
spec.type: xyz
```

ClusterIP Service Type

Only reachable from within the cluster or API service via proxy

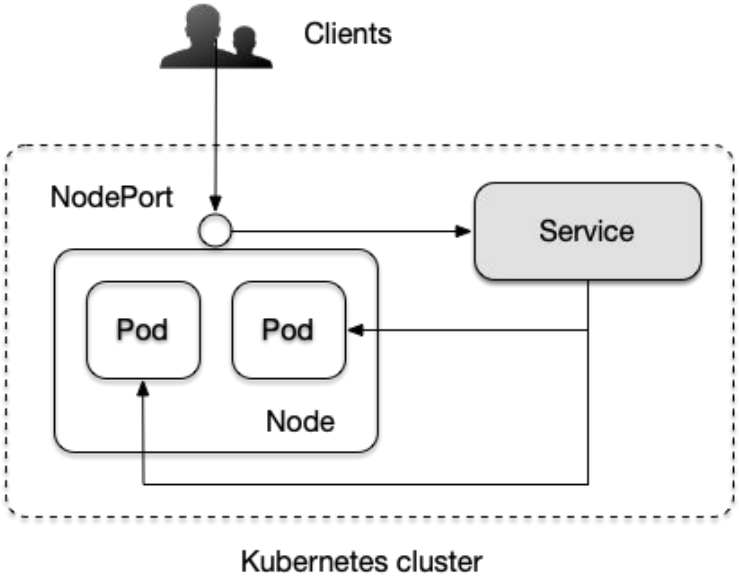


Exposes the Service on a cluster-internal IP address.

Can also be reached by proxy from outside of the cluster using the `kubectl proxy` command.

NodePort Service Type

Accessible from outside of the cluster

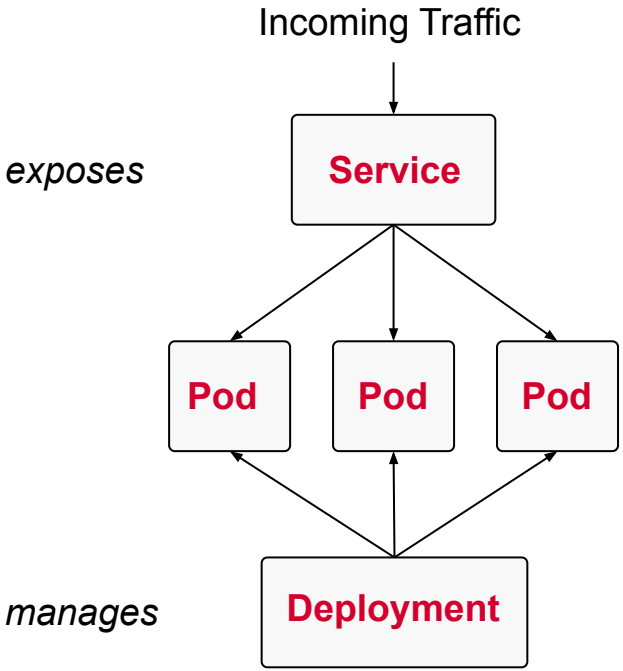


Node's IP address + port number in the range of 30000 and 32767, assigned automatically upon the creation of the Service.

Inspecting a Service

```
# Only reachable from within the cluster
$ kubectl get service nginx
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
nginx   ClusterIP   10.105.201.83   <none>        80/TCP     3h

# Accessible from outside of the cluster
$ kubectl get service nginx
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
nginx   NodePort    10.105.201.83   <none>        80:30184/TCP     3h
```

Deployments and Services



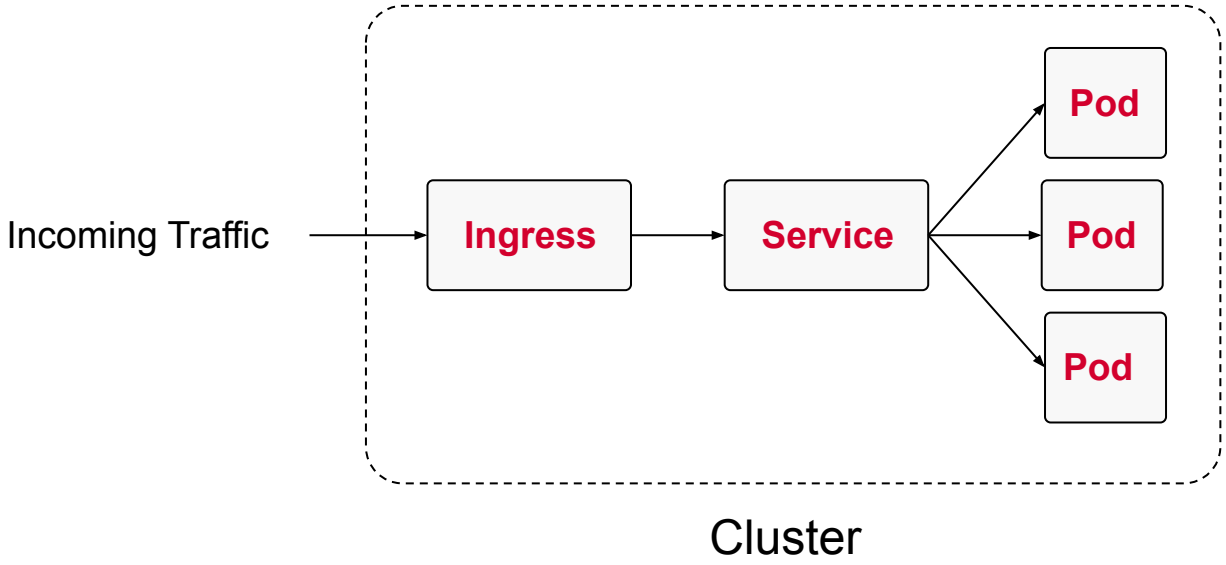
Two distinct concepts that complement each other

EXERCISE

Routing traffic to Pods from Inside and Outside of a Cluster

Understanding Ingress

Manages external access to the services in a cluster via HTTP(S)



Defining an Ingress

Traffic routing is controlled by rules defined on Ingress resource

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

Ingress Rules

Traffic routing is controlled by rules defined on Ingress resource

- Optional host. If not host is defined, then all inbound HTTP traffic is handled.
- A list of paths e.g. `/testpath`.
- The backend, a combination of Service name and port.

Path Types

Incoming URLs match based on type

	Rule	*Request*	
Exact	`/foo`	`/foo`	✓
		`/bar`	✗
Prefix	`/foo`	`/foo, /foo/`	✓
		`/bar`	✗

Listing an Ingress

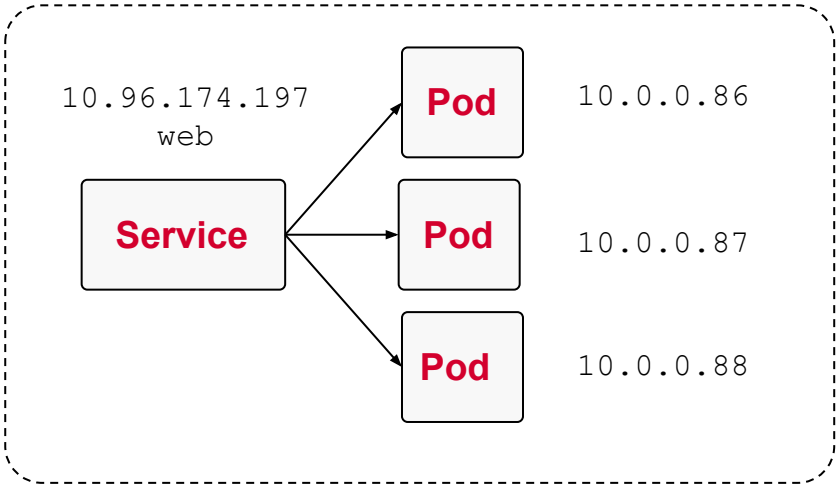
```
$ kubectl get ingress
Warning: extensions/v1beta1 Ingress is deprecated in
v1.14+, unavailable in v1.22+; use networking.k8s.io/v1
Ingress
NAME                CLASS    HOSTS   ADDRESS   PORTS   AGE
minimal-ingress     <none>   *                 80      17s
```

EXERCISE

Defining and Using
an Ingress

DNS for Services

Kubernetes DNS service creates record for Service

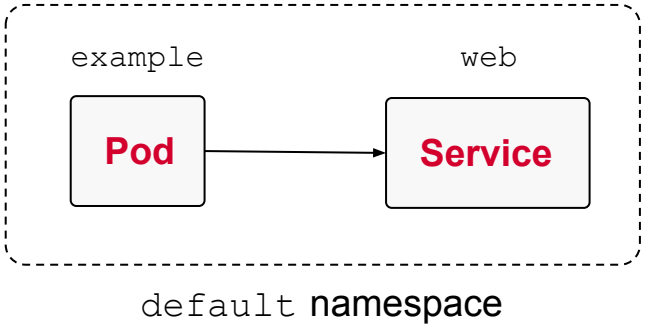


Kubernetes DNS Service

Hostname	IP Address
web	10.96.174.197

Resolving a Service by DNS

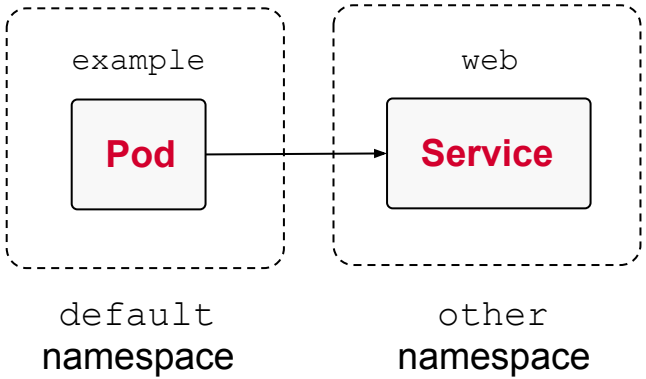
Resolve by hostname within the same namespace




```
$ curl http://web
Hello World
```

Resolving a Service by DNS

Resolve by namespace, type, root domain from another namespace




```
$ curl http://web.other
Hello World

$ curl http://web.other.svc
Hello World

$ curl http://web.other.svc.cluster.local
Hello World
```

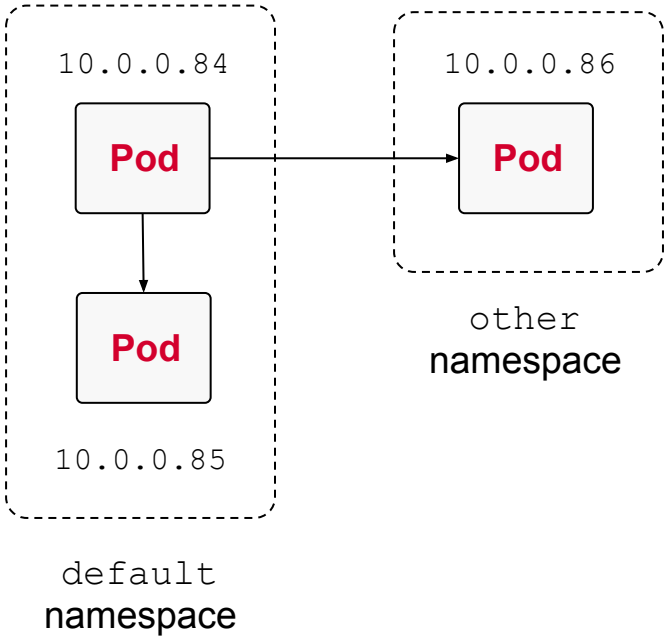
Namespace

Type

Root Domain

Resolving a Pod by DNS

DNS records are not created by default, resolve by IP address




```
$ curl 10.0.0.85
Hello World

$ curl 10.0.0.86
Hello World
```

Object Representation CoreDNS

Recommended Kubernetes DNS server implementation

```
$ kubectl get pods -n kube-system
NAME                       READY   STATUS      RESTARTS   AGE
coredns-f9fd979d6-skk6w    1/1     Running     2          67d

$ kubectl get services -n kube-system
NAME        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
kube-dns    ClusterIP   10.96.0.10       <none>        53/UDP,53/TCP,9153/TCP   195d
```

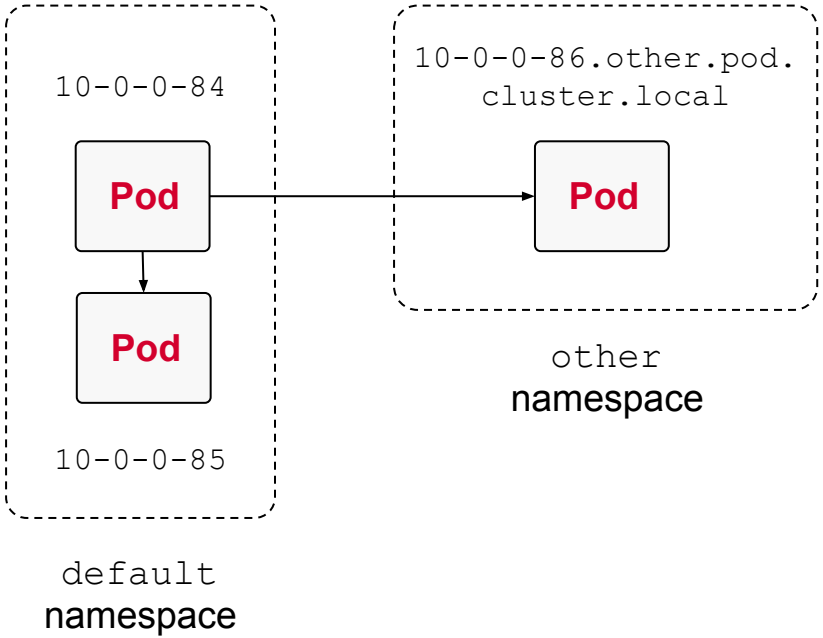
Configuring CoreDNS

Sets root domain and enables DNS for Pods

```
$ kubectl describe configmap coredns -n kube-system
Data
====
Corefile:
----
.:53 {
    ...
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
        ttl 30
    }
    ...
}
```

Resolving a Pod by DNS

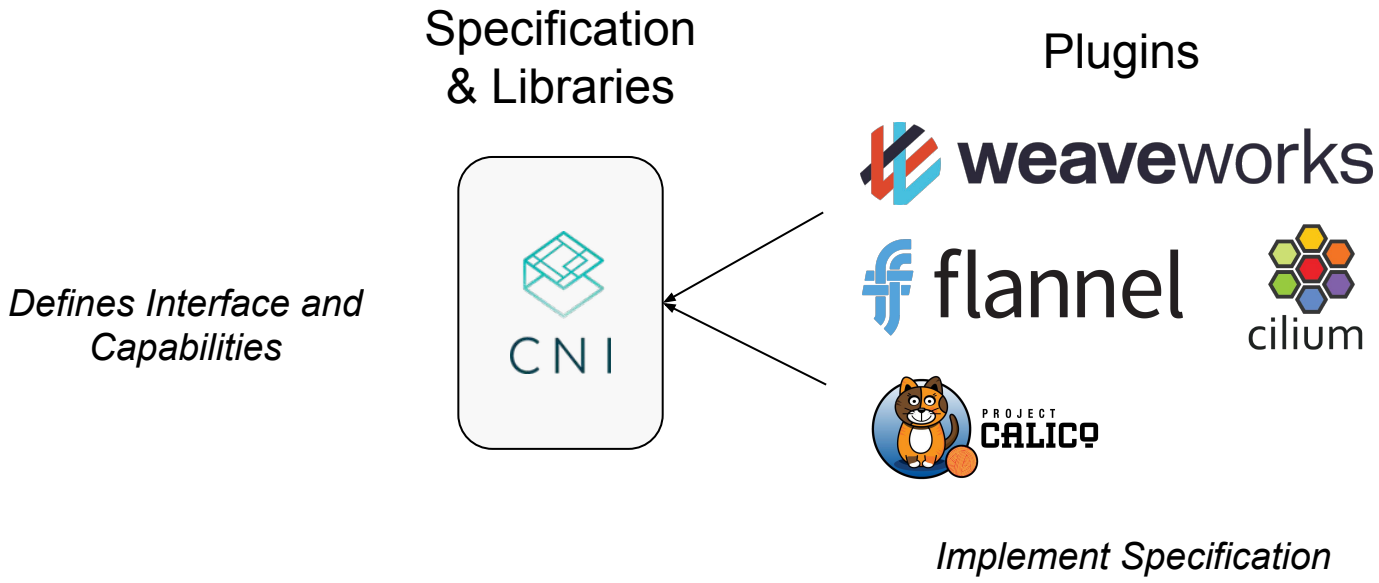
DNS records are not created by default, resolve by IP address




```
$ curl 10-0-0-85
Hello World

$ curl
10-0-0-86.other.pod.cluster
.local
Hello World
```

Understanding CNI

Kubernetes uses CNI for Pod networking



Choosing a CNI Plugin

Direct installation instructions only available for Weave Net

List of plugins:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

Installation instructions for Weave Net:

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/#steps-for-the-first-control-plane-node

Q & A

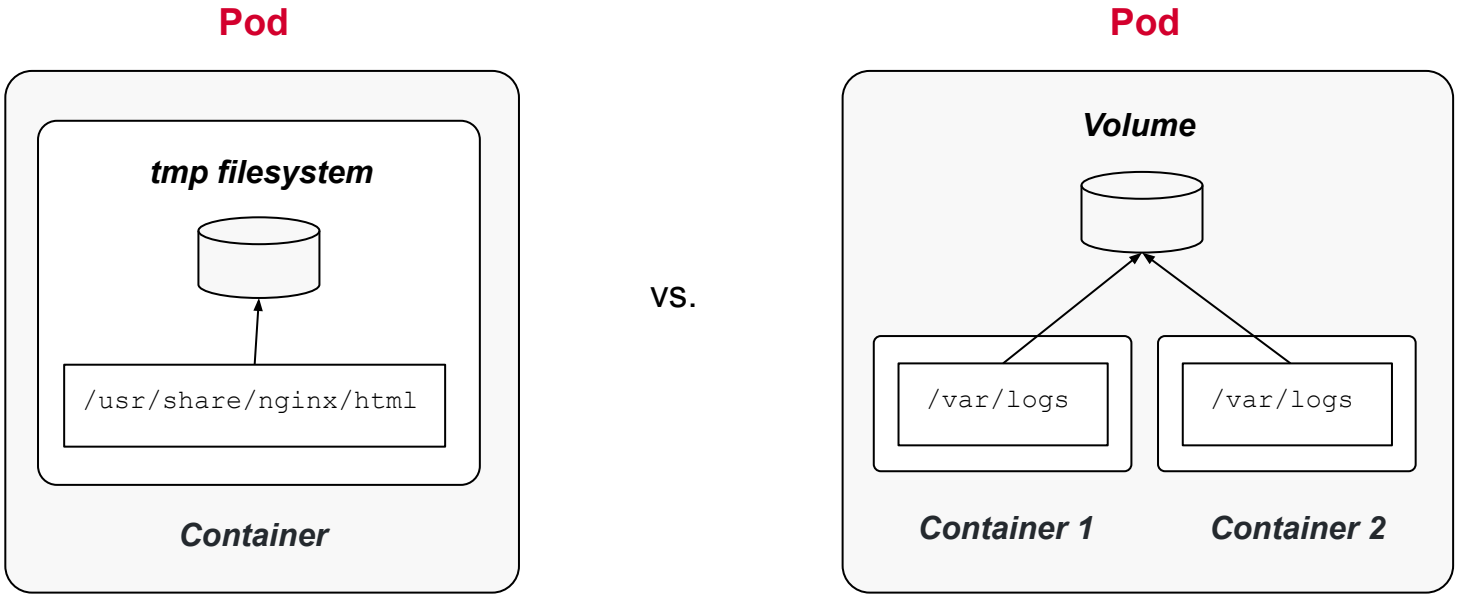
BREAK

Storage

Volumes, Volume Configuration Options,
PersistentVolumes with Static & Dynamic Binding

Understanding Volumes

Persist data that outlives a Pod restart



Types of Volumes

Type	Description
`emptyDir`	Empty directory in Pod. Only persisted for the lifespan of a Pod.
`hostPath`	File or directory from the host node's filesystem into your Pod.
`configMap, secret`	Provides a way to inject configuration data and secrets into Pods.
`nfs`	An existing NFS (Network File System) share to be mounted into your Pod. Preserves data after Pod restart.
Cloud provider solutions	Provider-specific implementation for AWS, GCE or Azure.

Creating a Volume

```
apiVersion: v1
kind: Pod
metadata:
  name: my-container
spec:
  volumes:
  - name: logs-volume
    emptyDir: {}
  containers:
  - image: nginx
    name: my-container
    volumeMounts:
    - mountPath: /var/logs
      name: logs-volume
```

Define Volume with a type

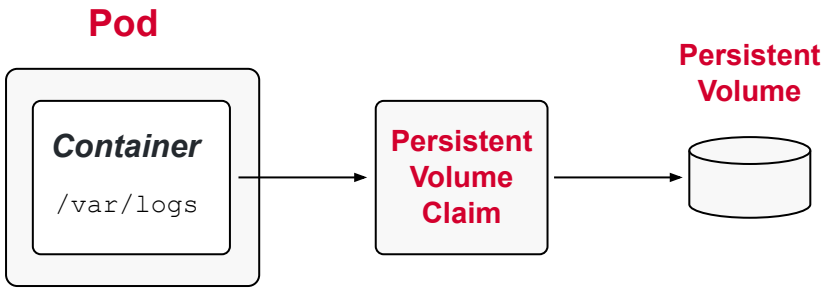
Mount Volume to a path

Using a Volume

```
# Create Pod with mounted Volume
$ kubectl create -f pod-with-vol.yaml
pod/my-container created

# Shell into container and use Volume
$ kubectl exec -it my-container -- /bin/sh
# cd /var/logs
# pwd
/var/logs
# touch app-logs.txt
# ls
app-logs.txt
```

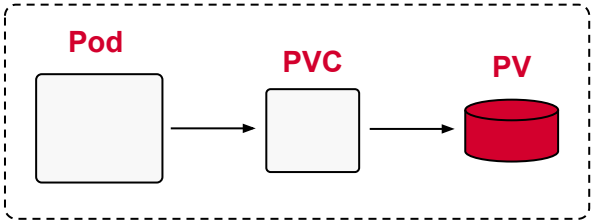
Understanding PersistentVolumes

Persist data that outlives a Pod, node, or cluster restart



Creating a PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv
spec:
  capacity:
    storage: 512m
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data/config
```



Defines a specific storage capacity

Read and/or write access
How many nodes can access volume?

Access Mode & Reclaim Policy

Configuration options for PersistentVolume

Access Mode

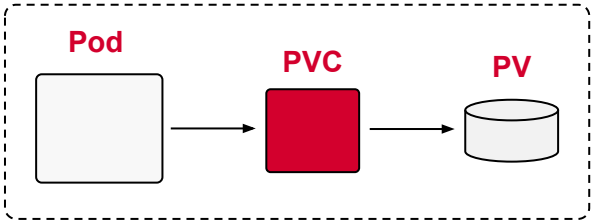
Type	Description
`ReadWriteOnce`	Read-write access by a single node.
`ReadOnlyMany`	Read-only access by many nodes.
`ReadWriteMany`	Read-write access by many nodes.

Reclaim Policy

Type	Description
`Retain`	Default. When PVC is deleted, PV is “released” and can be reclaimed.
`Delete`	Deletion removes PV and associated storage.
`Recycle`	Deprecated. Use dynamic binding instead.

Creating a Claim

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 256m
```

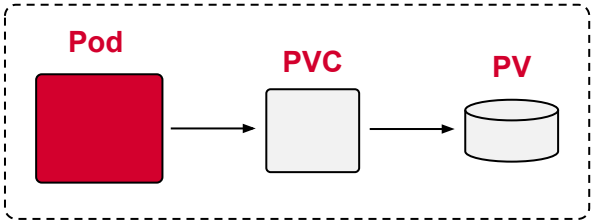


Read and/or write access
How many nodes can access volume?

Defines a specific storage capacity

Mounting a Claim

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: app
spec:
  volumes:
    - name: configpvc
      persistentVolumeClaim:
        claimName: pvc
  containers:
  - image: nginx
    name: app
    volumeMounts:
      - mountPath: "/data/app/config"
        name: configpvc
```



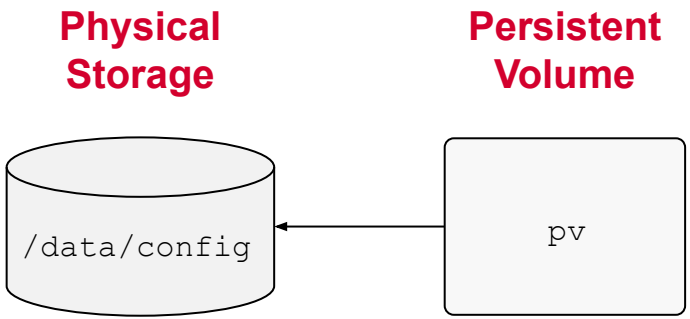
References the Volume with the claim name

Mounts Volume to path

Static Provisioning

Requires the physical storage to exist before PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv
spec:
  capacity:
    storage: 512m
  accessModes:
    - ReadWriteOnce
  storageClassName: shared
  hostPath:
    path: /data/config
```



Dynamic Provisioning

Creates PersistentVolume object automatically via storage class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 256m
  storageClassName: standard
```

EXERCISE

Creating a
Persistent Volume
with Static or
Dynamic Binding

Troubleshooting

Cluster/Node Logging, Monitoring Applications, Identifying and Fixing Application, Cluster, and Networking Issues

Monitoring Cluster Components

What metrics are of interest?

- Number of nodes in the cluster.
- Health status of nodes.
- Node performance metrics like CPU, memory, disk space, network.
- Pod-level performance metrics like CPU, memory consumption.

Monitoring Solution

Relevant to CKA exam: metrics server

Commercial Products

...

vs.

Free Solutions

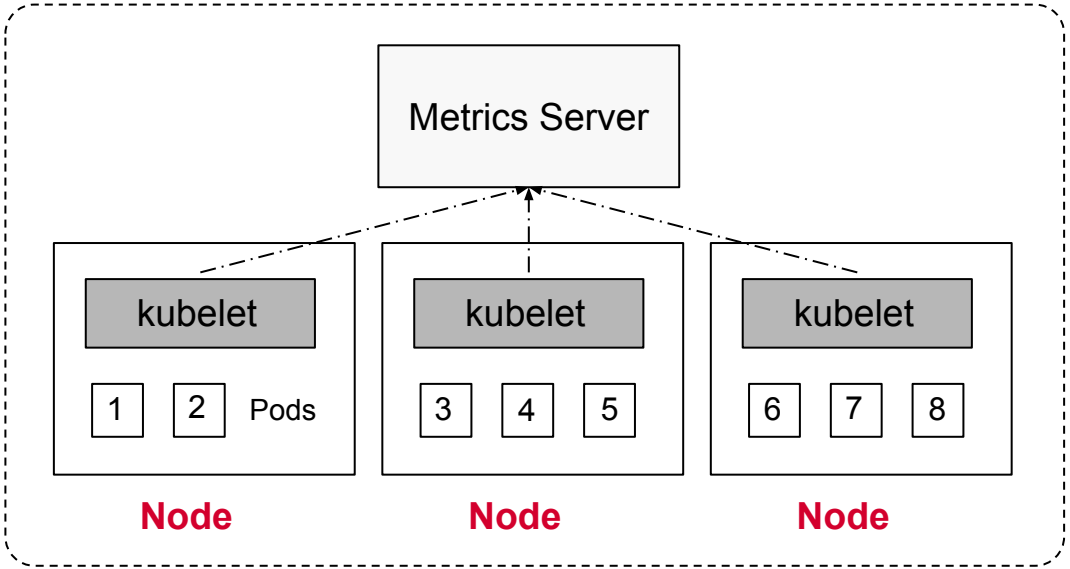
Heapster

✘ Retired

Metrics Server

Metrics Server

Cluster-wide metrics aggregator



Installing the Metrics Server

Add on-component for Minikube or creating objects

```
$ minikube addons enable
metrics-server
The 'metrics-server' addon is enabled
```

```
$ kubectl apply -f
https://github.com/kubernetes-sigs/metrics-server/
releases/latest/download/components.yaml
```

Using the Metrics Server

The `kubectl top` command can query nodes and Pods

```
$ kubectl top nodes
NAME       CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
minikube   283m         14%    1262Mi          32%
```

```
$ kubectl top pod frontend
NAME       CPU(cores)   MEMORY(bytes)
frontend   0m           2Mi
```

Accessing Container Logs

Simply use the `kubectl logs` command

```
$ kubectl logs hazelcast
...
May 25, 2020 3:36:26 PM com.hazelcast.core.LifecycleService
INFO: [10.1.0.46]:5701 [dev] [4.0.1] [10.1.0.46]:5701 is STARTED
```

Use the command line option `--f` to stream the logs

Accessing Container Logs

Specify container name for multi-container Pods

```
$ kubectl logs hazelcast -c app
...
May 25, 2020 3:36:26 PM com.hazelcast.core.LifecycleService
INFO: [10.1.0.46]:5701 [dev] [4.0.1] [10.1.0.46]:5701 is STARTED
```

Use the command line option `-c` or `--container`

Q & A

BREAK

Official Troubleshooting Docs

Detailed advice and help during exam

- Troubleshooting applications
- Troubleshooting cluster

Troubleshooting Services

Check Service type and call endpoint

```
$ kubectl get service nginx
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
nginx   NodePort    10.105.201.83   <none>        80:30184/TCP    3h
```

```
$ curl http://nginx:30184
curl: (6) Could not resolve host: nginx
```

✘ Connectivity issue

Troubleshooting Services

Ensure correct label selection

```
$ kubectl describe service myapp
Name:              myapp
Namespace:         default
Labels:            app=myapp
Annotations:       <none>
Selector:          app=myapp
Type:              ClusterIP
IP:                10.102.22.26
Port:              80-80  80/TCP
TargetPort:        80/TCP
Endpoints:         10.0.0.115:80
Session Affinity:  None
Events:            <none>
```

```
$ kubectl describe pod
myapp
Name:          myapp
...
Labels:        app=myapp
...
```

✓ Matching labels

Troubleshooting Pods

Check the status first - is it running?

```
$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
myapp   1/1     Running   0          12m
```

✓ Healthy status

Troubleshooting Pods

Check the event log - does it indicate issues?

✘ Failed mount

```
$ kubectl describe pod myapp
...
Events:
  Type     Reason       Age                  From               Message
  ----     ------       ----                 ----               -------
  Normal   Scheduled    <unknown>            default-scheduler  Successfully↵
  assigned default/secret-pod to minikube
  Warning  FailedMount  3m15s                kubelet, minikube  Unable to↵
  attach or mount volumes: unmounted volumes=[mysecret], unattached↵
  volumes=[default-token-bf8rh mysecret]: timed out waiting for the condition
  Warning  FailedMount  68s (x10 over 5m18s)  kubelet, minikube ↵
  MountVolume.SetUp failed for volume "mysecret" : secret "mysecret" not found
  ...
```

Troubleshooting Pods

Check the container logs - do you see anything suspicious?

```
$ kubectl logs myapp
...
2019-03-05 10:57:51.112  DEBUG  Receiving order
2019-03-05 10:57:51.112  INFO   Processing payment with ID 345993
2019-03-05 10:57:51.112  ERROR  Can't connect to payment system
```

✘ Connectivity issues

Use the command line option `--previous` to gets the logs from the previous instantiation of a container after a restart

EXERCISE

Troubleshooting an Issue for an Application

Troubleshooting Control Plane

Check the status of the cluster nodes first - are they running?

```
$ kubectl get nodes
NAME       STATUS   ROLES    AGE
VERSION
master     Ready    master   198d
v1.19.2
worker-1   Ready    master   198d
v1.19.2
worker-2   Ready    master   198d
v1.19.2
worker-3   Ready    master   198d
v1.19.2
```

✓ Healthy status

Use the `kubectl cluster-info dump` command for details

Troubleshooting Control Plane

Check the status of control plane Pods - do they indicate issues?

```
$ kubectl get pods -n kube-system
NAME                                READY   STATUS             RESTARTS   AGE
kube-apiserver-minikube             1/1     Running            49         70d
kube-controller-manager-minikube    1/1     CashLoopBackoff    2          70d
kube-proxy-mpgd9                    1/1     Running            2          70d
kube-scheduler-minikube             1/1     Running            2          70d
```

✘ Failing controller manager Pod

Troubleshooting Control Plane

Check the logs of API server Pod

```
$ kubectl logs kube-apiserver-minikube -n kube-system
E1231 21:06:22.978390       1 controller.go:116] loading OpenAPI
spec↲ for "v1beta1.metrics.k8s.io" failed with: OpenAPI spec does not
exist
I1231 21:06:22.978460       1 controller.go:129] OpenAPI
AggregationController: action for item v1beta1.metrics.k8s.io: Rate ↲
Limited Requeue.
I1231 21:06:49.825964       1 client.go:360] parsed scheme: ↲
"passthrough"
```

✓ No error messages

EXERCISE

Troubleshooting an Issue with the Control Plane

Troubleshooting Worker Nodes

Check the status of the worker nodes - are they ready?

```
$ kubectl get nodes
NAME        STATUS      ROLES    AGE    VERSION
master      Ready       master   198d   v1.19.2
worker-1    Ready       master   198d   v1.19.2
worker-2    NotReady    master   198d   v1.19.2
worker-3    Ready       master   198d   v1.19.2
```

✘ Issue with the node `worker-2`

Troubleshooting Worker Nodes

Check condition flags of failing worker node

```
$ kubectl describe node worker-1
...
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----                 ------  -----------------                 ------------------                ------                       -------
  NetworkUnavailable   False   Fri, 25 Dec 2020 10:33:34 -0700   Fri, 25 Dec 2020 10:33:34 -0700   CiliumIsUp                   Cilium is running on this node
  MemoryPressure       False   Thu, 31 Dec 2020 14:39:42 -0700   Thu, 31 Dec 2020 06:31:36 -0700   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure         False   Thu, 31 Dec 2020 14:39:42 -0700   Thu, 31 Dec 2020 06:31:36 -0700   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure          False   Thu, 31 Dec 2020 14:39:42 -0700   Thu, 31 Dec 2020 06:31:36 -0700   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready                True    Thu, 31 Dec 2020 14:39:42 -0700   Thu, 31 Dec 2020 06:31:36 -0700   KubeletReady                 kubelet is posting ready status
...
```

✓ Conditions are healthy

A node with condition status `Unknown` may have crashed

Troubleshooting Worker Nodes

Based on conditions check CPU, memory, processes, disk space

```
$ top
Processes: 568 total, 2 running, 566 sleeping, 2382 threads
15:30:40
Load Avg: 1.96, 1.80, 1.68  CPU usage: 2.49% user, 1.83% sys,
95.66% idle
SharedLibs: 706M resident, 108M data, 196M linkedit.
MemRegions: 192925 total, 6808M resident, 312M private, 5106M
shared. PhysMem: 33G used (4319M wired), 31G unused.
VM: 3461G vsize, 2315M framework vsize, 0(0) swapins, 0(0)
swapouts.
Networks: packets: 6818487/8719M in, 2169040/361M out. Disks:
1305228/17G read, 1354811/32G written.
```

✓ Sufficient memory

```
$ df -h
Filesystem       Size   Used  Avail Capacity iused
ifree %iused  Mounted on
/dev/disk1s1s1  1.8Ti   14Gi   1.6Ti     1%  567557
19538461203    0%   /
devfs           187Ki  187Ki    0Bi   100%     648
0  100%   /dev
/dev/disk1s5    1.8Ti   20Ki  1.6Ti     1%       0
19539028760    0%   /System/Volumes/VM
/dev/disk1s3    1.8Ti  282Mi  1.6Ti     1%     799
19539027961    0%   /System/Volumes/Preboot
/dev/disk1s6    1.8Ti  520Ki  1.6Ti     1%      16
19539028744    0%   /System/Volumes/Update
/dev/disk1s2    1.8Ti  172Gi  1.6Ti    10% 1071918
19537956842    0%   /System/Volumes/Data
map auto_home     0Bi    0Bi    0Bi   100%       0
0  100%   /System/Volumes/Data/home
```

✓ Available disk space

Troubleshooting Worker Nodes

Check Kubelet status

```
$ systemctl status kubelet.service
● kubelet.service - kubelet: The Kubernetes Node Agent
   Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/kubelet.service.d
           └─10-kubeadm.conf
   Active: active (running) since Thu 2021-01-21 22:59:54 UTC; 22min ago
     Docs: https://kubernetes.io/docs/home/
 Main PID: 6171 (kubelet)
    Tasks: 16 (limit: 1151)
   CGroup: /system.slice/kubelet.service
           └─6171 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf
--kubeconfig=/etc/kub
```

✓ Kubelet active

Troubleshooting Worker Nodes

View and inspect systemd logs

```
$ journalctl -u kubelet.service
Jan 22 15:51:25 kube-worker-1 systemd[1]: Started kubelet: The Kubernetes Node Agent.
Jan 22 15:51:25 kube-worker-1 systemd[1]: kubelet.service: Current command vanished from the
unit file, execution of the command list won't be resumed.
Jan 22 15:51:25 kube-worker-1 systemd[1]: Stopping kubelet: The Kubernetes Node Agent...
Jan 22 15:51:25 kube-worker-1 systemd[1]: Stopped kubelet: The Kubernetes Node Agent.
Jan 22 15:51:25 kube-worker-1 systemd[1]: Started kubelet: The Kubernetes Node Agent.
Jan 22 15:51:25 kube-worker-1 kubelet[4330]: F0122 15:51:25.656116    4330 server.go:198] failed
to load Kubelet config file /var/lib/kubelet/config.yaml, error failed to read kubelet config
file "/var/lib/kubelet/config
...
```

✘ Failed to read config file

Troubleshooting Worker Nodes

Check certificate on node

```
$ openssl x509 -in /var/lib/kubelet/pki/kubelet.crt -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = kube-worker-1-ca@1611330698
        Validity
            Not Before: Jan 22 14:51:38 2021 GMT
            Not After : Jan 22 14:51:38 2022 GMT
        Subject: CN = kube-worker-1@1611330698
...
```

✓ Certificate issued by correct CA and not expired

EXERCISE

Troubleshooting an Issue with a Worker Node

Q & A

Summary & Wrap Up

Last words of advice...

Gaining confidence

- Run through practice exams as often as you can
- Read through online documentation start to end
- Know your tools (especially vim, bash, YAML)
- Pick time you are most comfortable, get enough sleep
- Take your first attempt easy but give it your best

O'REILLY®

Thank you