



Software  
Consulting



DevOps  
Projects



Corporate  
Training

“  
**UpGrade**  
Your Business  
with  
**BEST BRAINS**”



Docker



Kubernetes



**Rajesh G**

Master Trainer & CTO,  
Brain Upgrade Academy,  
A division of Unisuraksha Tracking Systems Pvt Ltd

# Kubernetes

Rajesh G

CTO, Managing Partner

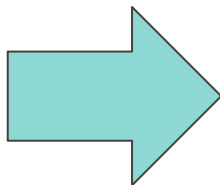
<https://unigps.in>





# Training Objectives

At the end of training,  
participants should be able to



- ☐ Know Kubernetes and Be a Helmsman
- ☐ Create and run PODs
- ☐ Bundle applications & Deploy
- ☐ Service apps using Load Balancers
- ☐ Troubleshoot

# Table of Contents



**BrainUpgrade**  
Academy

Part ONE		
Kubernetes Core	Pod	Pod - Advanced
Architecture	Overview	Patterns - Sidecar, Adapter etc
Components	Lifecycle	Jobs, CronJob
Objects	Probes, Init Containers	Labels, Selectors, Annotations
Kubectl	Topology	Demo & Practicals
Demos & Practicals	Demo & Practicals	
Part TWO		
Scalability	Persistence	Configuration
Deployments	PersistenceVolume	ConfigMaps
Rolling Updates	PersistenceVolumeClaim	Secrets
Rollbacks	Statefulset	SecurityContexts
Auto Scaling	Daemonset	Accounts
Demos & Practicals	Demos & Practicals	Demo & Practicals
Part THREE		
Scheduling	Services & Networking	Monitoring & Misc
Node Name	ClusterIP	Dashboard
Node Selector	NodePort	Logging & Debugging
Affinity	LoadBalancer	Port Forwarding
Taints & Tolerations	Ingress	Tips - CKAD & CKA (CNCF)
Demo & Practicals	Demos & Practicals	Monitoring & Debugging

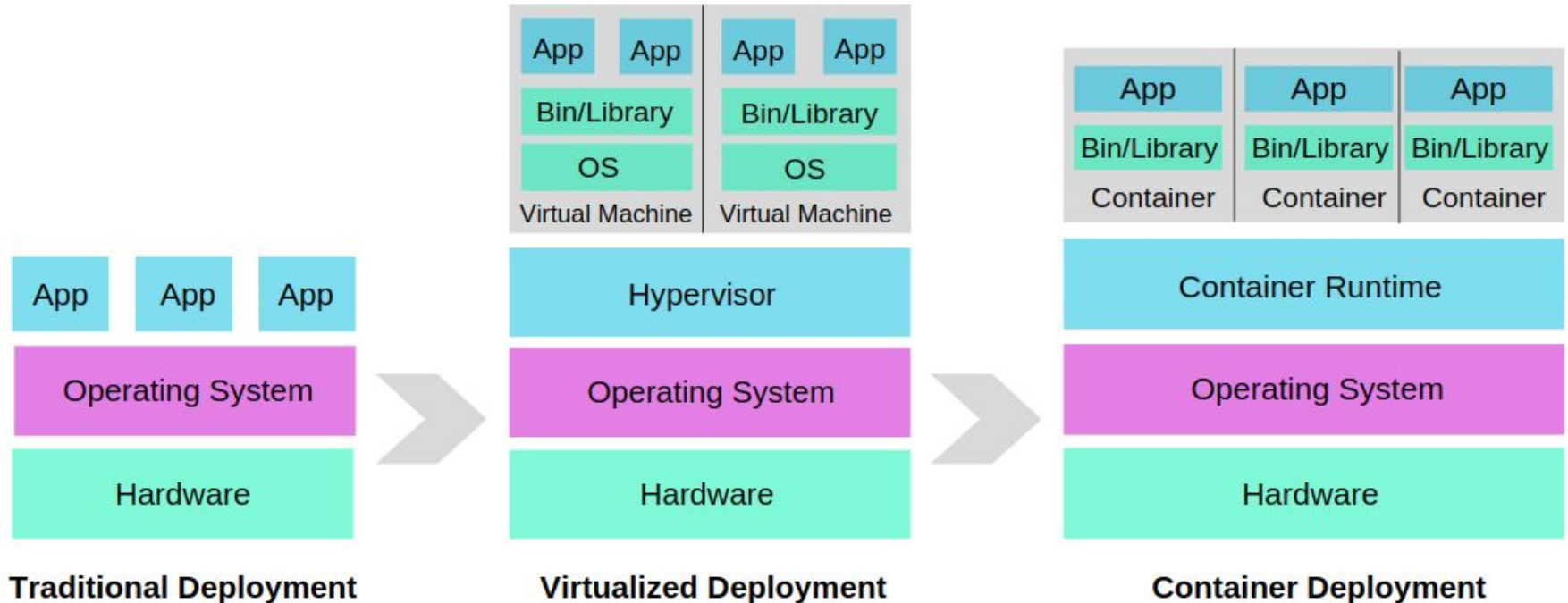


# Kubernetes Core

- Container Journey
  - Why Kubernetes
  - Architecture
  - Core Components
  - API Primitives
  - Kubectl
  - Demo
  - Practicals
-



# Container - Journey





# What is / why Kubernetes

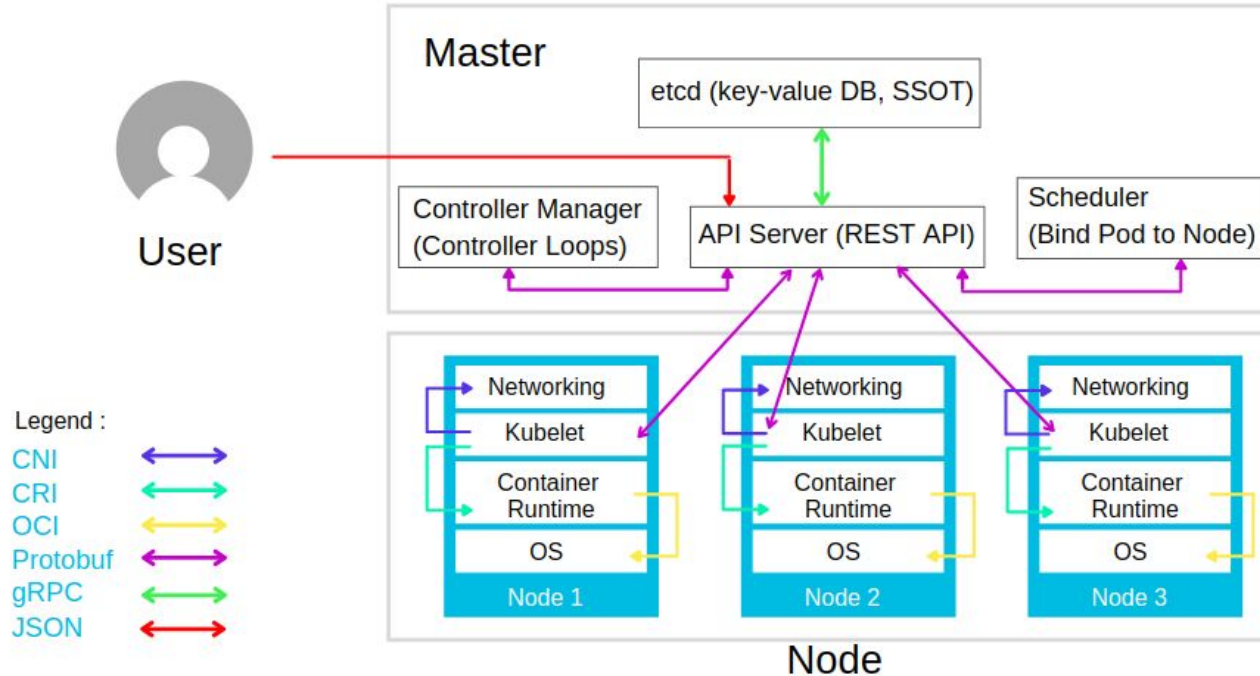
**Kubernetes** is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

## Why?

- Service Discovery & Load Balancing
- Storage Orchestration
- Automated rollouts & rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration Management



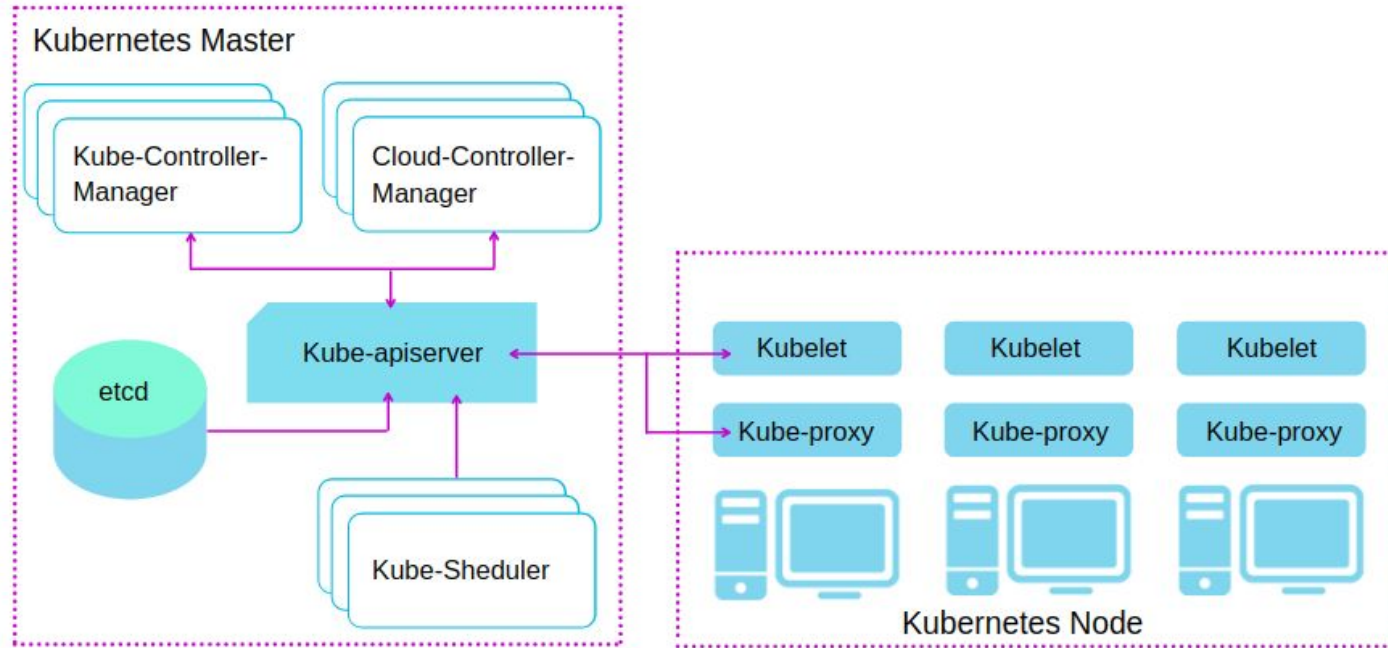
# Architecture Overview







# Architecture Overview





# Master Components - ETCD

- Distributed reliable key-value store that is simple, secure & fast
- Uses RAFT based consensus algorithm to work in distributed environment
- Key value store distributed database
- Runs on port 2379



# Master Components - API Server

- The central management entity
- Only component that connects to ETCD
- Designed for horizontal scaling

## Connectivity:

- External: kubectl
- Internal: kubelet
- Persistent Storage: ETCD



# Master Components - Scheduler

Schedules pods on appropriate Node(s)

Watches for newly created PODs that have no nodes assigned

Decision Parameters:

- Resource requirements (memory, cpu, disk type say SSD)
- Hardware, Software, Policy requirements
- Affinity, Anti-affinity
- Data locality
- Inter workload interference
- Deadlines



# Master Components - Kube Controller

- 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 Controller
  - Create default accounts and API access tokens for new namespaces



# Master Components - Cloud Controller

- 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

- KUBE-PROXY
  - Network proxy that runs on every node in cluster
  - Maintains network rules on nodes
  - Uses OS packet filtering layer else forwards traffic itself
- KUBELET
  - Runs on every node
  - Ensures containers are running & healthy in PODs
  - Doesn't manage container not created by K8S



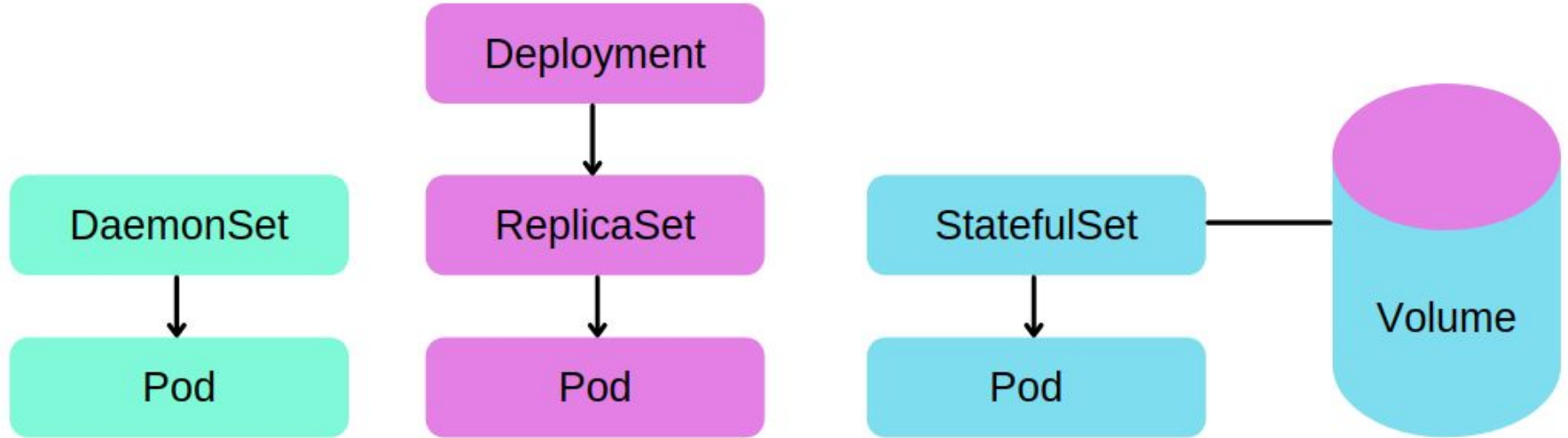
# Addon Components

- Cluster DNS
  - Cluster DNS is a DNS server, in addition to the other DNS server(s) in your environment, which serves DNS records for Kubernetes services
- Web UI
  - General purpose, web-based UI for Kubernetes clusters to view and manager cluster
- Container Resource Monitoring
  - Generic time-series metrics about containers in a central database, and provides a UI for browsing that data
- Cluster level Logging
  - Mechanism responsible for saving container logs to a central log store with search/browsing interface





# Objects





# Kubectl

**Command line tool to control kubernetes cluster**

- Imperative commands to manage objects (basic & intermediate)
  - Deploy commands
  - Cluster Management commands
  - Troubleshooting and Debugging
  - Advanced, Settings and Other
-



# kubectl - commands

- `kubectl get pods`
- `kubectl describe pod hello-world`
- `kubectl describe pod/nginx`
- `kubectl delete pod nginx`
- `kubectl cluster-info`
- `kubectl get pods -o yaml`
- `kubectl get services -o json`
- `kubectl get pods --sort-by=.metadata.name`
- `kubectl get rs,deployments,service`
- `kubectl describe pods`
- `kubectl get pod/<pod-name> svc/<svc-name>`
- `kubectl get pod -l name=<label-name>`
- `kubectl delete pods --all`
- `kubectl get nodes -o json | jq '.items[] | {name:.metadata.name, cap:.status.capacity}'`
- `kubectl get nodes -o yaml | egrep "\sname:|cpu:|memory:"`
- `kubectl get all`
- `kubectl run hello-world --image=tutum/hello-world --port=80`
- `kubectl run -it busybox --image=busybox --restart=Never`
- `kubectl run nginx --image=nginx`



## Exercises

All lab work is available on Google classroom

Join link:

<https://classroom.google.com/c/MzE3MjgxODUzMDUx?cjc=kun7p7g>

---



## POD

- Overview
  - Lifecycle
  - Init Containers
  - Preset
  - Topology Spread
-



# POD - Overview

- ❖ Smallest deployable unit
- ❖ Supports multiple cooperating processes (containers) that form cohesive unit of service
- ❖ Ephemeral Entity

## Encapsulates

- application container(s)
- Storage resources
- Unique network IP

## Shared Resources:

- Networking
- Storage



## Example

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   creationTimestamp: null
5   labels:
6     run: nginx
7   name: nginx
8 spec:
9   containers:
10    - image: nginx
11      name: nginx
12      resources: {}
13    dnsPolicy: ClusterFirst
14    restartPolicy: Always
15 status: {}
```



## Example

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   labels:
5     run: pod-busybox
6   name: pod-busybox
7 spec:
8   containers:
9     - command:
10       - sh
11       - -c
12         echo App is running! && sleep 30
13     image: busybox
14     name: pod-busybox
15     resources: {}
16 restartPolicy: Never
```





# POD - Lifecycle

- Phase
  - Pending (waiting to be scheduled, image downloading)
  - Running (all containers started and ready to serve)
  - Succeeded (all containers exited with success)
  - Failed (all containers exited but at least one with failure)
  - Unknown (unable to fetch status as node is unreachable)
- Container States
  - Waiting, Running, Terminated
- Restart Policy (**Always**, Never, OnFailure)
- Conditions
  - Type: PodScheduled, ContainersReady, Initialized, Ready  
(lastProbeTime, lastTransitionTime, Message, reason, status)
- Probes
  - Startup, Readiness, Liveness
- Lifecycle hooks



## Phase - Pending

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   creationTimestamp: null
5   labels:
6     run: nginx
7     name: nginx
8 spec:
9   containers:
10  - image: nginx
11    name: nginx
12    resources:
13      requests:
14        cpu: "1000m"
15        memory: "1Gi"
16    dnsPolicy: ClusterFirst
17    restartPolicy: Never
18 status: {}
```



## Phase - Running

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   creationTimestamp: null
5   labels:
6     run: busybox
7   name: busybox
8 spec:
9   containers:
10  - command:
11    - ping
12    - google.com
13    image: busybox
14    name: busybox
15    resources: {}
16  dnsPolicy: ClusterFirst
17  restartPolicy: Always
18 status: {}
```



## Phase - Succeeded

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   creationTimestamp: null
5   labels:
6     run: busybox
7   name: busybox
8 spec:
9   containers:
10  - image: busybox
11    name: busybox
12    resources: {}
13  dnsPolicy: ClusterFirst
14  restartPolicy: Never
15 status: {}
```



# Probes

- Types
  - Startup
  - Readiness
  - Liveness
- Methods
  - Http
  - Tcp
  - Command
- Settings
  - `initialDelaySeconds`
  - `periodSeconds`
  - `timeoutSeconds`
  - `successThreshold`
  - `failureThreshold`
- Http
  - Host
  - Scheme
  - Path
  - Port
  - Headers



## Probe - Liveness - Exec

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: probe-liveness-exec
5 spec:
6   containers:
7     - name: probe-liveness-exec
8       image: k8s.gcr.io/busybox
9       args:
10        - /bin/sh
11        - -c
12        - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
13     livenessProbe:
14       exec:
15         command:
16           - cat
17           - /tmp/healthy
18       initialDelaySeconds: 5
19       periodSeconds: 5
```



## Probe - Liveness - http

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: probe-liveness-http
5 spec:
6   containers:
7     - name: probe-liveness-http
8       image: k8s.gcr.io/liveness
9       args:
10        - /server
11       livenessProbe:
12         httpGet:
13           path: /healthz
14           port: 8080
15           httpHeaders:
16             - name: Custom-Header
17               value: Awesome
18         initialDelaySeconds: 3
19         periodSeconds: 3
```



# Probe - Liveness - readiness - tcp

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: probe-liveness-readiness-tcp
5 spec:
6   containers:
7     - name: probe-liveness-readiness-tcp
8       image: k8s.gcr.io/goproxy:0.1
9       ports:
10        - containerPort: 8080
11       readinessProbe:
12         tcpSocket:
13           port: 8080
14         initialDelaySeconds: 5
15         periodSeconds: 10
16       livenessProbe:
17         tcpSocket:
18           port: 8080
19         initialDelaySeconds: 15
20         periodSeconds: 20
```





# Probe - Liveness - startup - http

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: probe-liveness-startup-http
5 spec:
6   containers:
7     - name: probe-liveness-startup-http
8       image: k8s.gcr.io/liveness
9       args:
10        - /server
11       livenessProbe:
12         httpGet:
13           path: /healthz
14           port: 8080
15         failureThreshold: 1
16         periodSeconds: 10
17       startupProbe:
18         httpGet:
19           path: /healthz
20           port: 8080
21         failureThreshold: 30
22         periodSeconds: 10
```



## POD Init Containers

- Always run to completion
- Must complete successfully before next one
- Readiness probes not supported
- Run(s) before application containers

Examples:

- Custom code / utilities to run before app containers
- Block / delay app container startup
- App container image building can be separate



## POD Init - Statuses

- Init:N/M
- Init:Error
- Init:CrashLoopBackOff
- Pending
- PodInitializing
- Running



## Example

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: init-containers
5 spec:
6   containers:
7     - name: main-container
8       image: busybox:1.28
9       command: ['sh', '-c', 'echo The app is running! && sleep 3600']
10  initContainers:
11    - name: init-service
12      image: busybox:1.28
13      command: ['sh', '-c', "until nslookup myservice.$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace).
svc.cluster.local; do echo waiting for myservice; sleep 2; done"]
14    - name: init-mysql
15      image: busybox:1.28
16      command: ['sh', '-c', "until nslookup mysql.$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace).svc.c
luster.local; do echo waiting for mysql; sleep 2; done"]
```

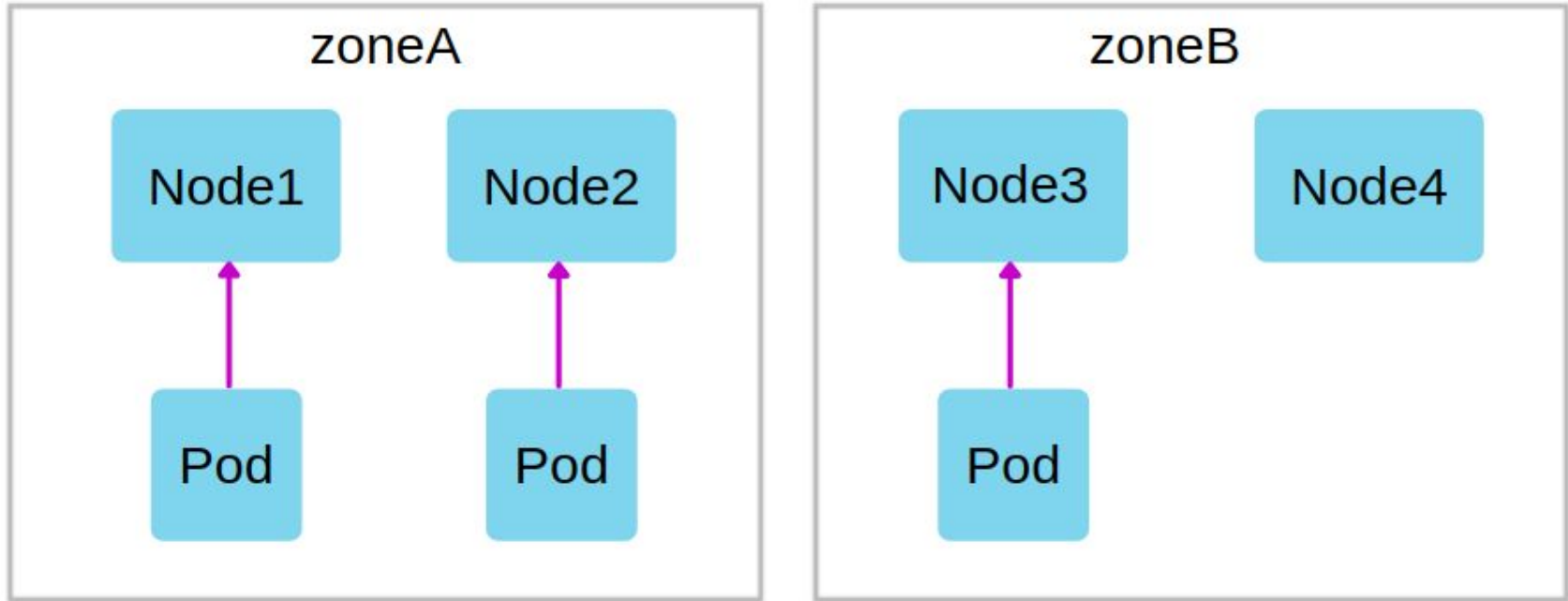


# POD - Topology Spread

- Objectives
  - To control how Pods are spread across **regions**, zones, nodes and other user defined topology domains
  - To achieve high availability
  - To achieve efficient resource utilization
- Spread Constraints
  - maxSkew
  - topologyKey
  - whenUnsatisfiable (DoNotSchedule / ScheduleAnyway)
  - labelSelector



## POD - Topology Spread



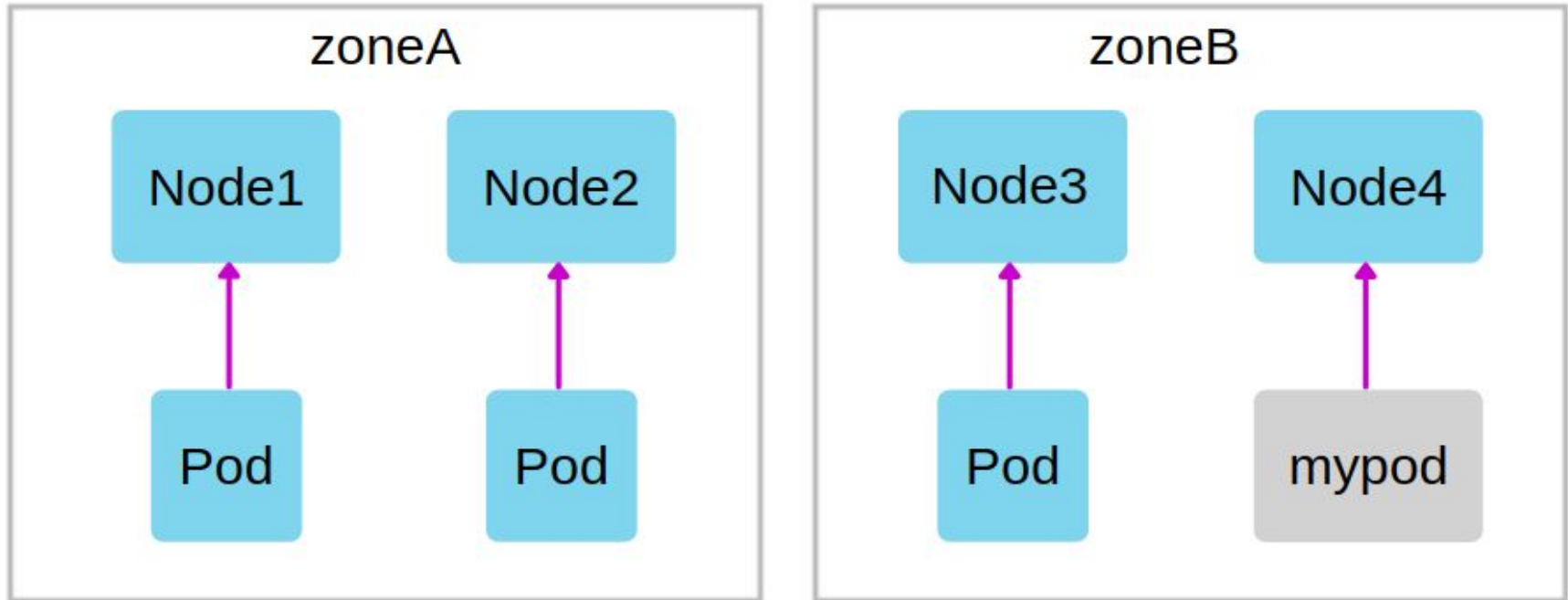


## POD - Topology Spread

```
1 kind: Pod
2 apiVersion: v1
3 metadata:
4   name: topology-zone-constraint-1
5   labels:
6     foo: bar
7 spec:
8   topologySpreadConstraints:
9     - maxSkew: 1
10     topologyKey: failure-domain.beta.kubernetes.io/zone
11     whenUnsatisfiable: DoNotSchedule
12     labelSelector:
13       matchLabels:
14         foo: bar
15   containers:
16     - name: pause
17       image: k8s.gcr.io/pause:3.1
```



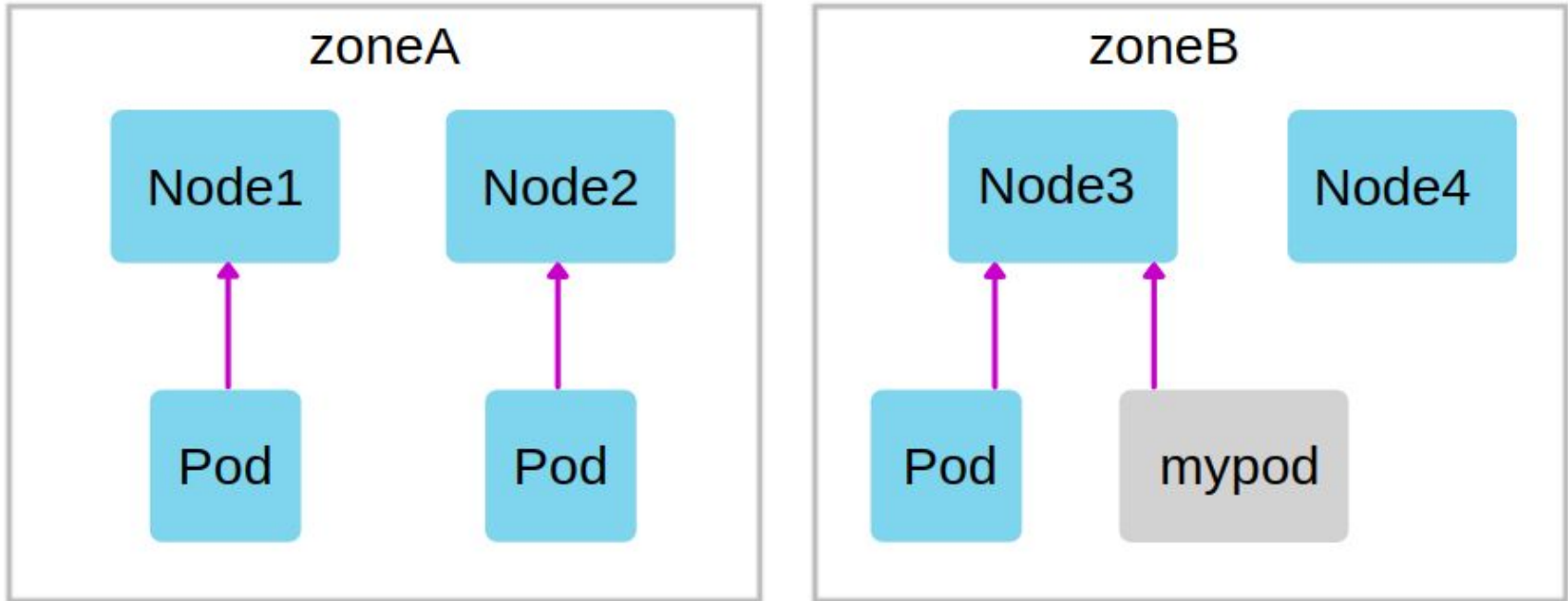
## POD - Topology Spread







## POD - Topology Spread





# Topology - Multiple Constraints

```
1 kind: Pod
2 apiVersion: v1
3 metadata:
4   name: topology-constraints-two
5   labels:
6     foo: bar
7 spec:
8   topologySpreadConstraints:
9     - maxSkew: 1
10     topologyKey: topology.kubernetes.io/zone
11     whenUnsatisfiable: DoNotSchedule
12     labelSelector:
13       matchLabels:
14         foo: bar
15     - maxSkew: 1
16     topologyKey: kubernetes.io/hostname
17     whenUnsatisfiable: DoNotSchedule
18     labelSelector:
19       matchLabels:
20         foo: bar
21   containers:
22     - name: pause
23       image: k8s.gcr.io/pause:3.1
```



# Topology - Constraint & Node Affinity

```
1 kind: Pod
2 apiVersion: v1
3 metadata:
4   name: topology-constraint-nodeaffinity
5   labels:
6     foo: bar
7 spec:
8   topologySpreadConstraints:
9     - maxSkew: 1
10     topologyKey: zone
11     whenUnsatisfiable: DoNotSchedule
12     labelSelector:
13       matchLabels:
14         foo: bar
15   affinity:
16     nodeAffinity:
17       requiredDuringSchedulingIgnoredDuringExecution:
18         nodeSelectorTerms:
19           - matchExpressions:
20             - key: zone
21               operator: NotIn
22               values:
23                 - zoneC
24   containers:
25     - name: pause
26       image: k8s.gcr.io/pause:3.1
```



## Exercises

All lab work is available on Google classroom

Join link:

<https://classroom.google.com/c/MzE3MjgxODUzMDUx?cjc=kun7p7g>

---



## Pod - Advanced

- Patterns
  - Ambassador
  - Sidecar
  - Adapter
- Jobs
- Cron Jobs
- Labels, Selectors, Annotations
- Demo
- Practicals



# Patterns - POD

- To extend the functionality of the existing container
- To have helper process enhancing work of the existing container
- To send logs to external server
- Types
  - Sidecar - To export logs
  - Ambassador - To proxy connection
  - Adapter - To standardise and normalize output



# Pattern - Sidecar

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: sidecar
5 spec:
6   volumes:
7     - name: shared-logs
8       emptyDir: {}
9   containers:
10    - name: main-container
11      image: alpine
12      command: ["/bin/sh"]
13      args: ["-c", "while true; do date >> /var/log/index.html; sleep 10;done"]
14      volumeMounts:
15        - name: shared-logs
16          mountPath: /var/log
17    - name: sidecar-container
18      image: nginx
19      ports:
20        - containerPort: 80
21      volumeMounts:
22        - name: shared-logs
23          mountPath: /usr/share/nginx/html
```



# Pattern - Adapter

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: adapter
5 spec:
6   volumes:
7     - name: shared-logs
8       emptyDir: {}
9   containers:
10    - name: main-container
11      image: alpine
12      command: ["/bin/sh"]
13      args: ["-c", "while true; do date > /var/log/top.txt && top -n 1 -b >> /var/log/top.txt; sleep 10;done"]
14      volumeMounts:
15        - name: shared-logs
16          mountPath: /var/log
17    - name: adapter-container
18      image: alpine
19      command: ["/bin/sh"]
20      args: ["-c", "while true; do (cat /var/log/top.txt | head -1 > /var/log/status.txt) && (cat /var/log/top.txt | head -2 | tail -1 | grep
21 -o -E '\\d+\\w' | head -1 >> /var/log/status.txt) && (cat /var/log/top.txt | head -3 | tail -1 | grep
22 -o -E '\\d+%' | head -1 >> /var/log/status.txt); sleep 5; done"]
23      volumeMounts:
24        - name: shared-logs
25          mountPath: /var/log
```





# Pattern - Ambassador

```
1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: ambassador-nginx-config
5 data:
6   nginx.conf: |
7     worker_processes 1;
8     worker_rlimit_nofile 4096;
9     events {
10       worker_connections 512;
11     }
12     http {
13       proxy_set_header Host $host;
14       proxy_set_header X-Forwarded-Proto $scheme;
15       proxy_set_header X-Real-IP $remote_addr;
16       proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
17       upstream backend {
18         server msn.com:80;
19       }
20       server {
21         listen 80;
22         location / {
23           proxy_pass http://backend;
24         }
25       }
26     }
27 ---
28 apiVersion: v1
29 kind: Pod
30 metadata:
31   name: multi-pod-ambassador
32 spec:
33   containers:
34     - name: main-app
35       image: busybox
36       imagePullPolicy: IfNotPresent
37       command: ["/bin/sh"]
38       args: ["-c", "while true;do wget -O /tmp/app.txt localhost ;sleep 30;done"]
39     - name: ambassador
40       image: nginx
41       imagePullPolicy: IfNotPresent
42       ports:
43         - containerPort: 80
44       volumeMounts:
45         - name: nginx-config
46           mountPath: /etc/nginx/nginx.conf
47           subPath: nginx.conf
48   volumes:
49     - name: nginx-config
50       configMap:
51         name: ambassador-nginx-config
```



# Job

- To provide reliable parallel execution of tasks
- Examples:
  - Send emails, transcode files, Scan database for a set of rows,
- Patterns
  - Non parallel job
  - Fixed completion count job
  - Work queue job



# Job

```
1 apiVersion: batch/v1
2 kind: Job
3 metadata:
4   name: job
5 spec:
6   template:
7     spec:
8       containers:
9         - name: perl
10           image: perl
11           command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
12       restartPolicy: Never
13   backoffLimit: 4
```



## Job - Timeout

```
1 apiVersion: batch/v1
2 kind: Job
3 metadata:
4   name: job-timeout
5 spec:
6   backoffLimit: 5
7   activeDeadlineSeconds: 100
8   template:
9     spec:
10      containers:
11      - name: perl
12        image: perl
13        command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
14      restartPolicy: Never
```



# CronJob

- Creates jobs on a repeating schedule
- Schedule times are based on kube-controller-manager
- Useful for tasks like migrating data to reporting server, sending emails, creating backups etc
- Schedule tasks at specific time (like when cluster is idle)

## Key Configurations:

- `startingDeadlineSeconds` - Missed occurrences in last X seconds will be counted
- `concurrencyPolicy`
  - If Allow, then job will run at least once
  - If Forbid, will be missed if previous instance is still running



# CronJob - Expression

```
# _____ minute (0 - 59)
# |_____ hour (0 - 23)
# ||_____ day of the month (1 - 31)
# |||_____ month (1 - 12)
# |||_____ day of the week (0 - 6) (Sunday to Saturday;
# |||                                     7 is also Sunday on some systems)
# |||
# |||
# |||
# * * * * * <command to execute>
```

## Examples:

- `*/15 0,8,16 * * * echo running backup` (every 15 minutes of 0,8 & 16th hour)
- `30 0 * * 6 /home/oracle/scripts/export_dump.sh` (last day of week at 00:30)
- `1 0 * * * printf "" > /var/log/apache/error_log` (everyday at 00:01 )



# CronJob - Expression

```
1 apiVersion: batch/v1beta1
2 kind: CronJob
3 metadata:
4   name: cron-job
5 spec:
6   schedule: "*/1 * * * *"
7   jobTemplate:
8     spec:
9       template:
10        spec:
11          containers:
12            - name: cron-job
13              image: busybox
14              args:
15                - /bin/sh
16                - -c
17                - date; echo Migrating data to reporting server...
18          restartPolicy: OnFailure
19
```



# Labels

- Labels
  - Key value pairs attached to objects
  - To specify identifying attributes of objects
  - To organize and select subset of objects
  - To query objects efficiently (cli as well gui monitoring tools)
  - Attached at creation time and can be added / modified at any time
  - Label key must be unique per object
- Example labels:
  - `"release" : "stable", "release" : "canary"`
  - `"environment" : "dev", "environment" : "qa", "environment" : "production"`
  - `"tier" : "frontend", "tier" : "backend", "tier" : "cache"`
  - `"partition" : "customerA", "partition" : "customerB"`
  - `"track" : "daily", "track" : "weekly"`





# Labels

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: pod-labels
5   labels:
6     environment: production
7     app: nginx
8 spec:
9   containers:
10  - name: nginx
11    image: nginx
12    ports:
13  - containerPort: 80
```



# Selectors

- Equality Based
  - `environment = production`
  - `tier != frontend`
- Set Based
  - `environment in (production, qa)`
  - `tier notin (frontend, backend)`
  - `partition`
  - `!partition`



# Selectors - Examples

- `kubectl get pods -l environment=production,tier=frontend`
- `kubectl get pods -l 'environment in (production),tier in (frontend)'`
- `kubectl get pods -l 'environment in (production, qa)'`
- `kubectl get pods -l 'environment,environment notin (frontend)'`

Jobs, Deployments, ReplicaSet, Daemonset

```
selector:  
  matchLabels:  
    component: redis  
  matchExpressions:  
    - {key: tier, operator: In, values: [cache]}  
    - {key: environment, operator: NotIn, values: [dev]}
```



## Selectors - Examples

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: selector-pod-node
5 spec:
6   containers:
7     - name: cuda-test
8       image: "k8s.gcr.io/cuda-vector-add:v0.1"
9       resources:
10        limits:
11          nvidia.com/gpu: 1
12   nodeSelector:
13     accelerator: nvidia-tesla-p100
```



# Annotations

- To attach non-identifying arbitrary metadata to objects
- Usage
  - Pointers for debugging purposes
  - Build, release, image hashes etc
  - Author info, contact details
  - Metadata to help tools for deployment, management, introspection

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: annotations-pod
5   annotations:
6     imageregistry: "https://hub.docker.com/"
7 spec:
8   containers:
9     - name: nginx
10      image: nginx
11      ports:
12        - containerPort: 80
```



## **Exercises - Reference**

All lab work is available on Google classroom

Join link:

<https://classroom.google.com/c/MzE3MjgxODUzMDUx?cjc=kun7p7g>

---



# Scalability

- Deployments
    - Rolling Updates & Rollbacks
    - Auto scaling pods
  - Demo
  - Practicals
-



# Deployments

## Use Cases

- To rollout a set of PODs
- To declare a new set of PODs
- To rollback to an earlier version of deployment
- To scale up deployment to facilitate more load
- To pause the deployment / rollout
- To autoscale deployment when cpu usage threshold reached





# Deployment - Example

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: test-app
5 spec:
6   replicas: 1
7   selector:
8     matchLabels:
9       component: test-app
10  template:
11    metadata:
12      labels:
13        component: test-app
14    spec:
15      containers:
16        - name: test-app
17          image: brainupgrade/test-app:all-tiers-in-one
18          imagePullPolicy: IfNotPresent
19          ports:
20            - containerPort: 8080
21      resources:
22        requests:
23          cpu: "100m"
24          memory: "250Mi"
25
```



# Deployment - Commands

- `kubectl create deployment nginx --image=nginx:1.15 --replicas=5`
- `kubectl get deployment/nginx`
- `kubectl describe deployment/nginx`
- `kubectl rollout history deployment/nginx`
- `kubectl set image deployment/nginx nginx=nginx:1.16`
- `kubectl rollout history deployment/nginx`
- `kubectl rollout undo deployment/nginx`
- `kubectl rollout undo deployment/nginx --to-revision=2`
- `kubectl scale --replicas=50 deployment/nginx`
- `kubectl rollout pause deployment/nginx`
- `kubectl rollout status deployment/nginx`
- `kubectl rollout resume deployment/nginx`
- `kubectl autoscale deployment/nginx --min=2 --max=10`



## Exercises

All lab work is available on Google classroom

Join link:

<https://classroom.google.com/c/MzE3MjgxODUzMDUx?cjc=kun7p7g>

---



## Persistence

- Persistence Volume
  - Persistence Volume Claim
  - Statefulset
  - Daemonset
-



# Overview - PV / PVC

- Ephemeral
  - Tightly coupled with POD lifetime
  - Deleted when POD is removed
  - Example: emptydir
- Persistent
  - Survives POD reboots
  - Meant for long term and independent of POD / Node lifecycle
  - Examples: hostpath, NFS, Cloud storage (EBS etc)
- The access modes are:
  - ReadWriteOnce -- the volume can be mounted as read-write by a single node
  - ReadOnlyMany -- the volume can be mounted read-only by many nodes
  - ReadWriteMany -- the volume can be mounted as read-write by many nodes



## Examples - emptyDir

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: volume-emptydir
5 spec:
6   containers:
7     - image: nginx
8       name: test-container
9       volumeMounts:
10      - mountPath: /cache
11        name: cache-volume
12   volumes:
13     - name: cache-volume
14       emptyDir: {}
```



## Examples - hostpath (file/dir)

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: hostpath-volume
5 spec:
6   containers:
7     - image: nginx
8       name: test-container
9       volumeMounts:
10        - mountPath: /data-mounted-as
11          name: hostpath-volume
12   volumes:
13     - name: hostpath-volume
14       hostPath:
15         # directory location on host
16         path: /data
17         # this field is optional
18         type: DirectoryOrCreate
```



# Persistent Volume - local & node

```
1 apiVersion: v1
2 kind: PersistentVolume
3 metadata:
4   name: pv-local-node-affinity
5 spec:
6   capacity:
7     storage: 10Gi
8   # volumeMode field requires BlockVolume Alpha feature gate to be enabled.
9   volumeMode: Filesystem
10  accessModes:
11    - ReadWriteMany
12  persistentVolumeReclaimPolicy: Delete
13  storageClassName: local-storage
14  local:
15    path: /mnt/disks/ssd1
16  nodeAffinity:
17    required:
18      nodeSelectorTerms:
19        - matchExpressions:
20          - key: kubernetes.io/hostname
21            operator: In
22            values:
23              - ip-172-31-87-231.ec2.internal
```



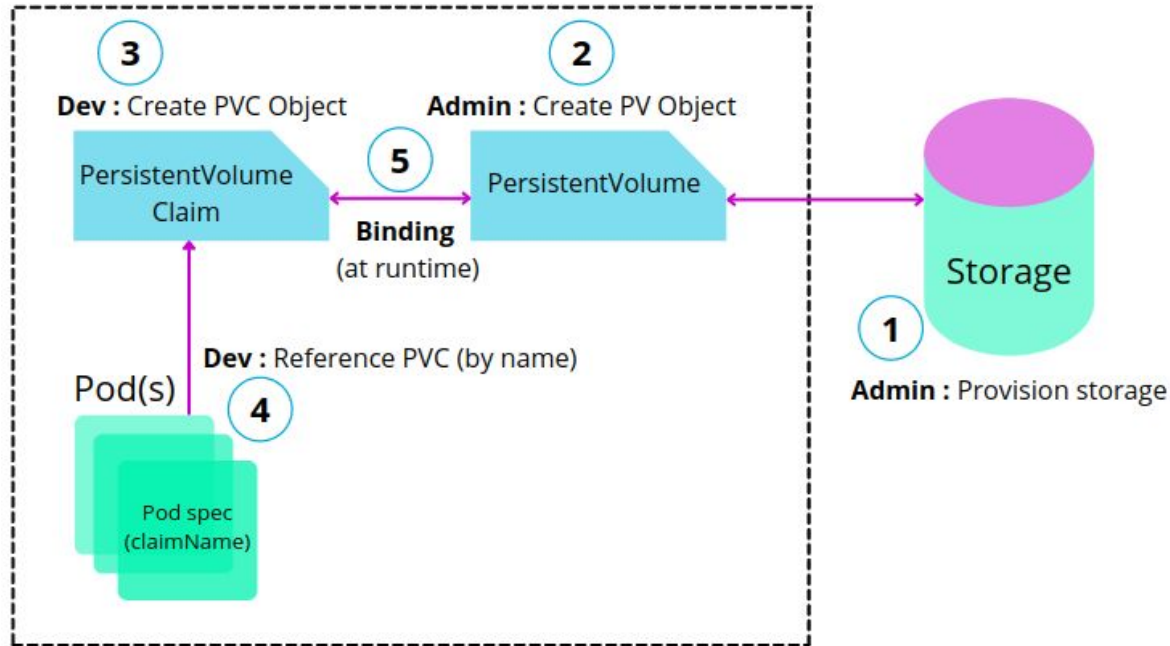


## Persistent Volume - EBS

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: volume-ebs
5 spec:
6   containers:
7     - image: k8s.gcr.io/test-webserver
8       name: test-container
9       volumeMounts:
10        - mountPath: /test-ebs
11          name: ebs-volume
12   volumes:
13     - name: ebs-volume
14       # This AWS EBS volume must already exist.
15       awsElasticBlockStore:
16         volumeID: <volume-id>
17         fsType: ext4
```



# Persistent Volume - static





## Example

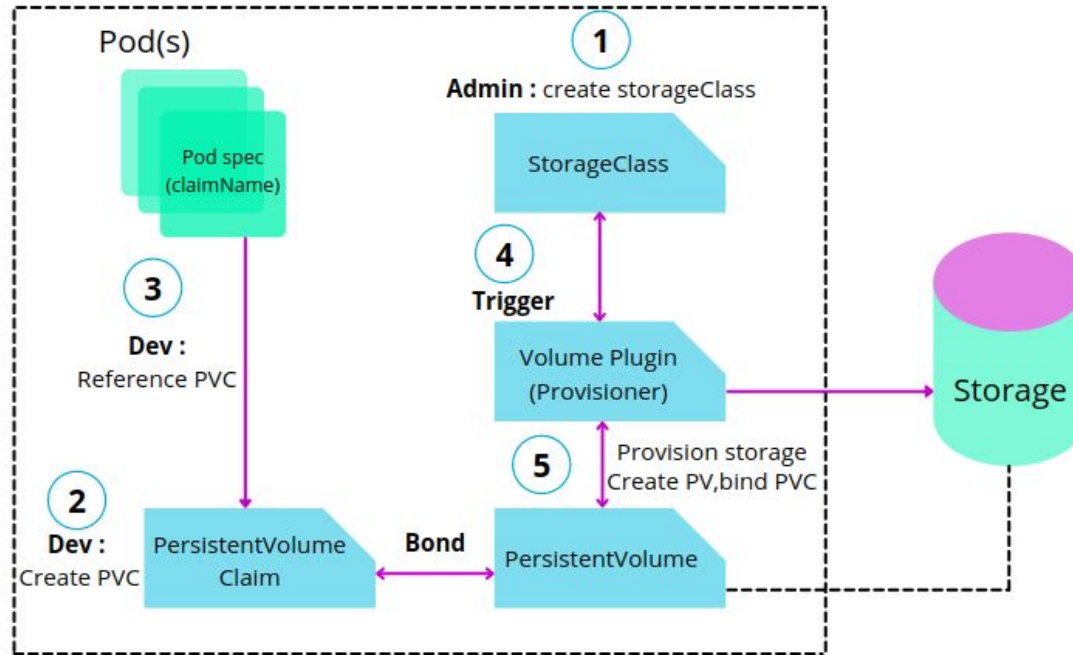
```
1 apiVersion: v1
2 kind: PersistentVolume
3 metadata:
4   name: pv-hostpath
5   annotations:
6     pv.beta.kubernetes.io/gid: "1234"
7   labels:
8     type: local
9 spec:
10  storageClassName: manual
11  capacity:
12    storage: 10Gi
13  accessModes:
14    - ReadWriteOnce
15  hostPath:
16    path: "/mnt/data"
```

```
1 apiVersion: v1
2 kind: PersistentVolumeClaim
3 metadata:
4   name: pvc-test
5 spec:
6   storageClassName: manual
7   accessModes:
8     - ReadWriteOnce
9   resources:
10     requests:
11       storage: 3Gi
```

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: pod-pv-pvc
5 spec:
6   volumes:
7     - name: pv-storage
8       persistentVolumeClaim:
9         claimName: pvc-test
10  containers:
11    - name: task-pv-container
12      image: nginx
13      ports:
14        - containerPort: 80
15          name: "http-server"
16      volumeMounts:
17        - mountPath: "/usr/share/nginx/html"
18          name: pv-storage
```



# Persistent Volume - Dynamic





## Example

```
1 kind: StorageClass
2 apiVersion: storage.k8s.io/v1
3 metadata:
4   name: storageclass-generic
5 provisioner: kubernetes.io/aws-ebs
6 parameters:
7   type: gp2
8   zones: us-east-1a,us-east-1b,us-east-1c
9   iopsPerGB: "10"
10  fsType: ext4
```

```
1 apiVersion: v1
2 kind: PersistentVolumeClaim
3 metadata:
4   name: pvc-dynamic
5   labels:
6     app: nginx
7 spec:
8   storageClassName: storageclass-generic
9   accessModes:
10    - ReadWriteOnce
11   resources:
12     requests:
13       storage: 1Gi
```

```
1 kind: Pod
2 apiVersion: v1
3 metadata:
4   name: pod-volume-dynamic
5   labels:
6     app: nginx
7 spec:
8   containers:
9     - name: nginx
10       image: nginx
11       volumeMounts:
12         - mountPath: "/var/www/html"
13           name: external
14   volumes:
15     - name: external
16       persistentVolumeClaim:
17         claimName: pvc-dynamic
```



# Persistent Volumes

- GCEPersistentDisk
- AWSElasticBlockStore
- AzureFile
- AzureDisk
- CSI
- FC (Fibre Channel)
- FlexVolume
- Flocker
- NFS
- iSCSI
- RBD (Ceph Block Device)
- CephFS
- Cinder (OpenStack block storage)
- Glusterfs
- VsphereVolume
- Quobyte Volumes
- HostPath (Single node testing only – local storage is not supported in any way and WILL NOT WORK in a multi-node cluster)
- Portworx Volumes
- ScaleIO Volumes
- StorageOS



# StatefulSet

## Use Cases

- Stable, unique network identifiers
- Stable, persistent storage
- Ordered, graceful deployment and scaling
- Ordered, automated rolling updates

## Limitations

- No automatic deletion of referenced volumes
- No PODs deletion guarantee when StatefulSet is deleted
- Rolling Updates not consistent always



# StatefulSet - Example

```
1 apiVersion: apps/v1
2 kind: StatefulSet
3 metadata:
4   name: sts-web
5 spec:
6   serviceName: "nginx"
7   replicas: 2
8   selector:
9     matchLabels:
10      app: nginx
11   template:
12     metadata:
13       labels:
14         app: nginx
15     spec:
16       containers:
17         - name: nginx
18           image: k8s.gcr.io/nginx-slim:0.8
19           ports:
20             - containerPort: 80
21               name: web
22           volumeMounts:
23             - name: www
24               mountPath: /usr/share/nginx/html
25   volumeClaimTemplates:
26     - metadata:
27       name: www
28     spec:
29       accessModes: [ "ReadWriteOnce" ]
30       resources:
31         requests:
32           storage: 1Gi
```

- Scale Up
- Scale Up
- Update (image)





# StatefulSet - Example

- Scale Up
- Scale down
- `kubectl set image sts/sts-web nginx=nginx:1.18`
- Staged Update
  - `kubectl patch statefulset sts-web -p '{"spec":{"updateStrategy":{"type":"RollingUpdate","rollingUpdate":{"partition":3}}}}'`
  - `kubectl patch statefulset sts-web --type=json' -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/image", "value":"nginx:1.17"}]'`
  - `kubectl get pod sts-web-1 --template '{{range $i, $c := .spec.containers}}{{ $c.image }}{{end}}'`



# DaemonSet

## Purpose

- To run a copy of a POD on all / some node(s)

## Use Cases

- Storage cluster daemon (gluster, ceph)
- Log Collectors (fluentd, logstash)
- Node Monitoring daemons (Prometheus, Dynatrace, collectd)



# DaemonSet - Example

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      tolerations:
        - key: node-role.kubernetes.io/master
          effect: NoSchedule
      containers:
        - name: fluentd-elasticsearch
          image: quay.io/fluentd_elasticsearch/fluentd:v2.5.2
          resources:
            limits:
              memory: 200Mi
            requests:
              cpu: 100m
              memory: 200Mi
          volumeMounts:
            - name: varlog
              mountPath: /var/log
            - name: varlibdockercontainers
              mountPath: /var/lib/docker/containers
              readOnly: true
      terminationGracePeriodSeconds: 30
      volumes:
        - name: varlog
          hostPath:
            path: /var/log
        - name: varlibdockercontainers
          hostPath:
            path: /var/lib/docker/containers
```



## Exercises

All lab work is available on Google classroom

Join link:

<https://classroom.google.com/c/MzE3MjgxODUzMDUx?cjc=kun7p7g>

---



# Configuration

- Config Maps
  - Secrets
  - Security Contexts
  - Accounts
  - Demo
  - Practicals
-



# ConfigMap

- To store non-confidential key-value pairs
- Can be consumed as env variables, command line args or config files in volume
- To decouple env specific config from images for portability
- Max data 1MB

```
1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: cm-game-demo
5 data:
6   # property-like keys; each key maps to a simple value
7   player_initial_lives: "3"
8   ui_properties_file_name: "user-interface.properties"
9   # file-like keys
10  game.properties: |
11    enemy.types=aliens,monsters
12    player.maximum-lives=5
13  user-interface.properties: |
14    color.good=purple
15    color.bad=yellow
16    allow.textmode=true
```

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: pod-configmap
5 spec:
6   containers:
7     - name: demo
8       image: alpine
9       command: ["sleep", "3600"]
10      env:
11        # Define the environment variable
12        - name: PLAYER_INITIAL_LIVES # Notice that the case is different here
13          valueFrom: # from the key name in the ConfigMap.
14            configMapKeyRef:
15              name: cm-game-demo # The ConfigMap this value comes from.
16              key: player_initial_lives # The key to fetch.
17        - name: UI_PROPERTIES_FILE_NAME
18          valueFrom:
19            configMapKeyRef:
20              name: cm-game-demo
21              key: ui_properties_file_name
22      volumeMounts:
23        - name: config
24          mountPath: "/config"
25          readOnly: true
26  volumes:
27    # You set volumes at the Pod level, then mount them into containers inside that Pod
28    - name: config
29      configMap:
30        # Provide the name of the ConfigMap you want to mount.
31        name: cm-game-demo
32        # An array of keys from the ConfigMap to create as files
33        items:
34          - key: "game.properties"
35            path: "game.properties"
36          - key: "user-interface.properties"
37            path: "user-interface.properties"
```



# Secret

- To manage sensitive info like password, oauthkeys, docker login, ssh keys, tls etc

- Examples

```
kubectl create secret docker-registry secret-tiger-docker \  
  --docker-username=tiger \  
  --docker-password=pass113 \  
  --docker-email=tiger@acme.com
```

```
kubectl create secret tls my-tls-secret \  
  --cert=path/to/cert/file \  
  --key=path/to/key/file
```



# Secret

```
1 apiVersion: v1
2 data:
3   username: YWRtaW4=
4   password: MWYyZDFlMmU2N2Rm
5 kind: Secret
6 metadata:
7   name: pod-secret
8   namespace: default
9   resourceVersion: "164619"
10  uid: cfee02d6-c137-11e5-8d73-42010af00002
11 type: Opaque
```

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: pod-secret
5 spec:
6   containers:
7   - name: mypod
8     image: redis
9     env:
10    - name: SECRET_USERNAME
11      valueFrom:
12        secretKeyRef:
13          name: mysecret
14          key: username
15    - name: SECRET_PASSWORD
16      valueFrom:
17        secretKeyRef:
18          name: mysecret
19          key: password
20    volumeMounts:
21    - name: foo
22      mountPath: "/etc/foo"
23      readOnly: true
24    volumes:
25    - name: foo
26      secret:
27        secretName: pod-secret
28        items:
29        - key: username
30          path: my-group/my-username
```





# Security Context

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: security-context-demo
5 spec:
6   securityContext:
7     runAsUser: 1000
8     runAsGroup: 3000
9     fsGroup: 2000
10  volumes:
11    - name: sec-ctx-vol
12      emptyDir: {}
13  containers:
14    - name: sec-ctx-demo
15      image: busybox
16      command: [ "sh", "-c", "sleep 1h" ]
17      volumeMounts:
18        - name: sec-ctx-vol
19          mountPath: /data/demo
20      securityContext:
21        allowPrivilegeEscalation: false
22        capabilities:
23          add: ["NET_ADMIN", "SYS_TIME"]
24
```



# User Accounts & Service Accounts

- User Accounts
  - User accounts are for humans.
  - User accounts are intended to be global. Names must be unique across all namespaces of a cluster.
- Service Accounts
  - Service accounts are for processes, which run in pods.
  - Service accounts are namespaced.
  - Service account creation is intended to be more lightweight



## Exercises

All lab work is available on Google classroom

Join link:

<https://classroom.google.com/c/MzE3MjgxODUzMDUx?cjc=kun7p7g>

---



# Scheduling

- Node Name
  - Node Selector
  - Affinity (Pod, Node)
  - Taints & Tolerations
  - Demo
  - Practicals
-



# Node Name

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: schedule-node
5 spec:
6   replicas: 1
7   selector:
8     matchLabels:
9       component: schedule-node
10  template:
11    metadata:
12      labels:
13        component: schedule-node
14    spec:
15      nodeName: ip-172-31-102-18.ec2.internal
16      containers:
17        - name: test-app
18          image: brainupgrade/test-app:all-tiers-in-one
19          imagePullPolicy: IfNotPresent
20          ports:
21            - containerPort: 8080
22          resources:
23            requests:
24              cpu: "100m"
25              memory: "250Mi"
```



# Node Selector

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: schedule-nodeselector
5 spec:
6   replicas: 1
7   selector:
8     matchLabels:
9       component: schedule-nodeselector
10  template:
11    metadata:
12      labels:
13        component: schedule-nodeselector
14    spec:
15      nodeSelector:
16        node-role.kubernetes.io/spot-worker: "true"
17      containers:
18        - name: test-app
19          image: nginx
20          imagePullPolicy: IfNotPresent
21          ports:
22            - containerPort: 80
```



# POD - Affinity & Anti Affinity (example 1)

```
14 spec:
15   affinity:
16     podAffinity:
17       requiredDuringSchedulingIgnoredDuringExecution:
18         - labelSelector:
19             matchExpressions:
20               - key: tier
21                 operator: In # NotIn, Exists, NotExists
22                 values:
23                   - cache
24             topologyKey: topology.kubernetes.io/zone
25     podAntiAffinity:
26       preferredDuringSchedulingIgnoredDuringExecution:
27         - weight: 100
28           podAffinityTerm:
29             labelSelector:
30               matchExpressions:
31                 - key: tier
32                   operator: In
33                   values:
34                     - messaging
35             topologyKey: topology.kubernetes.io/hostname
36   containers:
37     - name: pod-affinity-middle-tier
38       image: nginx
39       imagePullPolicy: IfNotPresent
40       ports:
41         - containerPort: 80
```



## POD - Affinity & Anti Affinity (example 2)

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: app-cache-store
5 spec:
6   selector:
7     matchLabels:
8       app: cache-store
9   replicas: 2
10  template:
11    metadata:
12      labels:
13        app: cache-store
14    spec:
15      affinity:
16        podAntiAffinity:
17          requiredDuringSchedulingIgnoredDuringExecution:
18            - labelSelector:
19                matchExpressions:
20                  - key: app
21                    operator: In
22                    values:
23                      - cache-store
24              topologyKey: "kubernetes.io/hostname"
25    containers:
26      - name: redis-server
27        image: redis:3.2-alpine
```

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: app-web-server
5 spec:
6   selector:
7     matchLabels:
8       app: web-server
9   replicas: 2
10  template:
11    metadata:
12      labels:
13        app: web-server
14    spec:
15      affinity:
16        podAntiAffinity:
17          requiredDuringSchedulingIgnoredDuringExecution:
18            - labelSelector:
19                matchExpressions:
20                  - key: app
21                    operator: In
22                    values:
23                      - web-server
24              topologyKey: "kubernetes.io/hostname"
25        podAffinity:
26          requiredDuringSchedulingIgnoredDuringExecution:
27            - labelSelector:
28                matchExpressions:
29                  - key: app
30                    operator: In
31                    values:
32                      - cache-store
33              topologyKey: "kubernetes.io/hostname"
34    containers:
35      - name: web-app
36        image: nginx:1.16-alpine
```





# Node - Affinity

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: node-affinity
5 spec:
6   replicas: 1
7   selector:
8     matchLabels:
9       tier: middle-tier
10  template:
11    metadata:
12      labels:
13        tier: middle-tier
14    spec:
15      affinity:
16        nodeAffinity:
17          requiredDuringSchedulingIgnoredDuringExecution:
18            nodeSelectorTerms:
19              - matchExpressions:
20                - key: topology.kubernetes.io/zone
21                  operator: In #In, NotIn, Exists, DoesNotExist, Gt, Lt
22                  values:
23                    - us-east-1b
24                    - us-east-1c
25          preferredDuringSchedulingIgnoredDuringExecution:
26            - weight: 1
27              preference:
28                matchExpressions:
29                  - key: kops.k8s.io/instancegroup
30                    operator: In #NotIn, DoesNotExist for AntiAffinity
31                    values:
32                      - spotnodes
33      containers:
34        - name: nginx
35          image: nginx
36          imagePullPolicy: IfNotPresent
37          ports:
38            - containerPort: 80
```



# Taints & Tolerations

- Objective

To steer pods away from nodes or evict pods that should not be running on particular nodes

- Use Cases

- Dedicated Nodes
- Nodes with special hardware

- Notes

- Taints - Node (to reject pods that don't tolerate taints)
- Tolerations - Pod (to allow pods to schedule onto nodes matching taints)



# Taints

```
lab@rajesh-Gazelle:~$ kubectl get nodes -o json | jq ".items[]|{name:.metadata.name, taints:.spec.taints}"
{
  "name": "ip-172-31-102-18.ec2.internal",
  "taints": null
}
{
  "name": "ip-172-31-34-27.ec2.internal",
  "taints": [
    {
      "effect": "NoSchedule",
      "key": "node-role.kubernetes.io/master"
    }
  ]
}
{
  "name": "ip-172-31-87-231.ec2.internal",
  "taints": null
}
```

Effect types: NoSchedule or PreferNoSchedule or NoExecute



# Taints

- Examples (Taints)

- To apply

`kubecttl taint nodes qanode release=qa:NoSchedule`

`kubecttl taint nodes devnode release=dev:PreferNoSchedule`

`kubecttl taint nodes prodnode release=prod:NoExecute`

- To remove

`kubecttl taint nodes node-name key=value:Effect-`



# Tolerations

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   labels:
5     release: qa
6   name: tolerations-release-qa
7 spec:
8   containers:
9     - image: nginx
10     name: tolerations-unreachable
11   dnsPolicy: ClusterFirst
12   restartPolicy: Always
13   tolerations:
14     - key: "release"
15       operator: "Exists"
16       value: "qa"
17       effect: "NoSchedule"
```



## Exercises

All lab work is available on Google classroom

Join link:

<https://classroom.google.com/c/MzE3MjgxODUzMDUx?cjc=kun7p7g>

---



## Services

- Cluster IP
  - Node Port
  - Load Balancer
  - Demo
  - Practicals
-



# Service Types

- Cluster IP
  - Service exposed on cluster internal IP
  - Reachable only within cluster
- Node Port
  - Exposed on each Node IP at static port
- Load Balancer
  - Exposed through external cloud load balancer





# Service

- An abstract way to expose an application running on pod as network service.
- Frontends and backends of application can connect without worrying about POD IPs
- Uses session affinity while connecting to backend PODs

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   labels:
5     app: nginx
6   name: nginx
7 spec:
8   ports:
9     - port: 80
10     protocol: TCP
11     targetPort: 80
12   selector:
13     app: nginx
```



# Service - External IP

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 9376
  externalIPs:
    - 80.11.12.10
```

## Use Cases:

- External DB Cluster in production
- To point a service in another namespace / cluster



# Ingress

- Provides load balancing, SSL Termination and Name based virtual hosting
- Provides externally reachable URLs to Services
- Used for HTTP / HTTPS protocols

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```



# Examples

- Nginx deployment, Expose as service and publish (Demo)
  - Spring Boot with DB (walkthrough)
  - API Services rollout (walkthrough)
-



## Exercises

All lab work is available on Google classroom

Join link:

<https://classroom.google.com/c/MzE3MjgxODUzMDUx?cjc=kun7p7g>

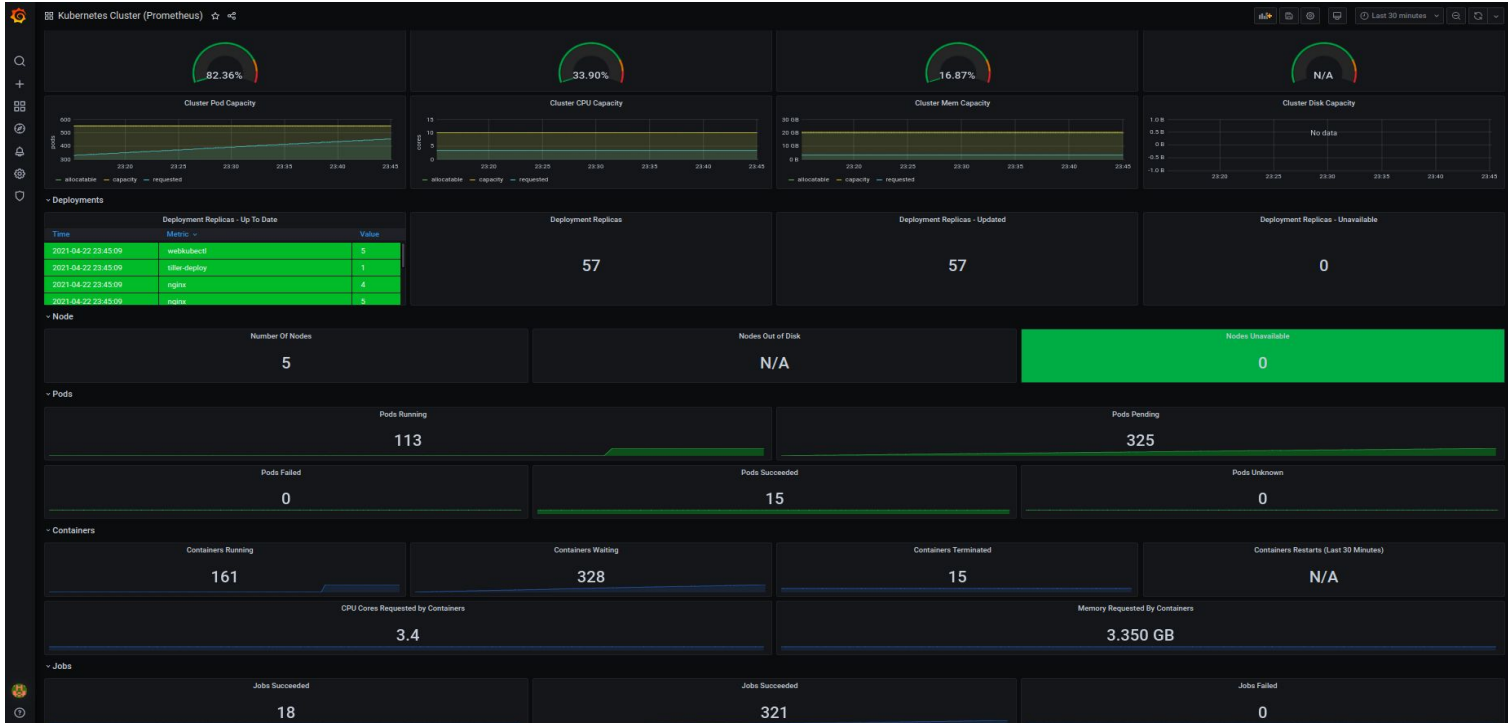
---



## Monitoring & Misc

- Kubernetes Dashboard
  - Port Forwarding
  - Introspection & Debugging
  - Container Logging
  - Best Practices
  - Demo
  - Practicals
-

# Monitoring Dashboard (Prometheus-Grafana)





# Port Forwarding

```
kubectl port-forward <pod> 7000:6379
```

```
kubectl port-forward <deployment> 7000:6379
```

```
kubectl port-forward <svc> 7000:6379
```

To access LoadBalancer service on localhost

```
minikube tunnel
```





# Introspection & debugging

```
kubectl get pods
kubectl get pod <pod-name> -o yaml
kubectl describe <pod-name>
kubectl describe <pod-name> -o yaml
kubectl get events
kubectl get events --namespace=my-namespace  (--all-namespaces)
kubectl get nodes
kubectl get node <node-name>
kubectl get node <node-name> -o yaml
kubectl describe node <node-name>
kubectl describe node <node-name> -o yaml
kubectl logs -f pod <pod-name>
```



# Shell to running container

```
rajesh@rajesh-Gazelle:~/git/kubernetes/debugging/shell$ kubectl apply -f shell-demo.yaml
```

```
kubectl get pod shell-demo
```

```
kubectl exec -it shell-demo -- /bin/bash
```

```
root@shell-demo:/# ls /
```

```
root@shell-demo:/# echo Hello shell demo > /usr/share/nginx/html/index.html
```

```
root@shell-demo:/# apt-get update
```

```
root@shell-demo:/# apt-get install curl
```

```
root@shell-demo:/# curl localhost
```

```
kubectl exec shell-demo env
```

```
kubectl exec -it my-pod --container main-app -- /bin/bash
```



# Best Practices

- Configuration - specify latest stable API version
- Keep config files in version control before pushing to cluster
- Prefer YAML over JSON
- Group related objects into one file whenever it makes more sense
- Don't specify default values unnecessarily
- Put Object descriptions as part of annotations
- Don't use naked PODs
- Avoid using hostPort for POD
- Use labels effectively
- Use image tag instead of using latest as the default
- Use kubectl run and expose to launch single container deployments & services



# Tips for CKAD & CKA (CNCF)

## Preparation

- Improve typing speed (Ref: <https://typing.com>)
- Improve VI skills (Ref: <https://www.youtube.com/watch?v=5r6yzFEXajQ>)
- Once concepts are clear then practice practice & practice (Get your linux os and install minikube or microk8s to play with the cluster locally)
- Question Difficulty levels: Mix of easy, intermediate and difficult (however most of the Qs are intermediate or less difficult)

## Exam day

- Adequate sleep previous day
- Choose quiet place for exam
- Ensure your body is hydrated enough
- Keep yourself calm & maintain it throughout exam period

## Resources

- Refer Kubernetes Documentation (<https://kubernetes.io/docs/home/>)
- DZone articles on Kubernetes: <https://dzone.com/users/4519738/rajeshg007.html>
- Explanation with code: <https://github.com/brainupgrade-in/kubernetes>
- Exam Syllabus: <https://github.com/cncf/curriculum>



## Recap

- Review
- Q & A



## Misc: Spring boot app monitoring

```
- name: management.endpoints.web.exposure.include
  value: "*"
- name: spring.application.name
  value: fleetLytics
- name: management.server.port
  value: "8888"
- name: management.metrics.web.server.request.autotime.enabled
  value: "true"
- name: management.metrics.tags.application
  value: fleetLytics
```

```
annotations:
  prometheus.io/scrape: "true"
  prometheus.io/port: "8888"
  prometheus.io/path: /actuator/prometheus
```

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

Ref: Microservices observability on Kubernetes

<https://dzone.com/articles/microservices-with-observability-on-kubernetes>



# **Thank You for your active participation!**

[info@brainupgrade.in](mailto:info@brainupgrade.in)

<https://www.facebook.com/brainupgrade.in>

<https://www.linkedin.com/company/brains-upgrade/>

<https://www.linkedin.com/in/rajesh-gheware/>

---