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

Brief about me

Rajesh Gheware

Author | TOGAF EA | CKA | Trainer (Cloud - Kubernetes & Docker) | Entrepreneur | Technology Advisor

Bangalore Urban, Karnataka, India · 500+ connections ·

UniGPS Solutions

Indian Institute of Technology Madras

My Certifications
CKA: Certified Kubernetes Administrator (2020)
TOGAF Certified Enterprise Architect (2015)
Spring – Core (3.2) Certified Professional (96% score – Aug 2013)
OMG UML (FL) - 2010
IBM OOAD-UML / Brainbench UML Certification (http://www.brainbench.com/transcript.jsp?pid=3730016) - 2005
Brainbench JAVA-EJB 2.0 Certification – 2001

My Interest Areas:
Kubernetes, Docker, AWS, IoT, Enterprise Architecture, Spring Framework, Java, UML

Some of my articles on Dzone.com as reference :
https://dzone.com/articles/microservices-with-observability-on-kubernetes
https://dzone.com/articles/scalable-jenkins-on-kubernetes-cluster-amp-pipelin
https://dzone.com/articles/expose-your-app-to-the-internet-using-ingress-cont

Misc:
ICC & IE and IEMS published my research work on Production Planning and Control (2001)
My articles in Open source magazine - http://opensourceforu.com/author/rajesh-gheware

Experience (20+ years in IT - Software Engineering)

UniGPS Solutions
3 yrs 9 mos

Founder & CTO
Dec 2018 – Present · 2 yrs 5 mos
Bengaluru Area, India

Managing Partner, CTO
Aug 2017 – Present · 3 yrs 9 mos
Bangalore

https://unigps.in

VP Technology
JPMorgan Chase & Co.
Jun 2015 – Aug 2017 · 2 yrs 3 mos
Bangalore Urban, Karnataka, India

Lead Solution Architect / Delivery Manager
Deutsche Bank Group
Dec 2009 – Jun 2015 · 5 yrs 7 mos

Part of OTC Derivatives since 2010

Technical Project Manager / Application Architect
Headstrong India Pvt Ltd
Jun 2006 – Dec 2009 · 3 yrs 7 mos

Kubernetes

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

Kubernetes Core

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

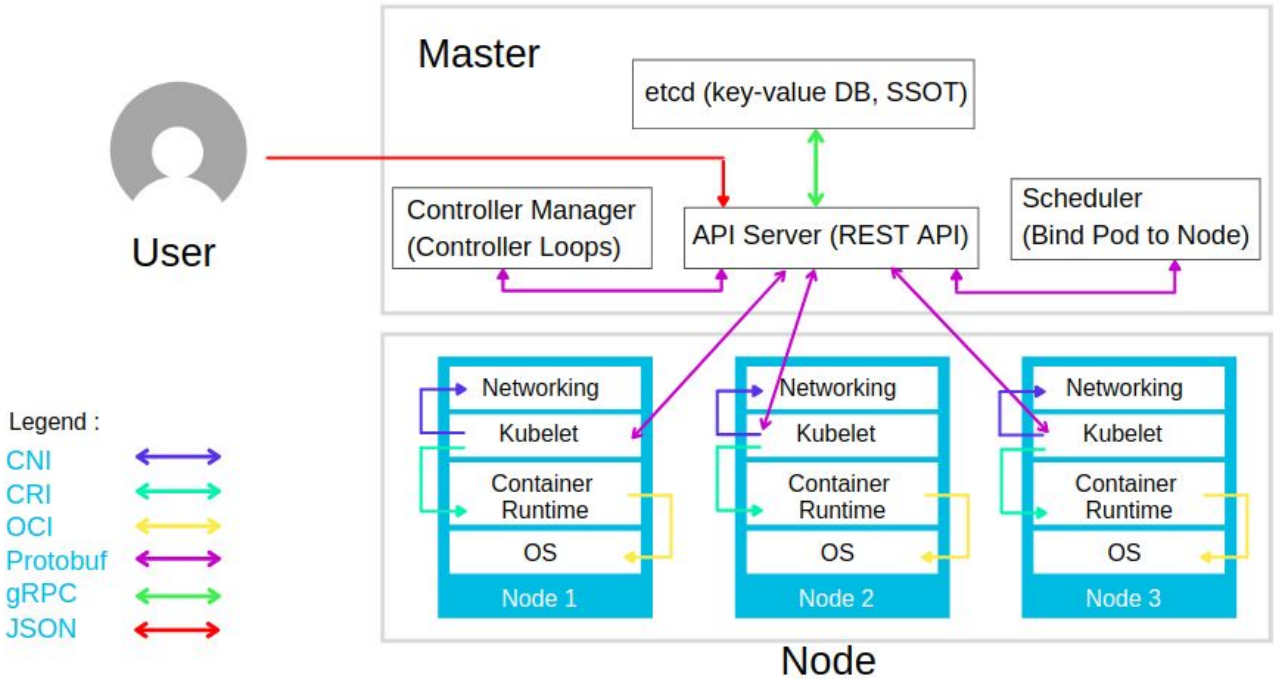
What is / why Kubernetes

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

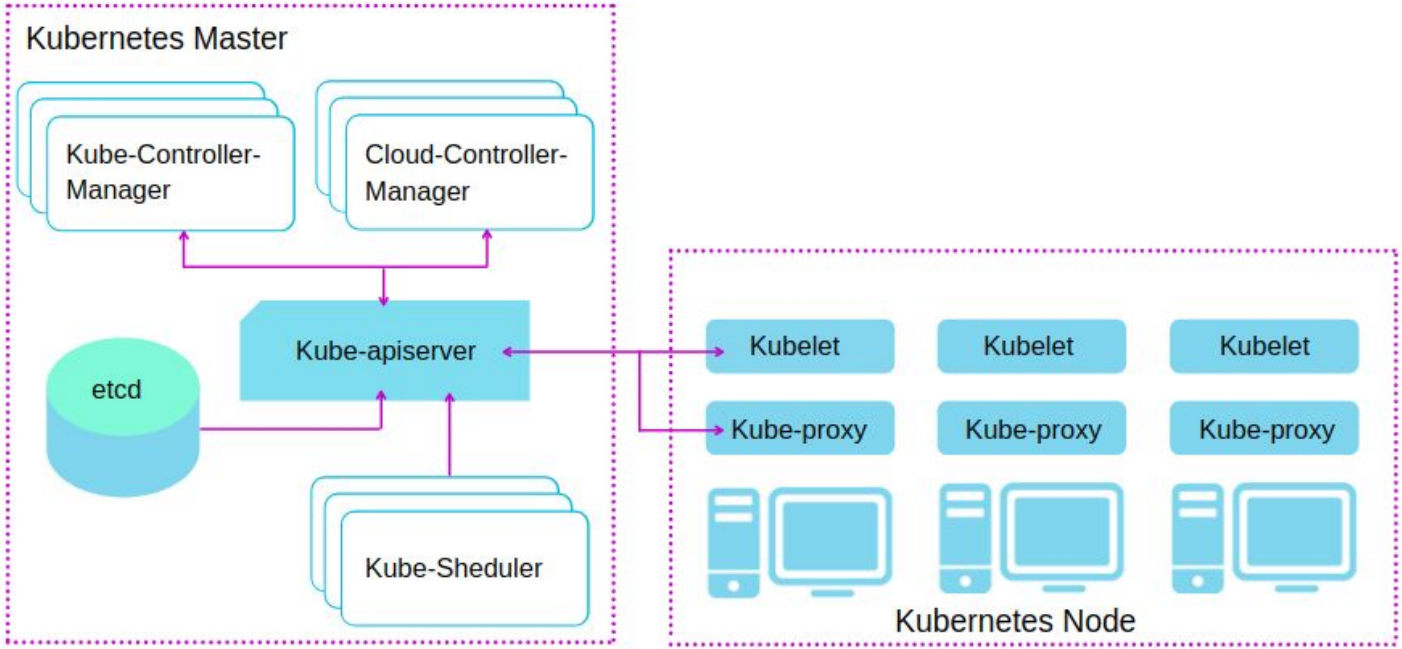
Why?

- 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

Master Components - Kube Controller

- 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
  - Updates Node objects when new servers are created in the cloud, including labels, hostname, network addresses, verify health etc
- Route Controller
  - For setting up routes in the underlying cloud infrastructure so that containers on different nodes can communicate with each other
- 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

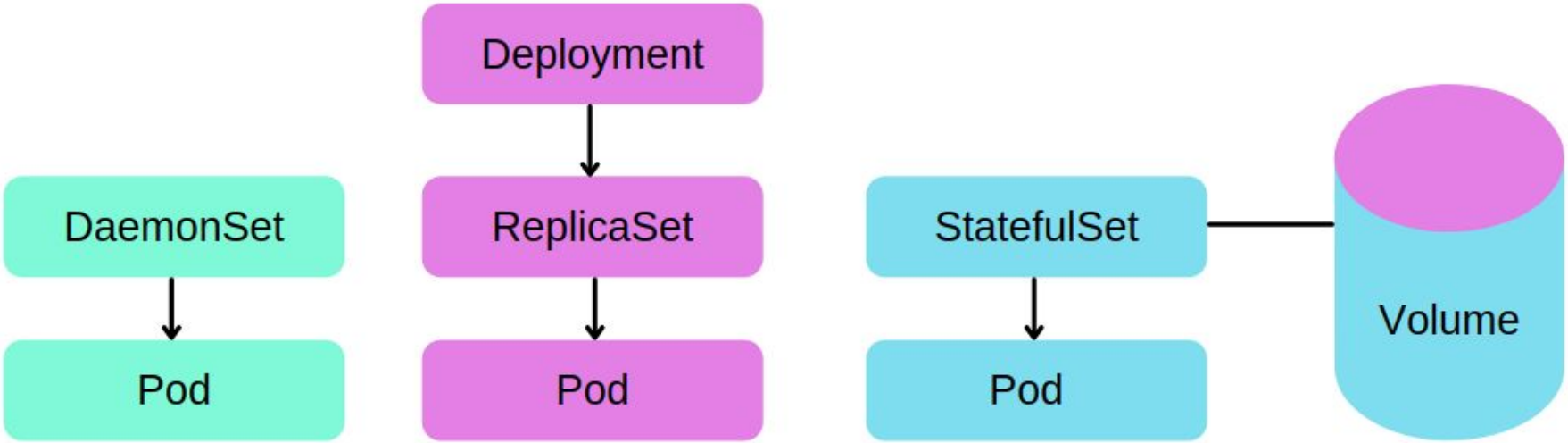
- KUBE-PROXY
  - Network proxy that runs on every node in cluster
  - Maintains network rules on nodes
  - Uses OS packet filtering layer else forwards traffic itself
- KUBELET
  - Runs on every node
  - Ensures containers are running & healthy in PODs
  - Doesn’t manage container not created by K8S

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

Objects


Deployment
DaemonSet
ReplicaSet
StatefulSet
Volume
Pod
Pod
Pod

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

Please refer the google classwork link given in the chat message

And do all the lab work as per the instructions noted in the classwork assignments

POD

- Overview
- Lifecycle
- Init Containers
- 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

training > kubernetes > lab > pod > ! nginx-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
  dnsPolicy: ClusterFirst
  restartPolicy: Always
```

training > kubernetes > lab > kubernetes-core > ! 01-monolith-pod.yaml

```yaml
# Monolith containing all app components
# including database
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: monolith
  name: monolith
spec:
  containers:
  - image: brainupgrade/weather:monolith
    name: monolith
  restartPolicy: Always
```

Example

training > kubernetes > lab > pod > ! pod-busybox.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: pod-busybox
  name: pod-busybox
spec:
  containers:
  - command:
    - sh
    - -c
    - echo App is running! && sleep 30
    image: busybox
    name: pod-busybox
  restartPolicy: Never
```

POD - Lifecyle ...

- Lifecycle Phases
  - 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)
- Probes
  - Startup, Readiness, Liveness
- Lifecycle hooks

Phase - Pending

training > kubernetes > lab > pod > ! lifecycle-phase-pending.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources:
      requests:
        cpu: "6000m"
        memory: "100Gi"
  dnsPolicy: ClusterFirst
  restartPolicy: Never
```

Phase - Running

training > kubernetes > lab > pod > ! lifecycle-phase-running.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox
  name: busybox
spec:
  containers:
  - command:
    - ping
    - google.com
    image: busybox
    name: busybox
  dnsPolicy: ClusterFirst
  restartPolicy: Always
```

Phase - Succeeded

training > kubernetes > lab > pod > ! lifecycle-phase-succeeded.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox
  name: busybox
spec:
  containers:
  - image: busybox
    name: busybox
  dnsPolicy: ClusterFirst
  restartPolicy: Never
```

Probes

- Types
  - Startup
  - Readiness
  - Liveness
- Methods
  - Http
  - Tcp
  - Command

- Settings
  - initialDelaySeconds
  - periodSeconds
  - timeoutSeconds
  - successThreshold
  - failureThreshold

- Http
  - Host
  - Scheme
  - Path
  - Port
  - Headers

Probe - Liveness - Exec

training > kubernetes > lab > pod > ! probe-liveness-exec.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: probe-liveness-exec
spec:
  containers:
  - name: probe-liveness-exec
    image: k8s.gcr.io/busybox
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
    livenessProbe:
      exec:
        command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
```

Probe - Liveness - http

training > kubernetes > lab > pod > ! probe-liveness-http.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: probe-liveness-http
spec:
  containers:
  - name: probe-liveness-http
    image: k8s.gcr.io/liveness
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
```

Ref https://github.com/kubernetes/kubernetes/blob/master/test/images/agnhost/liveness/server.go

Probe - Liveness - readiness - tcp

training > kubernetes > lab > pod > ! probe-liveness-readiness-tcp.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: probe-liveness-readiness-tcp
spec:
  containers:
  - name: probe-liveness-readiness-tcp
    image: k8s.gcr.io/goproxy:0.1
    ports:
    - containerPort: 8080
    readinessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 10
    livenessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 15
      periodSeconds: 20
```

Probe - Liveness - startup - http

training > kubernetes > lab > pod > ! probe-liveness-startup-http.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: probe-liveness-startup-http
spec:
  containers:
  - name: probe-liveness-startup-http
    image: k8s.gcr.io/liveness
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
      failureThreshold: 1
      periodSeconds: 10
    startupProbe:
      httpGet:
        path: /healthz
        port: 8080
      failureThreshold: 30
      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

Example

training > kubernetes > lab > pod > ! init-containers.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: init-containers
spec:
  containers:
  - name: main-container
    image: busybox:1.28
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-service
    image: busybox:1.28
    command:
    - sh
    - -c
    - until nslookup myservice.$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace).svc.cluster.local; do echo waiting for myservice; sleep 2; done
  - name: init-mydb
    image: busybox:1.28
    command: ['sh', '-c', "until nslookup mydb.$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace).svc.cluster.local; do echo waiting for mydb; sleep 2; done"]
```

Example

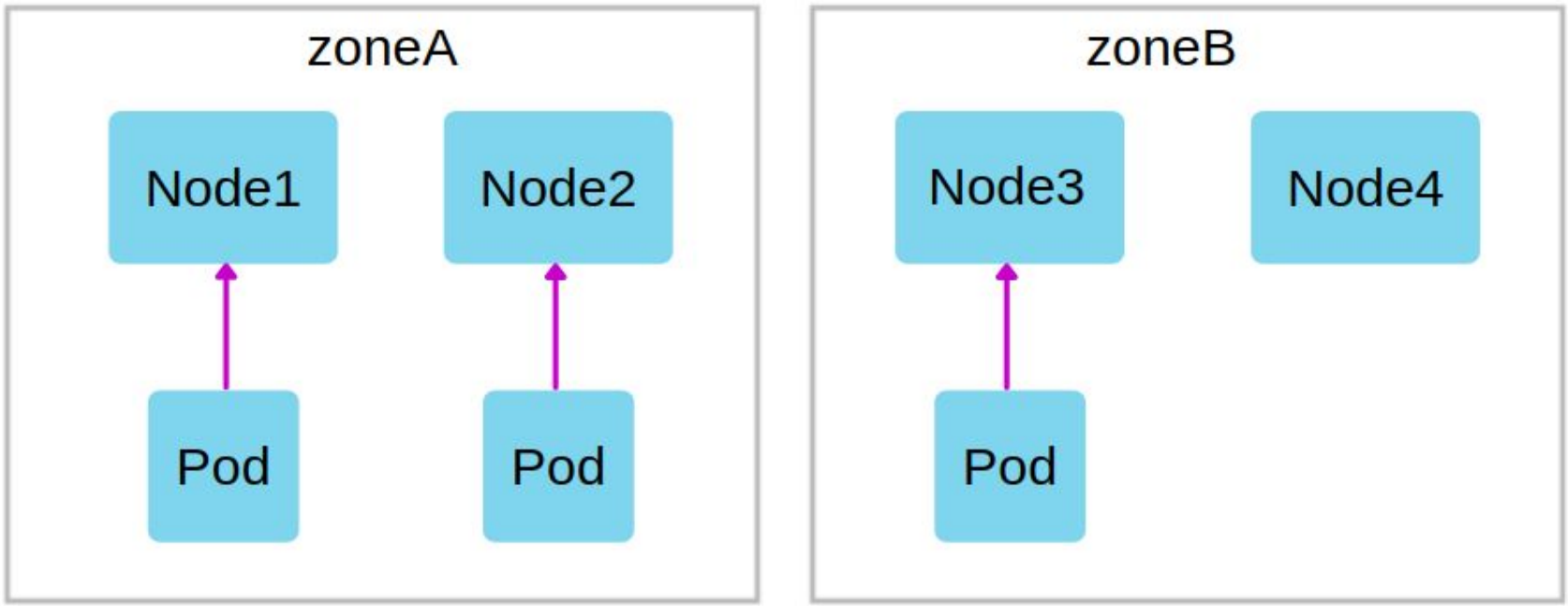
training > app > weather > ! 07b-apm-weather-front.yaml

```yaml
# microservices components 1 - front
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: weather
  name: weather-front
spec:
  replicas: 1
  selector:
    matchLabels:
      app: weather-front
  template:
    metadata:
      labels:
        app: weather-front
    spec:
      volumes:
      - name: elastic-apm-agent
        emptyDir: {}
      initContainers:
      - name: elastic-java-agent
        image: docker.elastic.co/observability/apm-agent-java:1.12.0
        volumeMounts:
        - mountPath: /elastic/apm/agent
          name: elastic-apm-agent
        command: ['cp', '-v', '/usr/agent/elastic-apm-agent.jar', '/elastic/apm/agent']
      containers:
        - image: brainupgrade/weather:microservices-front
          imagePullPolicy: Always
          name: weather-front
          volumeMounts:
          - mountPath: /elastic/apm/agent
            name: elastic-apm-agent
          env:
            - name: spring.application.name
              value: weather-front
            - name: spring.datasource.url
              value: jdbc:mariadb://weather-db:3306/weather
            - name: spring.datasource.username
              value: weather
            - name: spring.datasource.password
              value: weather
            - name: weatherServiceURL
              value: http://weather-services.weather.svc.cluster.local
            - name: ELASTIC_APM_SERVER_URL
              value: "http://apm-server.elasticsearch.svc.cluster.local:8200"
            - name: ELASTIC_APM_SERVICE_NAME
              value: "weather-front"
            - name: ELASTIC_APM_APPLICATION_PACKAGES
              value: "in.brainupgrade"
            - name: ELASTIC_APM_ENVIRONMENT
              value: prod
            - name: ELASTIC_APM_LOG_LEVEL
              value: DEBUG
            - name: JAVA_TOOL_OPTIONS
              value: -javaagent:/elastic/apm/agent/elastic-apm-agent.jar
          ports:
            - containerPort: 8080
```

POD - Topology Spread

- Objectives
  - To control how Pods are spread across 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

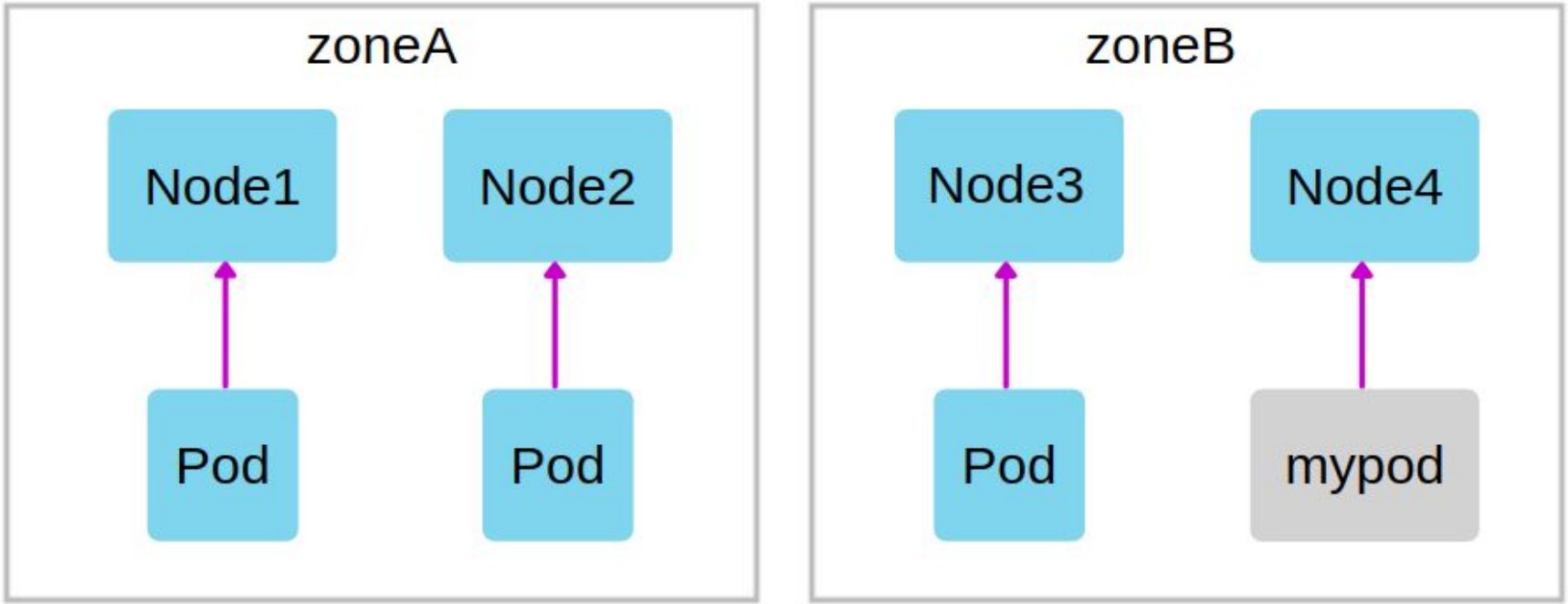


POD - Topology Spread

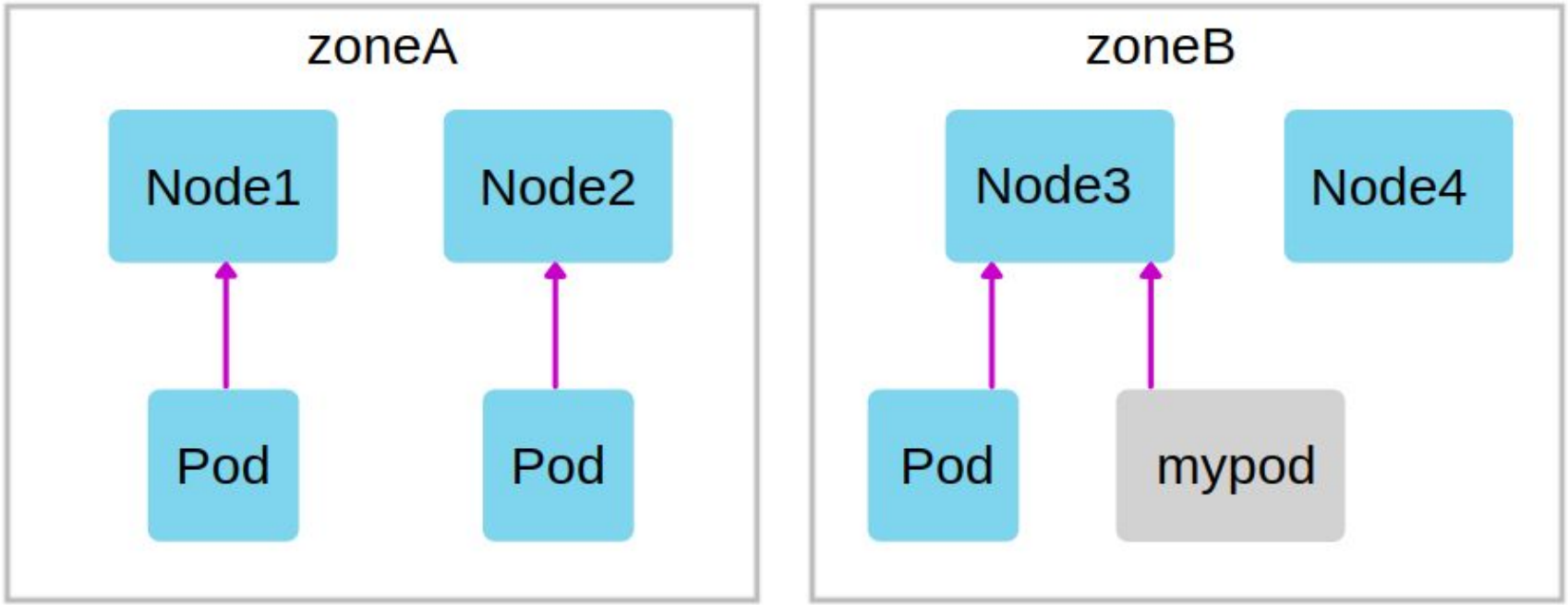
dockerk8s > kubernetes > lab > scheduling > topology > ! one-pod-per-zone.yaml >

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: one-pod-per-zone
spec:
  replicas: 1
  selector:
    matchLabels:
      app: one-pod-per-zone
  template:
    metadata:
      labels:
        app: one-pod-per-zone
    spec:
      nodeSelector:
        kubernetes.io/role: node
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: topology.kubernetes.io/zone
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: one-pod-per-zone
      containers:
      - image: gcr.io/google-samples/hello-app:1.0
        name: hello-app
```

POD - Topology Spread



POD - Topology Spread



Topology - Multiple Constraints

dockerk8s > kubernetes > lab > scheduling > topology > ! one-pod-per-node-zone.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: one-pod-per-node-zone
spec:
  replicas: 1
  selector:
    matchLabels:
      app: one-pod-per-node-zone
  template:
    metadata:
      labels:
        app: one-pod-per-node-zone
    spec:
      nodeSelector:
        kubernetes.io/role: node
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: topology.kubernetes.io/zone
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: one-pod-per-node-zone
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: one-pod-per-node-zone
      containers:
      - image: gcr.io/google-samples/hello-app:1.0
        name: hello-app
```

Exercises

Please refer the google classwork link given in the chat message

And do all the lab work as per the instructions noted in the classwork assignments

Pod - Advanced

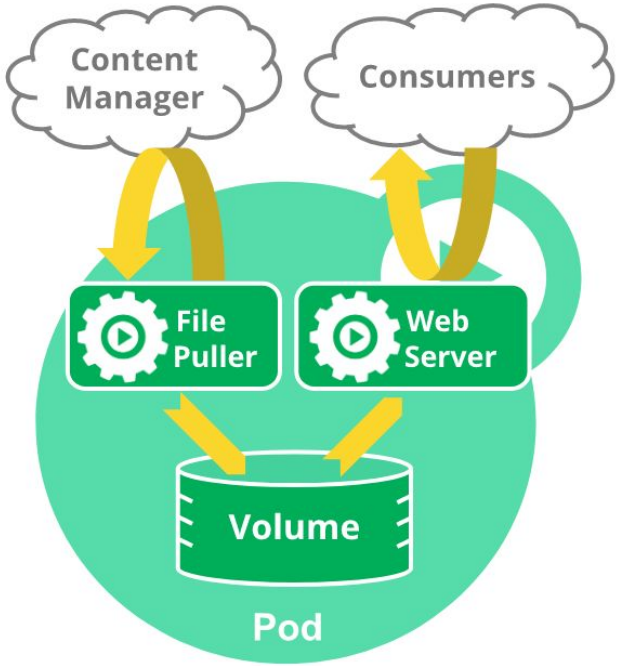
- Patterns
  - Sidecar
  - Adapter
  - Ambassador
- Jobs
- Cron Jobs
- Labels, Selectors, Annotations
- Demo
- Practicals

Patterns - POD

- To extend the functionality of the existing container
- To have helper process enhancing work of the existing container

- Types
  - Sidecar - To export logs, generate / pull data etc.
  - Ambassador - To proxy connection
  - Adapter - To standardise and normalize output

Pattern - Sidecar

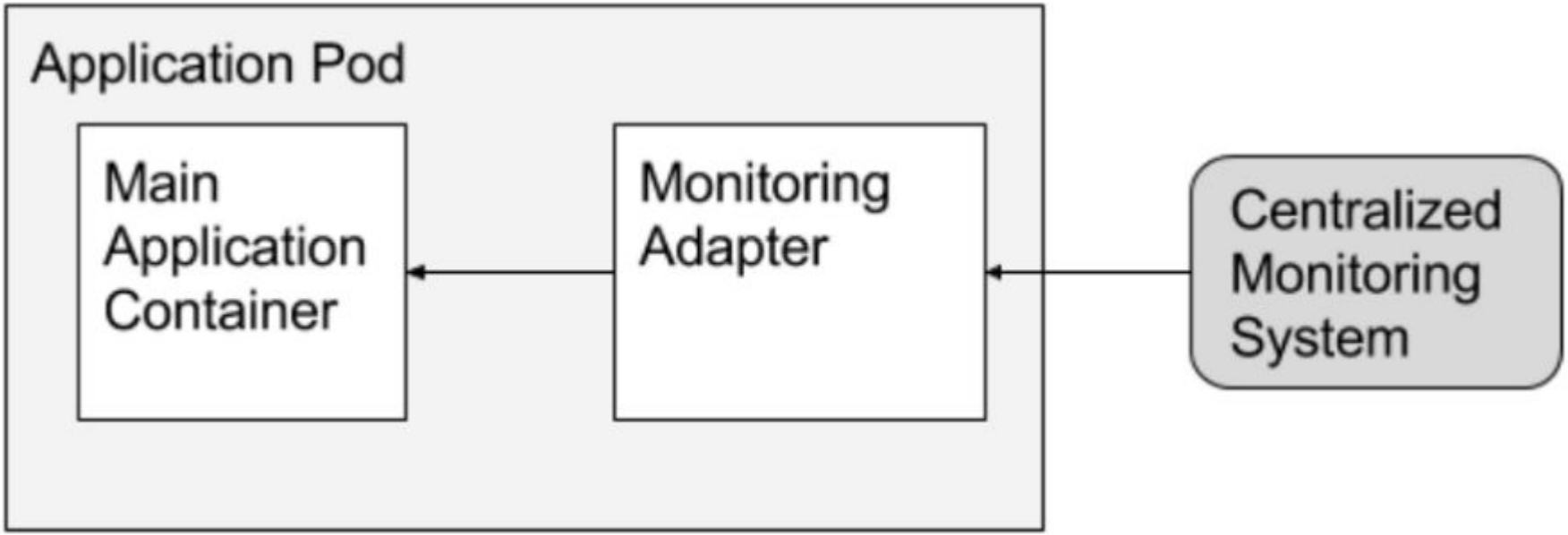


Pattern - Sidecar

dockerk8s > kubernetes > lab > pod-advanced > pod-multi-container > ! sidecar.yaml > apiVersion

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sidecar
spec:
  imagePullSecrets:
    - name: regcred
  volumes:
  - name: shared-logs
    emptyDir: {}
  containers:
  - name: main-container
    image: busybox
    command: ["/bin/sh"]
    args: ["-c", "while true; do ping -c4 google.com >> /var/log/index-input.html; sleep 10;done"]
    resources:
      limits:
        cpu: "10m"
        memory: "10Mi"
    volumeMounts:
    - name: shared-logs
      mountPath: /var/log
  - name: sidecar-container
    image: busybox # splunk adapter
    resources:
      limits:
        cpu: "10m"
        memory: "10Mi"
    command: ["/bin/sh"]
    args: ["-c", "while true; do cat /var/log/index-input.html > /tmp/index-output.html; sleep 60;done"]
    volumeMounts:
    - name: shared-logs
      mountPath: /var/log
```

Pattern - Adapter

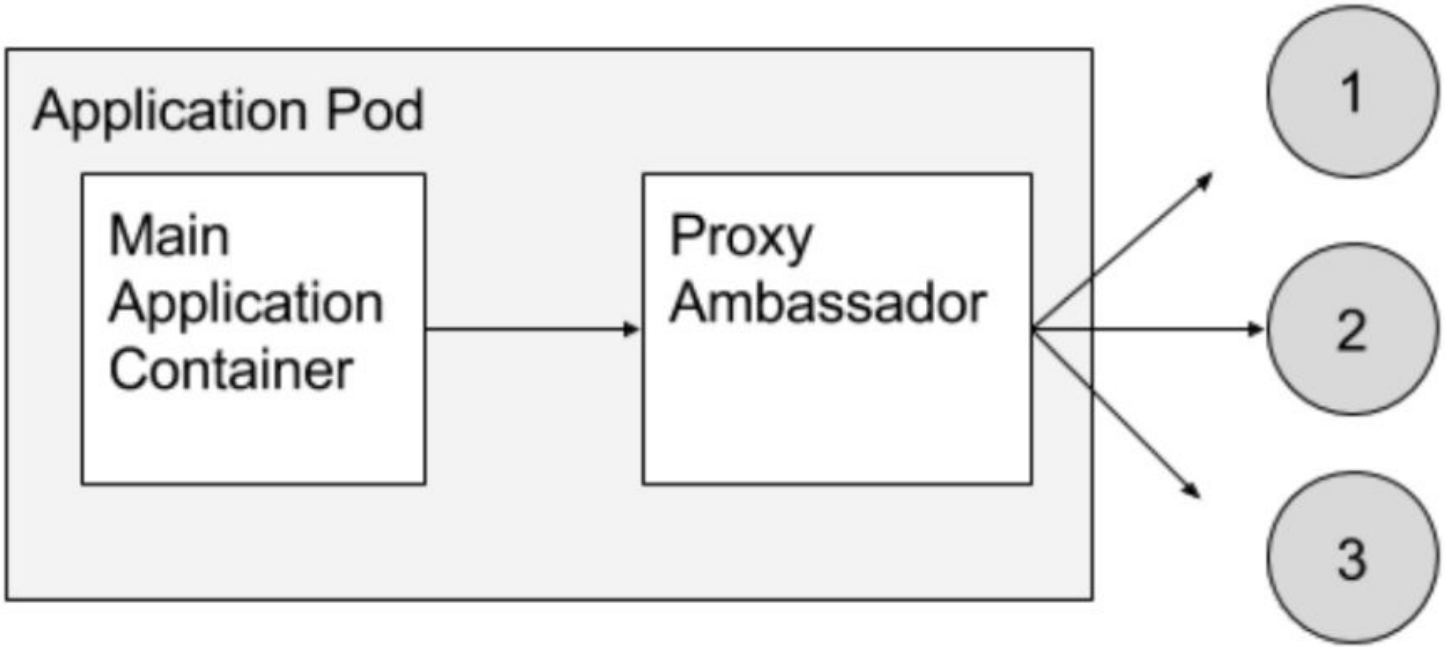


Pattern - Adapter

dockerk8s > kubernetes > lab > pod-advanced > pod-multi-container > ! adapter.yaml > [ ] -o -E '\\d+\\w' | head -1 >> /var/log/status.txt) && (cat /var/log/top.txt | head -3 | tail -1 | grep -o -E '\\d+%' | head -1

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: adapter
spec:
  imagePullSecrets:
    - name: regcred
  volumes:
  - name: shared-logs
    emptyDir: {}
  containers:
  - name: main-container
    image: alpine
    command: ["/bin/sh"]
    args: ["-c", "while true; do date > /var/log/top.txt && top -n 1 -b >> /var/log/top.txt; sleep 10;done"]
    volumeMounts:
    - name: shared-logs
      mountPath: /var/log
  - name: adapter-container
    image: alpine  # logstash message formatter
    command: ["/bin/sh"]
    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
 -o -E '\\d+\\w' | head -1 >> /var/log/status.txt) && (cat /var/log/top.txt | head -3 | tail -1 | grep
-o -E '\\d+%' | head -1 >> /var/log/status.txt); sleep 5; done"]
    volumeMounts:
    - name: shared-logs
      mountPath: /var/log
```

Pattern - Ambassador



Pattern - Ambassador

training > kubernetes > lab > pod-advanced > pod-multi-container > ! ambassador.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: ambassador-nginx-config
data:
  nginx.conf: |
    worker_processes 4;
    events { worker_connections 1024; }
    http {
        server {
            listen 80;
            location / {
                proxy_pass https://www.brainupgrade.in;
            }
        }
    }
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: multi-pod-ambassador
spec:
  imagePullSecrets:
    - name:  regcred
  containers:
  - name: main-app
    image: busybox
    imagePullPolicy: IfNotPresent
    command: ["/bin/sh"]
    args: ["-c","while true;do wget -O /tmp/app.txt localhost ;sleep 30;done"]
  - name: ambassador
    image: nginx
    imagePullPolicy: IfNotPresent
    ports:
    - containerPort: 80
    volumeMounts:
      - name: nginx-config
        mountPath: /etc/nginx
  volumes:
    - name: nginx-config
      configMap:
        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
  - Parallel job
  - Fixed completion count job
  - Work queue job

Job

dockerk8s > kubernetes > lab > pod-advanced > ! job.yaml > ...

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: migration-task
spec:
  backoffLimit: 5
  activeDeadlineSeconds: 100
  parallelism: 10
  template:
    spec:
      containers:
      - name: cron-job
        image: busybox
        args:
        - /bin/sh
        - -c
        - date; echo Migrating data to reporting server...;sleep 1;
      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

training > kubernetes > lab > pod-advanced > ! cron-job.yaml

```yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: cron-job
spec:
  schedule: "*/10 * * * *"
  startingDeadlineSeconds: 60
  concurrencyPolicy: "Allow"
  jobTemplate:
    spec:
      template:
        spec:
          imagePullSecrets:
            - name:  regcred
          containers:
          - name: cron-job
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Migrating data to reporting server...;sleep 60;
          restartPolicy: OnFailure
```

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

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-labels
  labels:
    environment: production
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - 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

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: selector-pod-node
spec:
  containers:
    - name: cuda-test
      image: "k8s.gcr.io/cuda-vector-add:v0.1"
      resources:
        limits:
          nvidia.com/gpu: 1
  nodeSelector:
    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

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: annotations-pod
  annotations:
    imageregistry: "https://hub.docker.com/"
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

Annotations

training > app > weather > ! 08c-metrics-weather-services.yaml

```yaml
# microservices components 2 - api services
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: weather-services
  name: weather-services
  namespace: weather
spec:
  replicas: 1
  selector:
    matchLabels:
      app: weather-services
  template:
    metadata:
      labels:
        app: weather-services
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "8888"
        prometheus.io/path: /actuator/prometheus
    spec:
```

Scalability

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

Exercises - Reference

Please refer the google classwork link given in the chat message

And do all the lab work as per the instructions noted in the classwork assignments

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

dockerk8s > kubernetes > lab > scalability > ! scalability.yaml > ...

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: google
  name: google
spec:
  replicas: 1
  selector:
    matchLabels:
      app: google
  template:
    metadata:
      labels:
        app: google
    spec:
      imagePullSecrets:
        - name: regcred
      containers:
      - image: gcr.io/google-samples/hello-app:1.0
        name: google
        ports:
        - containerPort: 8080
        resources:
          limits:
            cpu: "10m"
            memory: "50Mi"
```

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=10 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

Please refer the google classwork link given in the chat message

And do all the lab work as per the instructions noted in the classwork assignments

Networking

- CNI
- CNI Flow
- Routing:
  - Within Host, Across Hosts
- Calico - Architecture
- CoreDNS
- Benchmarks / Comparison

Networking

```
Fetched 73.5 MB in 3s (21.3 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 64018 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.5-2_amd64.deb ...
Unpacking conntrack (1:1.4.5-2) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.19.0-00_amd64.deb ...
Unpacking cri-tools (1.19.0-00) ...
Selecting previously unselected package ebtables.
Preparing to unpack .../2-ebtables_2.0.11-3build1_amd64.deb ...
Unpacking ebtables (2.0.11-3build1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../3-kubernetes-cni_0.8.7-00_amd64.deb ...
Unpacking kubernetes-cni (0.8.7-00) ...
Selecting previously unselected package socat.
Preparing to unpack .../4-socat_1.7.3.3-2_amd64.deb ...
Unpacking socat (1.7.3.3-2) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.22.3-00_amd64.deb ...
Unpacking kubelet (1.22.3-00) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../6-kubectl_1.22.3-00_amd64.deb ...
Unpacking kubectl (1.22.3-00) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../7-kubeadm_1.22.3-00_amd64.deb ...
Unpacking kubeadm (1.22.3-00) ...
Setting up conntrack (1:1.4.5-2) ...
Setting up kubectl (1.22.3-00) ...
Setting up ebtables (2.0.11-3build1) ...
Setting up socat (1.7.3.3-2) ...
Setting up cri-tools (1.19.0-00) ...
Setting up kubernetes-cni (0.8.7-00) ...
Setting up kubelet (1.22.3-00) ...
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /lib/systemd/system/kubelet.service.
Setting up kubeadm (1.22.3-00) ...
Processing triggers for man-db (2.9.1-1) ...
```

Networking

```
unigps@ip-172-31-107-158:~$ k get pods -A
NAMESPACE     NAME                                          READY   STATUS    RESTARTS   AGE
kube-system   coredns-78fcd69978-56f5h                      0/1     Pending   0          92s
kube-system   coredns-78fcd69978-b7fb4                      0/1     Pending   0          92s
kube-system   etcd-ip-172-31-107-158                        1/1     Running   0          100s
kube-system   kube-apiserver-ip-172-31-107-158              1/1     Running   0          97s
kube-system   kube-controller-manager-ip-172-31-107-158     1/1     Running   0          97s
kube-system   kube-proxy-2d8nx                              1/1     Running   0          93s
kube-system   kube-scheduler-ip-172-31-107-158              1/1     Running   0          97s
unigps@ip-172-31-107-158:~$
```

Networking

```
root@ip-172-31-107-158:/etc/kubernetes# service kubelet status
● 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 2021-11-11 05:23:22 UTC; 16min ago
       Docs: https://kubernetes.io/docs/home/
   Main PID: 4945 (kubelet)
      Tasks: 14 (limit: 2327)
     Memory: 40.1M
     CGroup: /system.slice/kubelet.service
             └─4945 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/lib/kubelet/config.yaml --network-plugin=cni -->

Nov 11 05:39:22 ip-172-31-107-158 kubelet[4945]: I1111 05:39:22.678325    4945 cni.go:239] "Unable to update cni config" err="no networks found in /etc/cni/net.d"
Nov 11 05:39:25 ip-172-31-107-158 kubelet[4945]: E1111 05:39:25.556760    4945 kubelet.go:2337] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady messag>
Nov 11 05:39:27 ip-172-31-107-158 kubelet[4945]: I1111 05:39:27.678493    4945 cni.go:239] "Unable to update cni config" err="no networks found in /etc/cni/net.d"
Nov 11 05:39:30 ip-172-31-107-158 kubelet[4945]: E1111 05:39:30.567582    4945 kubelet.go:2337] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady messag>
Nov 11 05:39:32 ip-172-31-107-158 kubelet[4945]: I1111 05:39:32.679221    4945 cni.go:239] "Unable to update cni config" err="no networks found in /etc/cni/net.d"
Nov 11 05:39:35 ip-172-31-107-158 kubelet[4945]: E1111 05:39:35.578361    4945 kubelet.go:2337] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady messag>
Nov 11 05:39:37 ip-172-31-107-158 kubelet[4945]: I1111 05:39:37.679897    4945 cni.go:239] "Unable to update cni config" err="no networks found in /etc/cni/net.d"
Nov 11 05:39:40 ip-172-31-107-158 kubelet[4945]: E1111 05:39:40.588890    4945 kubelet.go:2337] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady messag>
Nov 11 05:39:42 ip-172-31-107-158 kubelet[4945]: I1111 05:39:42.680298    4945 cni.go:239] "Unable to update cni config" err="no networks found in /etc/cni/net.d"
Nov 11 05:39:45 ip-172-31-107-158 kubelet[4945]: E1111 05:39:45.599653    4945 kubelet.go:2337] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady messag>
```

Networking

```
root@ip-172-31-107-158:/etc/kubernetes# docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS     NAMES
a7099bc9d43e   6120bd723dce           "/usr/local/bin/kube…"   19 minutes ago   Up 19 minutes            k8s_kube-proxy_kube-proxy-2d8nx_kube-system_df5d7374-6a19-4a62-8289-ec4d7b0c0a99_0
49a813743c95   k8s.gcr.io/pause:3.5   "/pause"                 19 minutes ago   Up 19 minutes            k8s_POD_kube-proxy-2d8nx_kube-system_df5d7374-6a19-4a62-8289-ec4d7b0c0a99_0
3363f0d3ca15   53224b502ea4           "kube-apiserver --ad…"   19 minutes ago   Up 19 minutes            k8s_kube-apiserver_kube-apiserver-ip-172-31-107-158_kube-system_ad6ee19c867dd22e55ff3914824341b4_
0
5357e5614358   004811815584           "etcd --advertise-cl…"   19 minutes ago   Up 19 minutes            k8s_etcd_etcd-ip-172-31-107-158_kube-system_42a716c5a59ce2866ebe61ba568164db_0
7930bffe223d   0aa9c7e31d30           "kube-scheduler --au…"   19 minutes ago   Up 19 minutes            k8s_kube-scheduler_kube-scheduler-ip-172-31-107-158_kube-system_aeb141e721a61956af81d401d190f311_
0
ceee5e692ebd   05c905cef780           "kube-controller-man…"   19 minutes ago   Up 19 minutes            k8s_kube-controller-manager_kube-controller-manager-ip-172-31-107-158_kube-system_09dcfaf554836a3
3271cf6166acc0755_0
26bebcddc57a   k8s.gcr.io/pause:3.5   "/pause"                 19 minutes ago   Up 19 minutes            k8s_POD_kube-scheduler-ip-172-31-107-158_kube-system_aeb141e721a61956af81d401d190f311_0
934f2af6644b   k8s.gcr.io/pause:3.5   "/pause"                 19 minutes ago   Up 19 minutes            k8s_POD_kube-controller-manager-ip-172-31-107-158_kube-system_09dcfaf554836a33271cf6166acc0755_0
60bc7ca7c123   k8s.gcr.io/pause:3.5   "/pause"                 19 minutes ago   Up 19 minutes            k8s_POD_kube-apiserver-ip-172-31-107-158_kube-system_ad6ee19c867dd22e55ff3914824341b4_0
aed4d7056f21   k8s.gcr.io/pause:3.5   "/pause"                 19 minutes ago   Up 19 minutes            k8s_POD_etcd-ip-172-31-107-158_kube-system_42a716c5a59ce2866ebe61ba568164db_0
root@ip-172-31-107-158:/etc/kubernetes#
```

CNI - Interface Methods

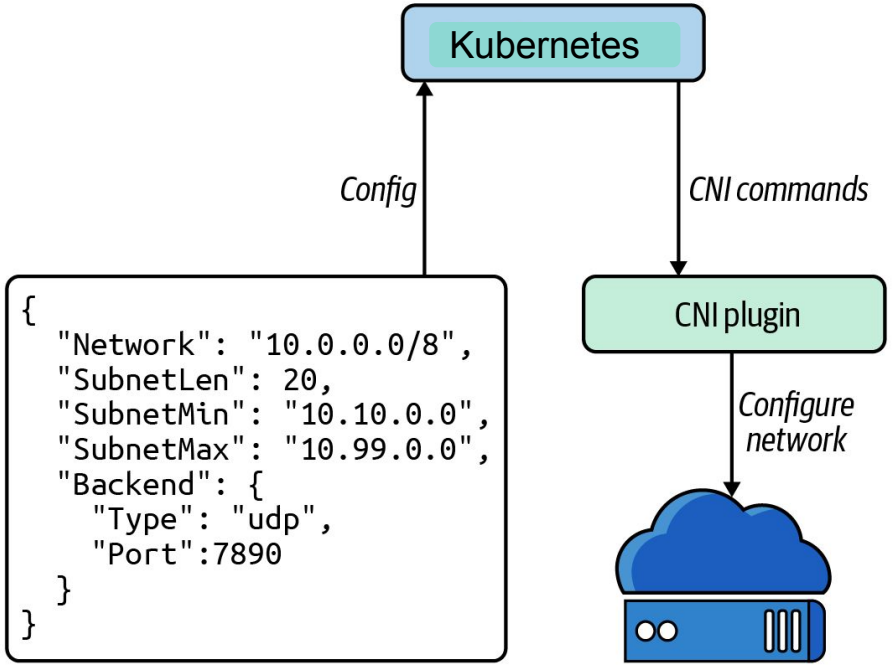
- Add
  - Add a container to the network
- DEL
  - Delete a container from the network
- CHECK
  - Return an error if any issue
- VERSION
  - Report version info about the plugin

Ref: https://github.com/containernetworking/cni/blob/master/SPEC.md

CNI - Flow

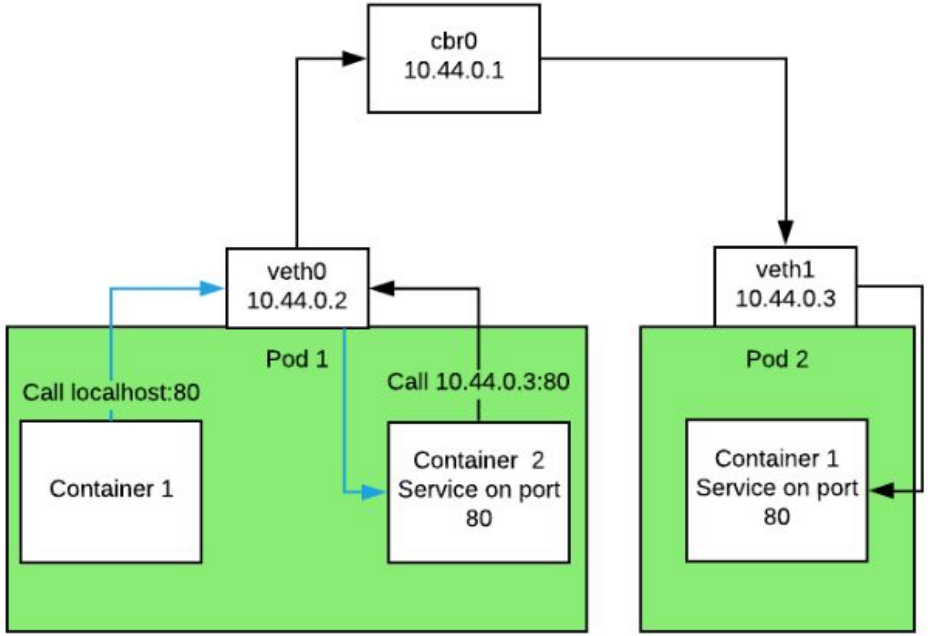
CNI Requirements

- IPAM - IP Address Management
- Connectivity in Container Network
- Route Advertisement

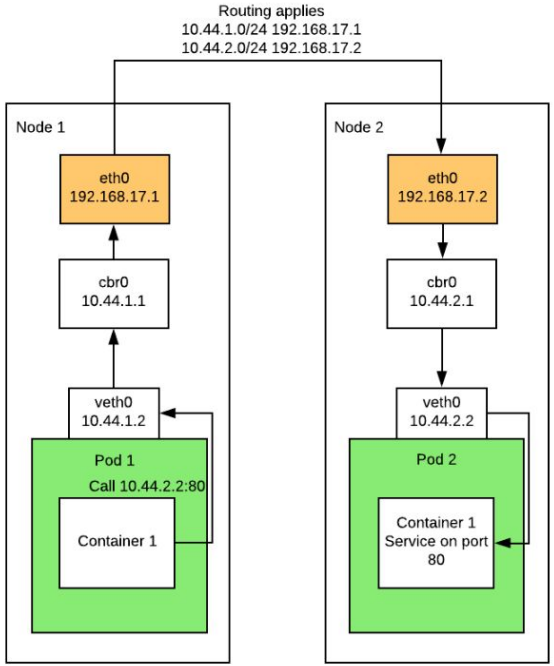


Ref: Oreilly docs

Network Routing - Same Host



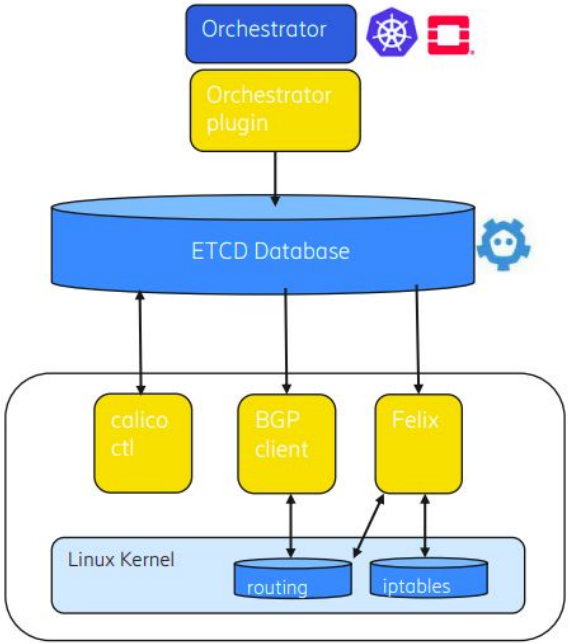
Network Routing - Across Nodes



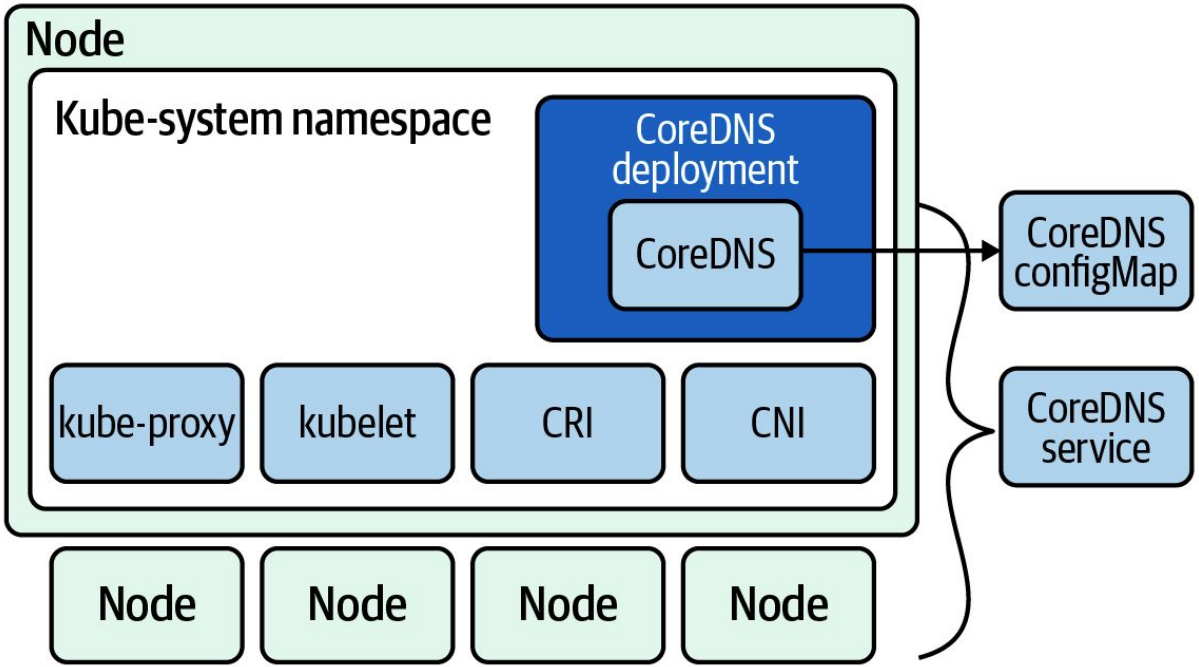
Calico Architecture

Features

- Layer 3 Routing
- BGP for Routes distribution
- Support Network Policy
- Uses iptables & kernel routing



Networking - CoreDNS



Ref: Oreilly docs

CNI Plugins - Benchmarking

CNI Benchmark August 2020 infraBuilder	Config	Performances (bandwidth)				Resources consumption (cpu/ram)					Security features			
	MTU	Pod to Pod		Pod to Service		Idle	Pod to Pod		Pod to Service		Network Policies		Encryption	
	setting	TCP	UDP	TCP	UDP	none	TCP	UDP	TCP	UDP	in	out	activation	Performance
Antrea	auto	Very fast	Very fast	Very fast	Slow	Low	Low	Low	Low	Low	yes	yes	at deploy time	Slow
Calico	manual	Very fast	Very fast	Very fast	Fast	Low	Very low	Very low	Very low	Very low	yes	yes	anytime	Very fast
Canal	manual	Very fast	Very fast	Very fast	Very fast	Low	Very low	Very low	Very low	Very low	yes	yes	no	n/a
Cilium	auto	Fast	Very fast	Very fast	Very fast	High	High	High	High	High	yes	yes	at deploy time	Slow
Flannel	auto	Very fast	Very fast	Very fast	Very fast	Very low	Very low	Very low	Very low	Very low	no	no	no	n/a
Kube-OVN	auto	Fast	Very slow	Fast	Very slow	High	High	High	High	High	yes	yes	no	n/a
Kube-router	none	Slow	Very slow	Slow	Very slow	Low	Very low	Low	Very low	Low	yes	yes	no	n/a
Weave Net	manual	Very fast	Very fast	Very fast	Fast	Very low	Low	Low	Low	Low	yes	yes	at deploy time	Slow

Ref: https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49

Volumes | Persistence

- Volumes
- Persistence Volume (pv)
- Persistence Volume Claim (pvc)
- Statefulset (sts)
- Daemonset (ds)

Overview - Volumes | Persistent

- Ephemeral Volumes
  - Tightly coupled with POD lifetime
  - Deleted when POD is removed
  - Example: emptydir
- Persistent Volumes
  - 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

Volume - emptyDir

training > kubernetes > lab > persistence > ! emptyDir-volume.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: volume-emptydir
spec:
  containers:
  - image: nginx
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

PV - hostpath

sljpmk8s > kubernetes > lab > persistence > ! hostpath-volume.yaml >

io.k8s.api.core.v1.Pod (v1@pod.json)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hostpath-volume
spec:
  containers:
  - image: nginx
    name: test-container
    volumeMounts:
    - mountPath: /data-mounted-as
      name: hostpath-volume
    resources:
      limits:
        cpu: "20m"
        memory: "50Mi"
  volumes:
  - name: hostpath-volume
    hostPath:
      # directory location on host
      path: /data
      # this field is optional
      type: DirectoryOrCreate
```

PV - Cloud Storage (AWS EBS)

sljpmk8s > kubernetes > lab > persistence > ! 03-aws-ebs-volume.yaml >

io.k8s.api.core.v1.Pod (v1@pod.json)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: volume-ebs
spec:
  nodeName: ip-172-31-58-74.ec2.internal
  containers:
  - image: nginx
    name: test-container
    volumeMounts:
    - mountPath: /test-ebs
      name: ebs-volume
    resources:
      limits:
        cpu: "20m"
        memory: "50Mi"
  volumes:
  - name: ebs-volume
    # This AWS EBS volume must already exist.
    awsElasticBlockStore:
      volumeID: vol-081b401a592b794a2
      fsType: ext4
```

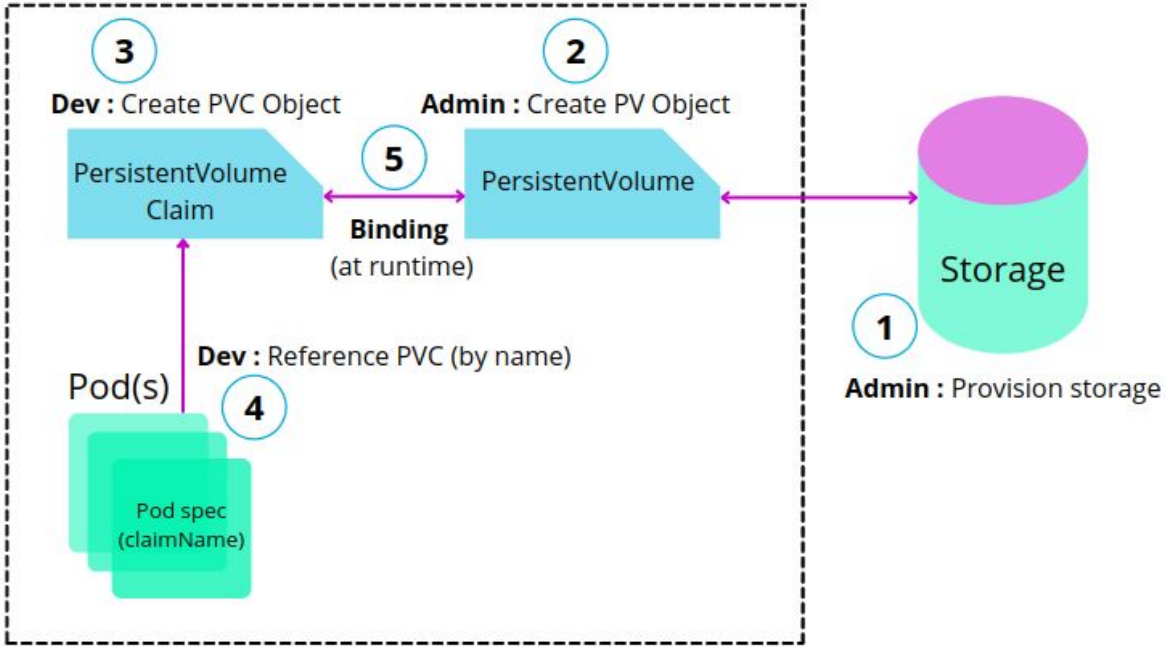
PV - Config Volume

sljpmk8s > kubernetes > lab > persistence > ! 04-config-volume.yaml > io.k8s.api.core.v1.Pod (v1@pod.json) | io.k8s.api.core.v1.ConfigMap (v1

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: config-volume
spec:
  containers:
    - name: test
      image: nginx
      volumeMounts:
        - name: config-vol
          mountPath: /etc/config
      resources:
        limits:
          cpu: "20m"
          memory: "50Mi"
  volumes:
    - name: config-vol
      configMap:
        name: config-volume
        items:
          - key: log_level
            path: log.properties
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: config-volume
data:
  log_level: |
    debug=true
```

Persistent Volume - static



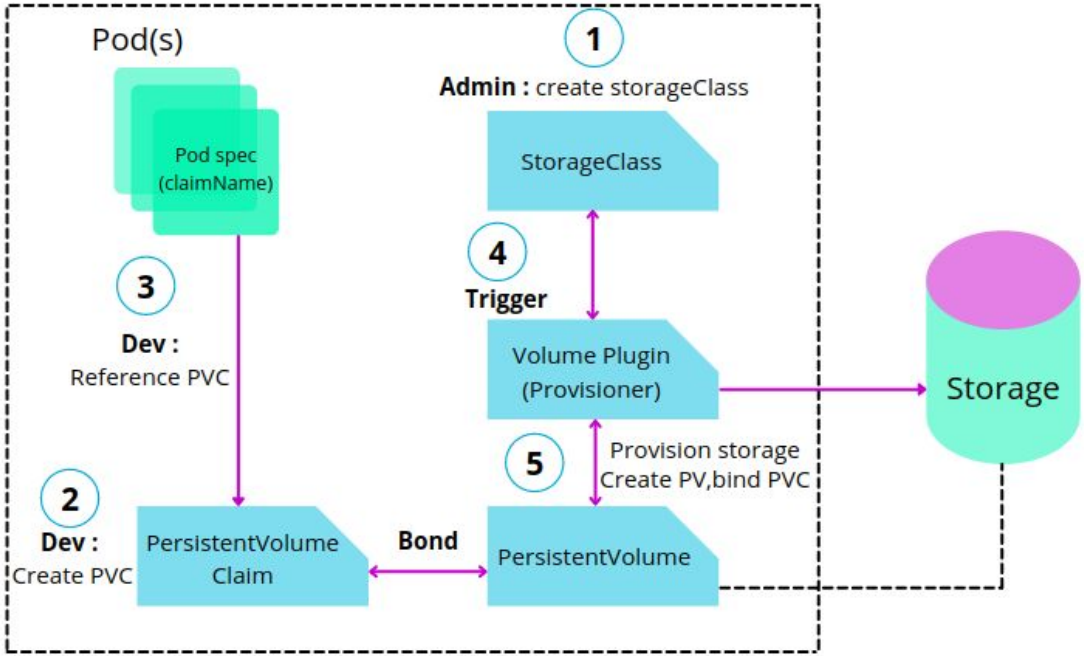
PV - Static

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-hostpath
  annotations:
    pv.beta.kubernetes.io/gid: "1234"
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-pv-pvc
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: pvc-test
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: pv-storage
```

Persistent Volume - Dynamic



Example

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

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-dynamic
  labels:
    app: nginx
spec:
  storageClassName: storageclass-generic
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: pod-volume-dynamic
  labels:
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: external
  volumes:
    - name: external
      persistentVolumeClaim:
        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

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: sts-web
spec:
  serviceName: "nginx"
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: k8s.gcr.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

- 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

```yaml
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

Please refer the google classwork link given in the chat message

And do all the lab work as per the instructions noted in the classwork assignments

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

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-game-demo
data:
  # property-like keys; each key maps to a simple value
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"
  # file-like keys
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-configmap
spec:
  containers:
    - name: demo
      image: alpine
      command: ["sleep", "3600"]
      env:
        # Define the environment variable
        - name: PLAYER_INITIAL_LIVES # Notice that the case is different here
                                     # from the key name in the ConfigMap.
          valueFrom:
            configMapKeyRef:
              name: cm-game-demo           # The ConfigMap this value comes from.
              key: player_initial_lives # The key to fetch.
        - name: UI_PROPERTIES_FILE_NAME
          valueFrom:
            configMapKeyRef:
              name: cm-game-demo
              key: ui_properties_file_name
      volumeMounts:
      - name: config
        mountPath: "/config"
        readOnly: true
  volumes:
    # You set volumes at the Pod level, then mount them into containers inside that Pod
    - name: config
      configMap:
        # Provide the name of the ConfigMap you want to mount.
        name: cm-game-demo
        # An array of keys from the ConfigMap to create as files
        items:
        - key: "game.properties"
          path: "game.properties"
        - key: "user-interface.properties"
          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

```yaml
apiVersion: v1
data:
  username: YWRtaW4=
  password: MWYyZDFlMmU2N2Rm
kind: Secret
metadata:
  name: pod-secret
  namespace: default
  resourceVersion: "164619"
  uid: cfee02d6-c137-11e5-8d73-42010af00002
type: Opaque
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-secret
spec:
  containers:
  - name: mypod
    image: redis
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: username
      - name: SECRET_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: password
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: pod-secret
      items:
      - key: username
        path: my-group/my-username
```

Security Context

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: sec-ctx-vol
    emptyDir: {}
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: sec-ctx-vol
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        add: ["NET_ADMIN", "SYS_TIME"]

```

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

Please refer the google classwork link given in the chat message

And do all the lab work as per the instructions noted in the classwork assignments

Scheduling

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

Overview

Scheduling - Matching PODs to Nodes so that kubelet can run them

Filtering
Scoring

Node Name

training > kubernetes > lab > scheduling > ! schedule-nodename.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: schedule-node
spec:
  replicas: 1
  selector:
    matchLabels:
      component: schedule-node
  template:
    metadata:
      labels:
        component: schedule-node
    spec:
      nodeName: ip-172-31-102-18.ec2.internal
      containers:
        - name: test-app
          image: brainupgrade/test-app:all-tiers-in-one
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8080
          resources:
            requests:
              cpu: "100m"
              memory: "250Mi"
```

Node Selector

training > kubernetes > lab > scheduling > ! schedule-nodeselector.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: schedule-nodeselector
spec:
  replicas: 1
  selector:
    matchLabels:
      component: schedule-nodeselector
  template:
    metadata:
      labels:
        component: schedule-nodeselector
    spec:
        nodeSelector:
          node-role.kubernetes.io/spot-worker: "true"
        containers:
          - name: test-app
            image: nginx
            imagePullPolicy: IfNotPresent
            ports:
              - containerPort: 80
```

POD - Affinity & Anti Affinity (example 1)

training > kubernetes > lab > scheduling > ! pod-affinity.yaml

```yaml
    spec:
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: tier
                operator: In  # NotIn, Exists, NotExists
                values:
                - cache
            topologyKey: topology.kubernetes.io/zone
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 100
            podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: tier
                  operator: In
                  values:
                  - messaging
              topologyKey: kubernetes.io/hostname
      containers:
        - name: pod-affinity-middle-tier
          image: nginx
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
```

POD - Affinity & Anti Affinity (example 2)

training > kubernetes > lab > scheduling > pod-affinity-web-cache > ! app-cache-store.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-cache-store
spec:
  selector:
    matchLabels:
      app: cache-store
  replicas: 2
  template:
    metadata:
      labels:
        app: cache-store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - cache-store
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: redis-server
        image: redis:3.2-alpine
```

training > kubernetes > lab > scheduling > pod-affinity-web-cache > ! app-web-server.yaml

```yaml
  replicas: 2
  template:
    metadata:
      labels:
        app: web-server
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - web-server
            topologyKey: "kubernetes.io/hostname"
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - cache-store
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: web-app
        image: nginx:1.16-alpine
```

Node - Affinity

training > kubernetes > lab > scheduling > ! node-affinity.yaml

```yaml
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In  #In, NotIn, Exists, DoesNotExist, Gt, Lt
            values:
            - us-east-1b
            - us-east-1c
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: kops.k8s.io/instancegroup
            operator: In  #NotIn,  DoesNotExist for AntiAffinity
            values:
            - spotnodes
  containers:
    - name: nginx
      image: nginx
      imagePullPolicy: IfNotPresent
      ports:
        - 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

    kubectl taint nodes qanode release=qa:NoSchedule

    kubectl taint nodes devnode release=dev:PreferNoSchedule

    kubectl taint nodes prodnode release=prod:NoExecute
  - To remove

    kubectl taint nodes node-name key=value:Effect-

Tolerations

training > kubernetes > lab > scheduling > ! tolerations-master.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: tolerations-master
spec:
  containers:
  - image: nginx
    name: nginx
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  tolerations:
    - key: "node-role.kubernetes.io/master"
      operator: "Exists"
      effect: "NoSchedule"
```

Exercises

Please refer the google classwork link given in the chat message

And do all the lab work as per the instructions noted in the classwork assignments

Services

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

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

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
```

Service Types

- Cluster IP
  - Service exposed on cluster internal IP
  - Reachable only within cluster
- Node Port
  - Exposed on each Node IP at static port (30000-32767)
- Load Balancer
  - Exposed through external cloud load balancer
- ExternalName
  - Maps the service to the external name like api.example.com

Service - External Name

training > kubernetes > lab > services > ! externalName.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: external-name
spec:
  type: ExternalName
  externalName: google.com
```

Service - External IP

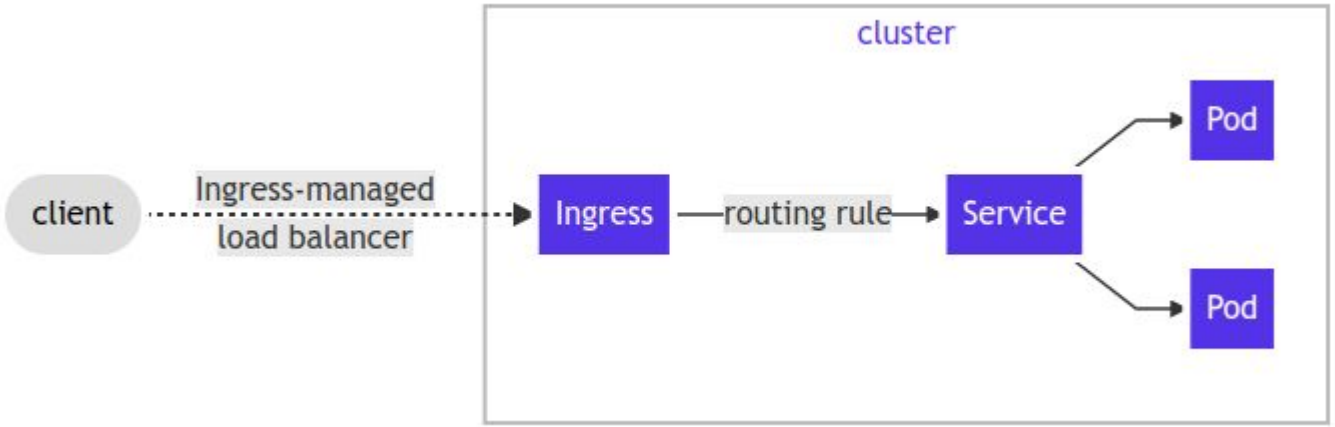
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 9376
  externalIPs:
    - 80.11.12.10
```

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



Source: kubernetes.io

Ingress

training > kubernetes > lab > ingress > ! ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: rajeshg.brainupgrade.in
  annotations:
    cert-manager.io/cluster-issuer: letsencrypt-prod
    kubernetes.io/ingress.class: nginx
    kubernetes.io/tls-acme: "true"
    nginx.ingress.kubernetes.io/enable-cors: "true"
    nginx.ingress.kubernetes.io/proxy-body-size: 50m
    nginx.ingress.kubernetes.io/proxy-read-timeout: "3600"
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/use-regex: "true"
spec:
  tls:
  - hosts:
    - rajeshg.brainupgrade.in
    secretName: rajeshg.brainupgrade.in
  rules:
    - host: rajeshg.brainupgrade.in
      http:
        paths:
          - path: /?(.*)
            backend:
              serviceName: api-v1
              servicePort: 8080
          - path: /api/?(.*)
            backend:
              serviceName: api-v2
              servicePort: 8080
          - path: /logger/?(.*)
            backend:
              serviceName: logger
              servicePort: 80
```

Network Policy

training > kubernetes > lab > networking > ! network-policy.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: weather-network-policy
  namespace: weather
spec:
  podSelector:
    matchLabels:
      app: weather-db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    # - ipBlock:
    #     cidr: 172.17.0.0/16
    #     except:
    #     - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          app: weather
    - podSelector:
        matchLabels:
          app: weather-service
    ports:
    - protocol: TCP
      port: 3306
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

Examples

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

Exercises

Please refer the google classwork link given in the chat message

And do all the lab work as per the instructions noted in the classwork assignments

Cluster Upgrade - KubeAdm

Master Nodes

- apt update && kubeadm upgrade apply
- apt-cache madison kubeadm
- apt-mark unhold kubeadm && apt-get update && apt-get install -y kubeadm=1.18.5-00 && apt-mark hold kubeadm
- kubeadm version
- kubectl drain <cp-node-name> --ignore-daemonsets
- sudo kubeadm upgrade apply v1.18.x
- kubectl uncordon <cp-node-name>
- kubectl drain <cp-node-name> --ignore-daemonsets
- kubeadm upgrade node
- kubectl uncordon <cp-node-name>

Worker Nodes

- apt-mark unhold kubelet kubectl && apt-get update && apt-get install -y kubelet=1.18.x-00 kubectl=1.18.x-00 && apt-mark hold kubelet kubectl
- sudo systemctl daemon-reload
- sudo systemctl restart kubelet
- apt-mark unhold kubeadm && apt-get update && apt-get install -y kubeadm=1.18.x-00 && apt-mark hold kubeadm
- kubectl drain <node-to-drain> --ignore-daemonsets
- kubeadm upgrade node
- apt-mark unhold kubelet && apt-get update && apt-get install -y kubelet=1.18.x-00 && apt-mark hold kubelet
- kubectl uncordon <node-to-drain>

Helm

- What it is
  - Kubernetes package manager, collection of templates & settings also known as helm charts

- Benefits
  - Smooth deployment of sophisticated applications
  - Especially useful when app to be deployed across many clusters with custom configuration
  - Easy upgrade and rollback

Helm - Package Publish Deploy

Package & Publish

- Create git repo named helm-charts and use branch gh-pages
- Create helm chart (kubernetes application package)
  - helm create my-nginx
- Package the app
  - helm package my-nginx
- Create helm repo index from outside of git directory
  - helm repo index helm-charts --url https://<git-username>.github.io/helm-charts
- Add git changes, commit and push the changes to gh-pages
  - git add .
  - git commit -m “my helm chart” .
  - git push origin gh-pages

Deploy

- helm repo add my-charts https://<git-username>.github.io/helm-charts
- helm repo update
- helm install my-nginx my-charts/my-nginx

Upgrade & Rollback

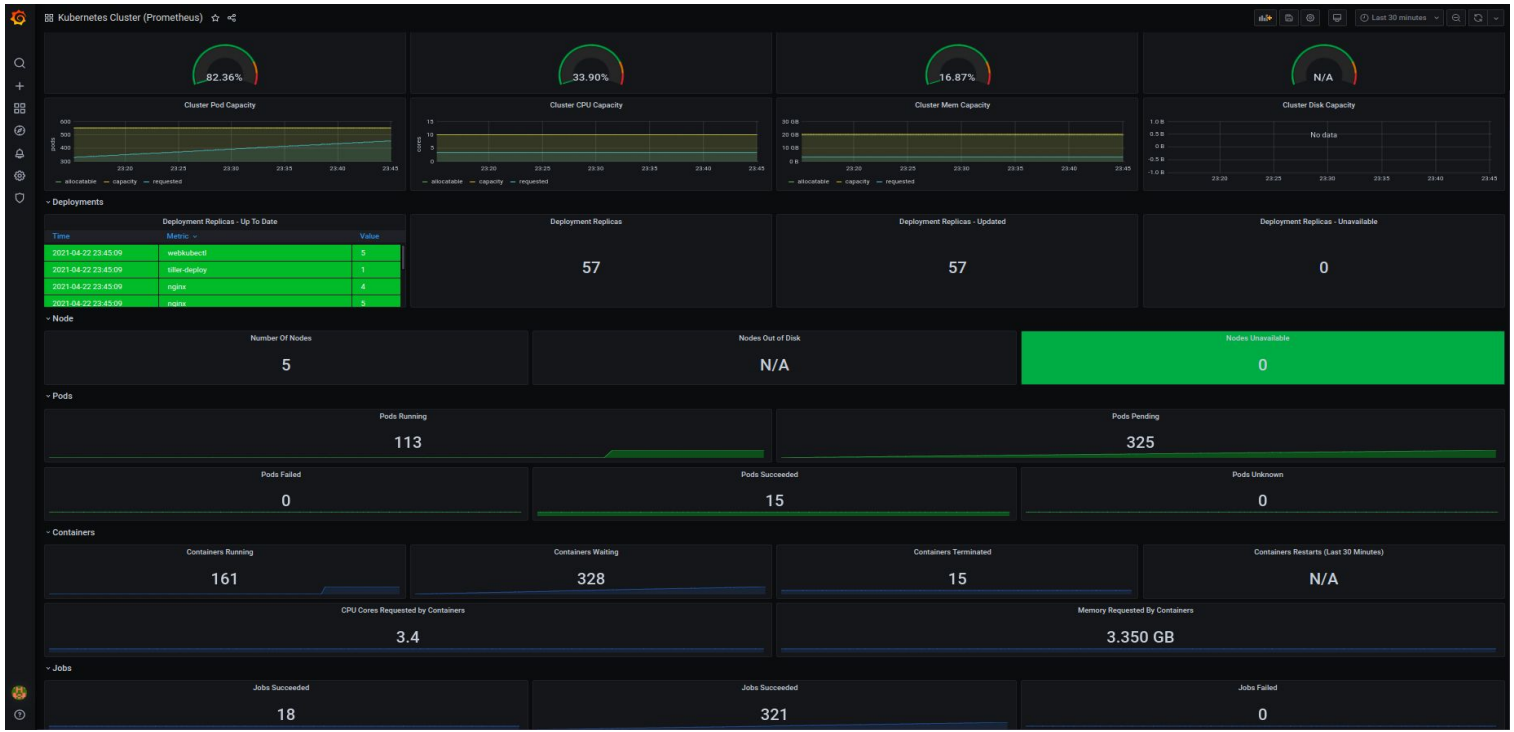
- helm upgrade my-nginx my-nginx
- helm rollback my-nginx 1

Monitoring & Misc

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

BrainUpgrade Academy

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

Thank You for your active participation!

info@brainupgrade.in

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

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

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