Kubernetes

Verizon Study Groups

Agenda

- Introduction
- Prerequisites
- What is Kubernetes?
- Why you need Kubernetes?
- Kubernetes Components
  - Control Plane Components
  - Node Components

Prerequisites

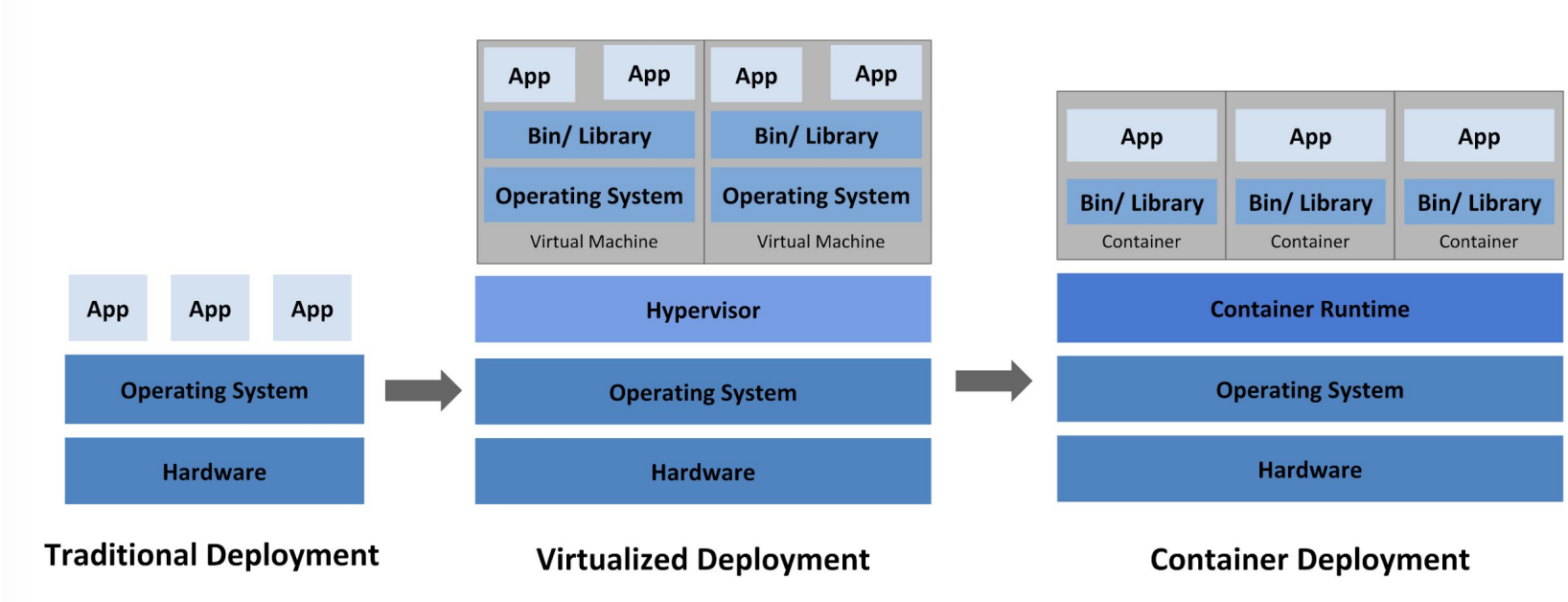
- Docker
- Setup lab environment for hands-on (kubectl, minikube,)
  https://kubernetes.io/docs/tasks/tools/

  Verizon Kubernetes Sandbox/Playground

What is Kubernetes?

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

Evolution


App
App
App
Operating System
Hardware
Traditional Deployment
App
App
App
App
Bin/ Library
Bin/ Library
Operating System
Operating System
Virtual Machine
Virtual Machine
Hypervisor
Operating System
Hardware
Virtualized Deployment
App
App
App
Bin/ Library
Bin/ Library
Bin/ Library
Container
Container
Container
Container Runtime
Operating System
Hardware
Container Deployment

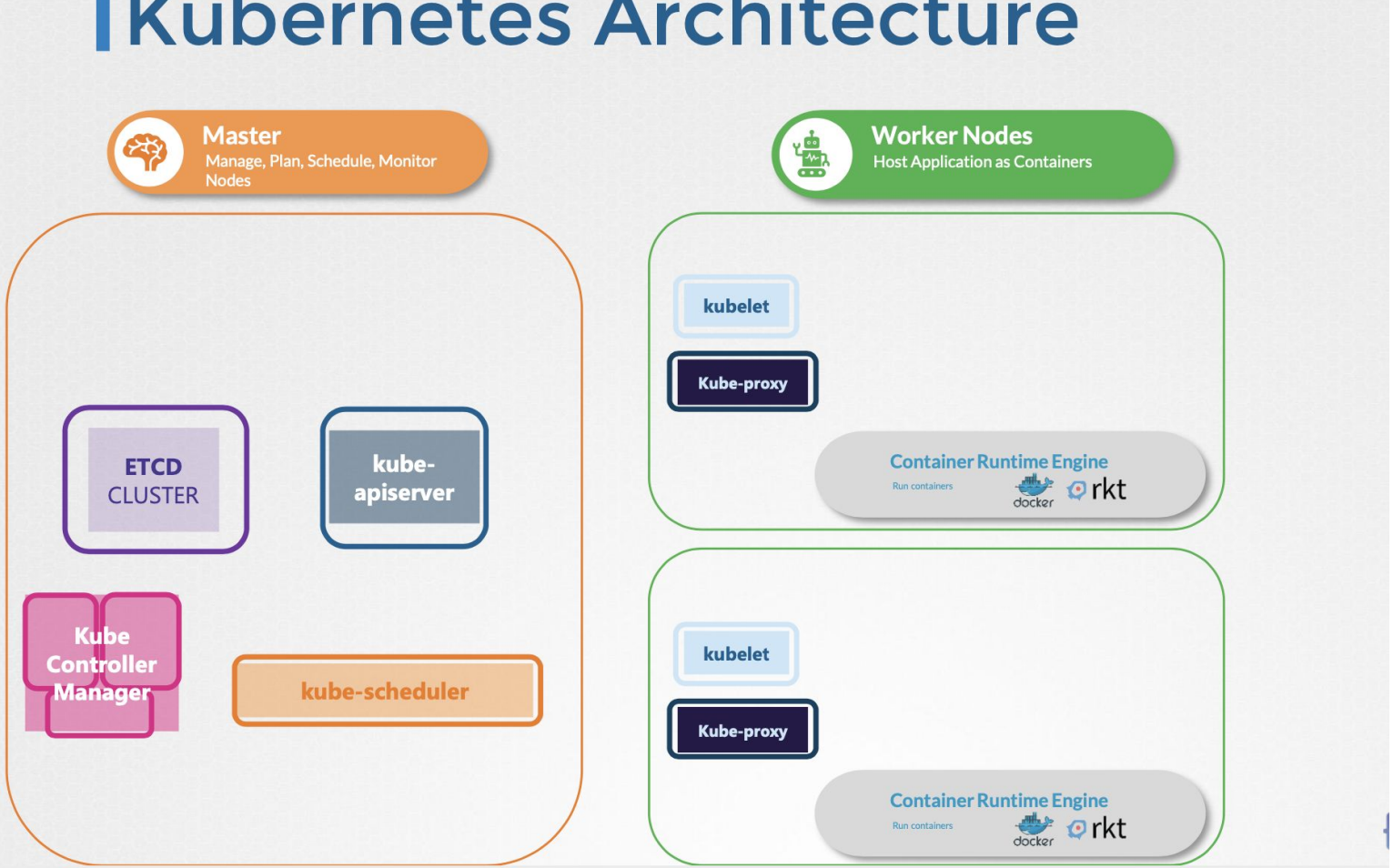
Why you need Kubernetes?

Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more

Kubernetes Components

https://www.youtube.com/watch?v=8C_SCDbUJTg

Kubernetes Architecture



kube-apiserver

The core of Kubernetes' control plane

The API server exposes an HTTP API which can be used to query the cluster

ETCD - https://etcd.io/

etcd is a consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

KEY	VALUE
Name	Smith
Age	33
Location	New York

```
{
Name : Smith,
Age  : 33,
Location : New York
}
```

kube-scheduler

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

kube-controller-manager

Control Plane component that runs controller processes.

cloud-controller-manager

A Kubernetes control plane component that embeds cloud-specific control logic

References

https://kubernetes.io/docs/home/

https://www.youtube.com/watch?v=8C_SCDbUJTg

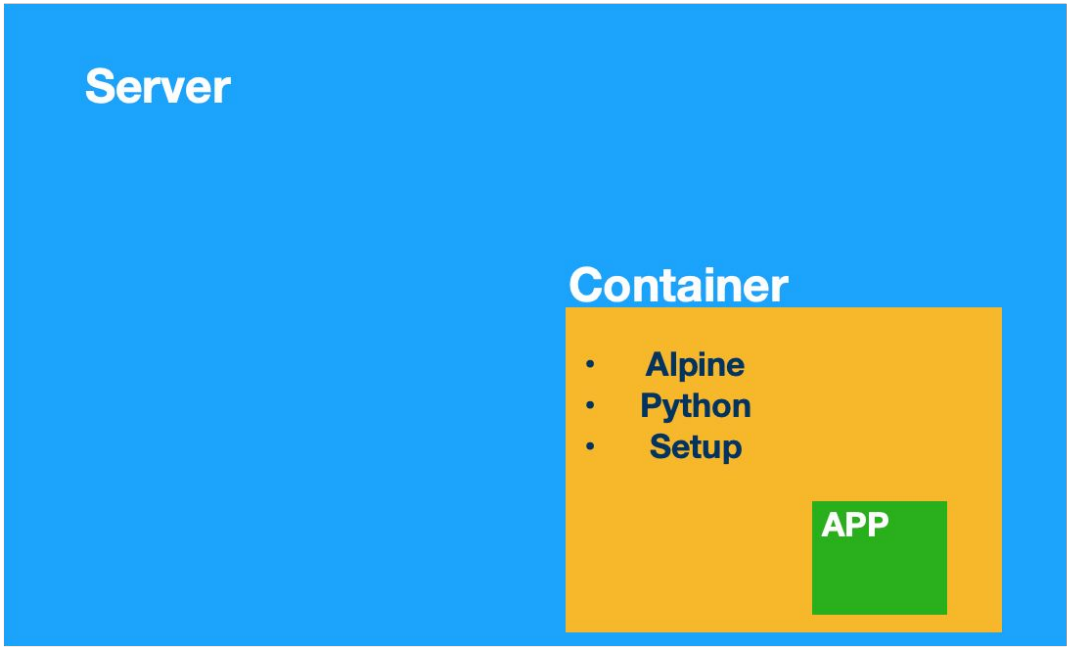
https://etcd.io/

Agenda - Day 2

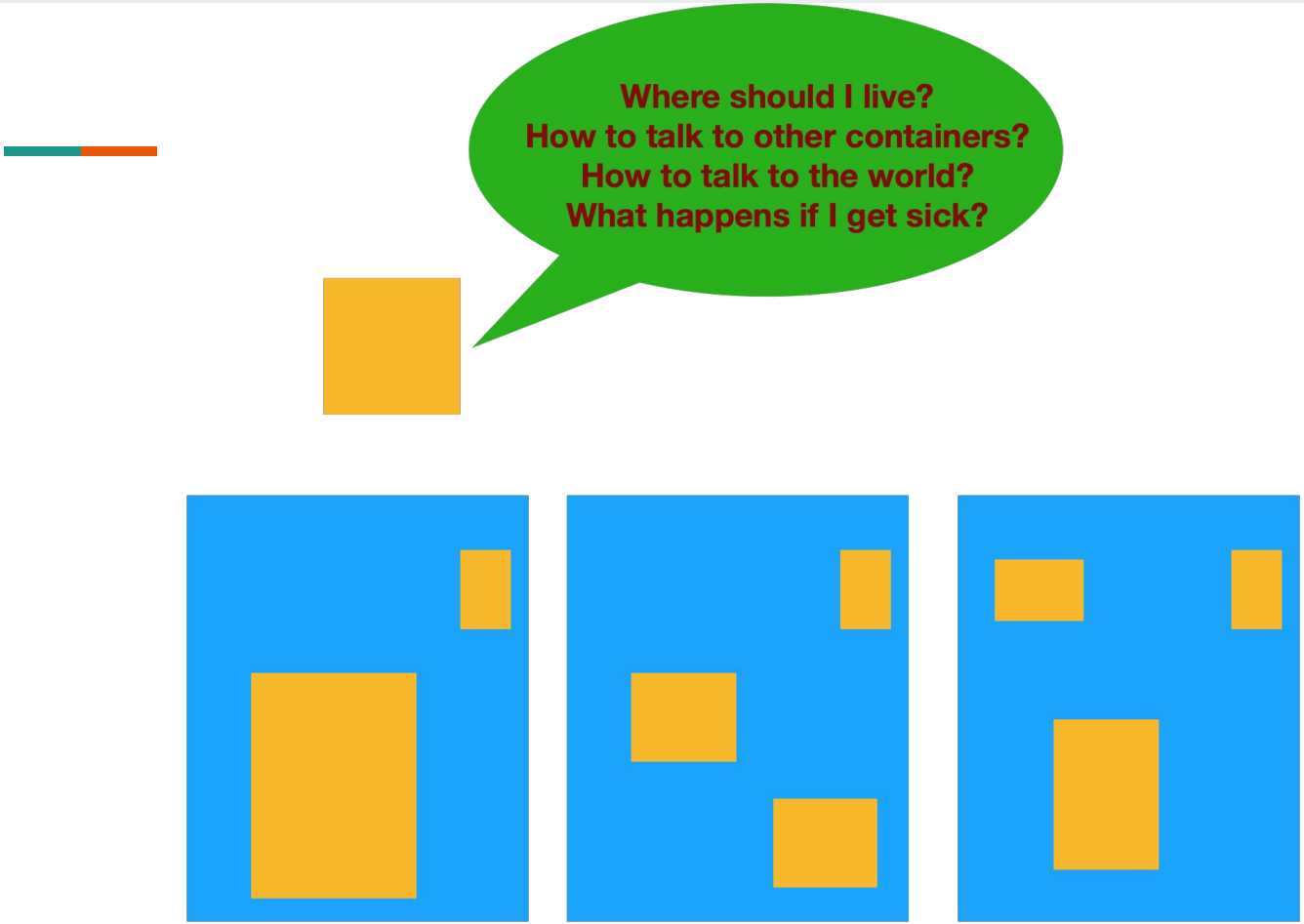
- Pods
- Deployments (Replicasets)
- Services
- Namespaces
- YAML
- Static Pods
- DaemonSets

Container

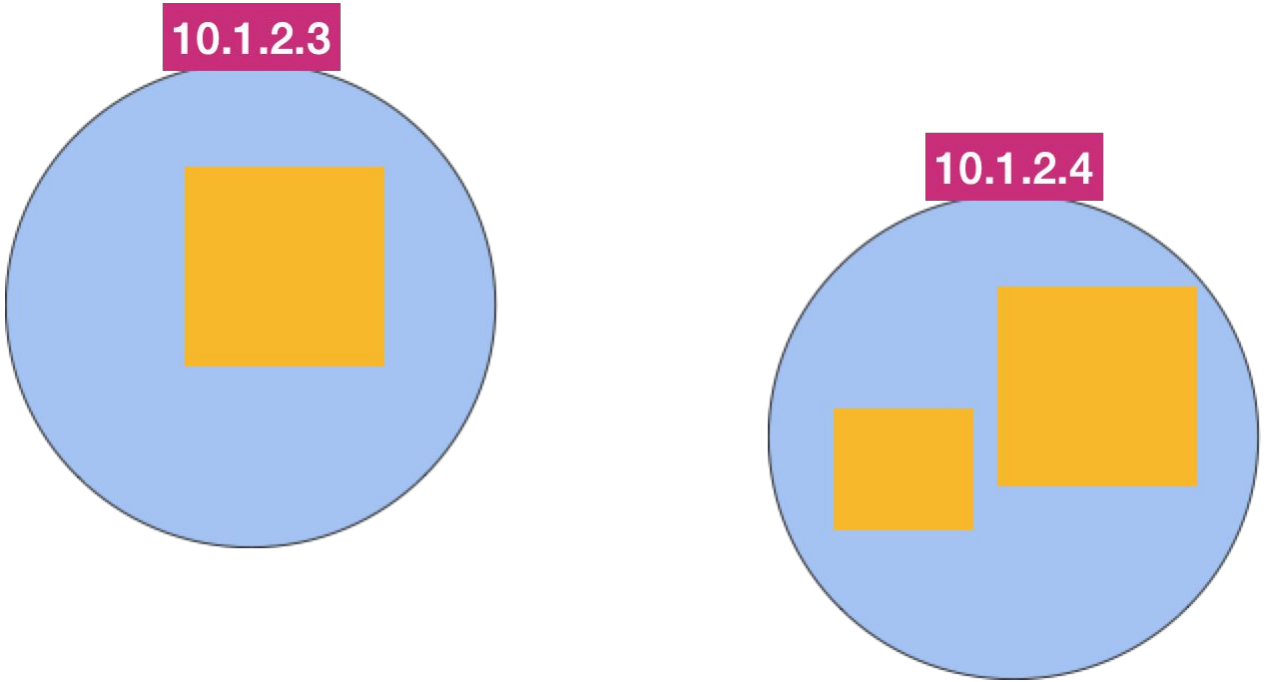
Baby computer inside a computer



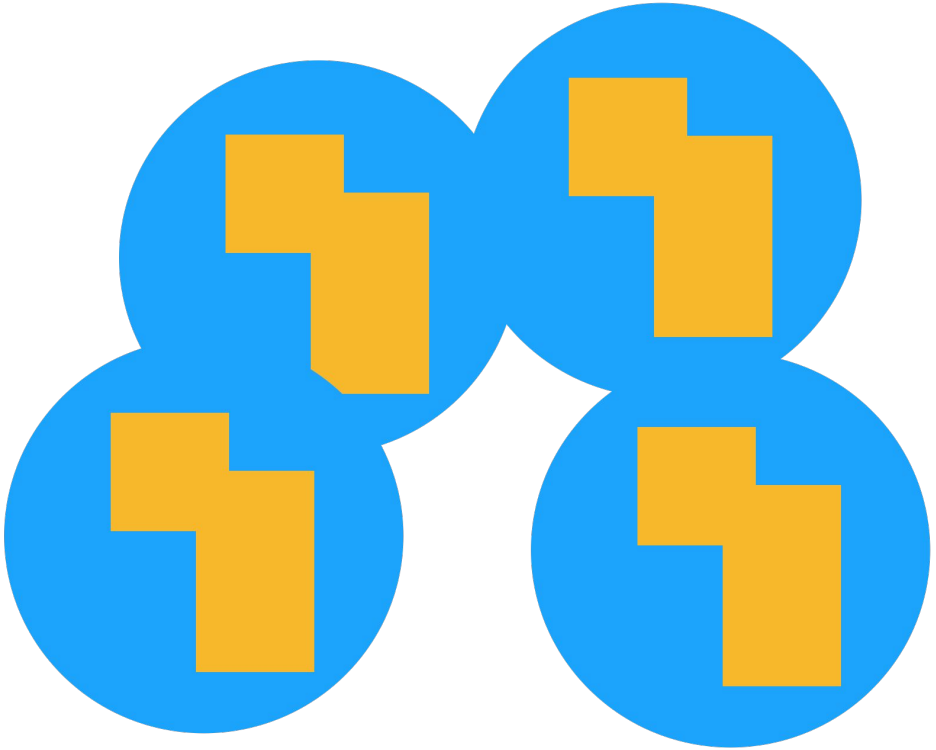

```
FROM  python:alpine

WORKDIR /app

RUN execute_cmnds

EXPOSE 80

CMD [“python”, “app.py”]
```


Where should I live?
How to talk to other containers?
How to talk to the world?
What happens if I get sick?

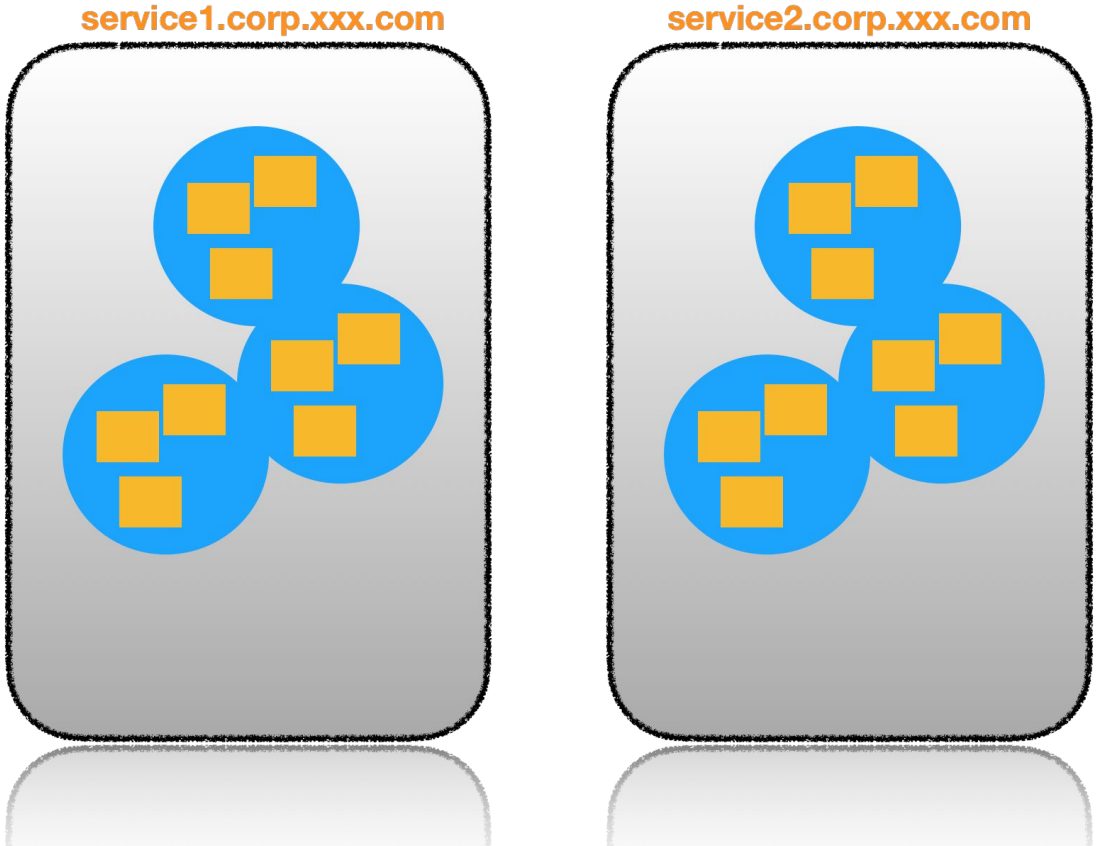
POD



Deployments



Services



YAML (Yet Another Markup Language)

apiVersion:

kind: `

metadata:

spec:

Namespaces

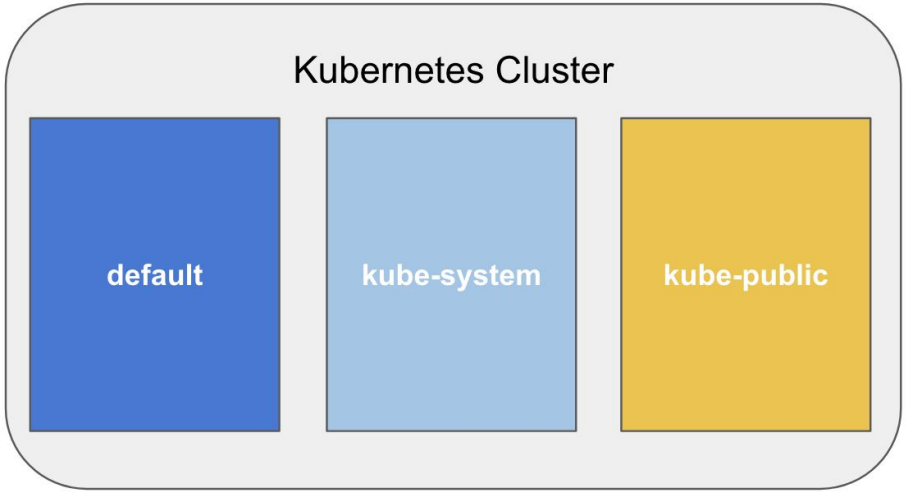
Virtual cluster inside your Kubernetes cluster

//creating namespace with single command

$ kubectl create namespace dev

namespace/dev created.

Imperative Way



PODS

kubectl run nginx **--image**=nginx

kubectl create -f nginx.yaml

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#run

Deployments

kubectl create deployment nginx --image nginx

kubectl create -f nginx.yaml

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#-em-deployment-em-

Services

kubectl create service

kubectl expose pod valid-pod **--port**=444 **--name**=front

kubectl create -f nginx.yaml

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#expose

Static Pods

Static Pods are managed directly by the **kubelet daemon** on a specific node, without the API server observing them. Unlike Pods that are managed by the control plane (for example, a Deployment); instead, the kubelet watches each static Pod (and restarts it if it fails).

Multiple Pods

Resources that manage one or more Pods:

- Deployment
- StatefulSet
- DaemonSet

DaemonSets

A *DaemonSet* ensures that all (or some) Nodes run **a copy** of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.

Use Cases:

- Logs, monitoring daemons

References

YAML: https://www.youtube.com/watch?v=1uFVr15xDGg

https://www.youtube.com/watch?v=qmDzcu5uY1I

Lab Practice

- List all the namespaces
- List all the pods in your namespace
- Create a namespace.
- Create a pod with image nginx in the namespace created
- Create a service to expose the pod on port 8080
- Create a deployment with “redis” image on port 6379 and expose a service.
- Increase the number of replicas to 3

Agenda - Day 3

- Services - Kinds
- Scheduling
- Labels and Selectors
- Taints and Tolerations
- Node Selectors
- Node Affinity
- Lab Practice

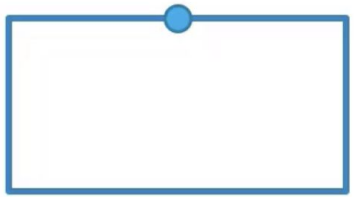
Services - Kinds

Kubernetes ServiceTypes allow you to specify what kind of Service you want.

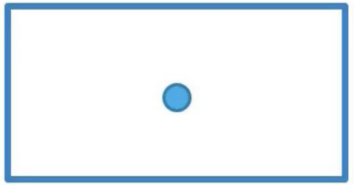
Type values:

- ClusterIP (default)
- NodePort
- LoadBalancer
- ExternalName

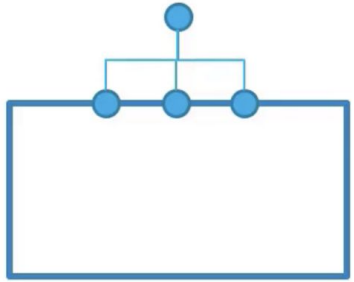
Services Types



NodePort



ClusterIP



LoadBalancer

Services - Reference

https://www.youtube.com/watch?v=5IzUpDtmWgM

https://www.youtube.com/watch?v=T4Z7visMM4E

Scheduling

What to schedule?

Where to schedule?

Manual Scheduling - Use NodeName before creating the pod

https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/

Scheduling - Reference

https://www.youtube.com/watch?v=rDCWxkvPlAw

https://www.youtube.com/watch?v=E3ExWruji7g

Labels and Selectors

Labels - Grouping

Selectors - Filtering

```
"metadata": {
  "labels": {
    "key1" : "value1",
    "key2" : "value2"
  }
}
```

Labels and Selectors

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"

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: label-demo
  labels:
    environment: production
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

Selector Examples

```
kubectl get pods --show-labels
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)'
kubectl get pods --selector version=1.0
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: my-first
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
```

Taints and Tolerations

https://www.youtube.com/watch?v=mo2UrkjA7FE

Taints - On Nodes to prevent pods to be placed on a node

Tolerations - On Pods to tolerate the taint

```
kubectl taint nodes node1 key1=value1:NoSchedule
```

https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/

NodeSelectors

Select the node on which a pod is to be placed

`kubectl label nodes kubernetes-foo-node-1.c.a-robinson.internal disktype=ssd`.

https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/

NodeAffinity

It allows you to constrain which nodes your pod is eligible to be scheduled on, based on labels on the node.

requiredDuringSchedulingIgnoredDuringExecution and preferredDuringSchedulingIgnoredDuringExecution

https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#node-affinity

```
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/e2e-az-name
            operator: In
            values:
            - e2e-az1
            - e2e-az2
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: another-node-label-key
            operator: In
            values:
            - another-node-label-value
  containers:
  - name: with-node-affinity
    image: k8s.gcr.io/pause:2.0
```

Node Affinity - References

https://medium.com/kokster/scheduling-in-kubernetes-part-1-node-affinity-b77c97556424

https://docs.openshift.com/container-platform/4.2/nodes/scheduling/nodes-scheduler-node-affinity.html

Lab Practice

- Create pod redis with label name “type” as “db”, image=redis
- Choose one of your nodes, and add a label “distype=ssd”
- Create a pod that is only scheduled on SSD nodes. (use two pods one for required and other for preferred)
- Create a service that uses an nodeport and points to a 3 pod cluster running nginx.
- List all objects in your namespace that has label “image=redis”

Lab Practice - Contd.

- Write a service that exposes nginx on a nodeport
  a. Change it to use a cluster port
  b. Scale the service
  c. Change it to use an external IP
  d. Change it to use a load balancer
- Get all worker nodes (use a selector to exclude results that have a label named 'node-role.kubernetes.io/master')

(Refer: https://kubernetes.io/docs/reference/kubectl/cheatsheet)

Agenda - Day 4

- Multiple Schedulers
- Monitoring
- Metrics Server
- Logging
- Lab Practice

Multiple Schedulers

https://kubernetes.io/docs/tasks/extend-kubernetes/configure-multiple-schedulers/

Monitoring

- Node resource utilization
- Pod container metrics
- Application metrics

Monitoring

- Metrics Server
- Heapster/InfluxDB/Grafana
- Prometheus/Grafana
- Dynatrace
- Datadog

Metrics Server

- https://kubernetes.io/docs/tasks/debug-application-cluster/resource-usage-monitoring/
- https://github.com/kubernetes-sigs/metrics-server

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

Commands

Kubectl top node

Kubetcl top pod

https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Logging

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

kubectl logs -f <container_name>

Lab Practice

1) Create a custom-scheduler called <yourname>-scheduler
2) Create a pod nginx which is to be scheduled using the custom scheduler created. Verify if the pod is scheduled using the custom-scheduler
3) Deploy metrics server by referring https://github.com/kubernetes-sigs/metrics-server
4) Identify the node with high CPU utilization and high memory utilization
5) Show metrics for a given pod and sort it by 'cpu' or 'memory'

Agenda - Day 5

- Update Deployments
- Commands and Arguments
- Environment Variables
- Lab Practice

Update Deployments

Deployment Strategies:

https://www.cncf.io/wp-content/uploads/2020/08/CNCF-Presentation-Template-K8s-Deployment.pdf

https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

kubectl create

kubectl get deployments

kubectl set image --record

kubectl rollout status

kubectl rollout history

kubectl rollout undo

Commands and Arguments

https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/

https://phoenixnap.com/kb/docker-cmd-vs-entrypoint

Environment Variables

https://kubernetes.io/docs/tasks/inject-data-application/distribute-credentials-secure/#define-container-environment-variables-using-secret-data

https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/#define-container-environment-variables-using-configmap-data

Lab Practice

1) Create a busybox deployment with 2 replicas and which has “sleep 30” as arguments.. Identify the deployment strategy which is used by default
2) Scale the busybox deployment to 4 replicas and observe the rollout status.
3) Update the busybox deployment to use busybox:1.33 image and record the change so that its visible in rollout history
4) Undo the busybox rollout to version 2
5) Print the env for the busybox deployment