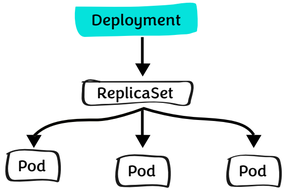
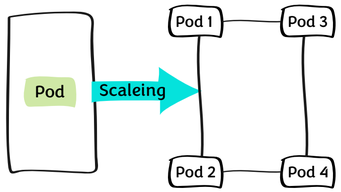
Google Cloud Platform

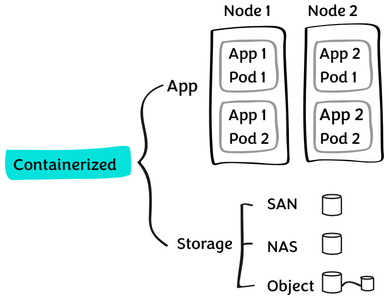
Kubernetes Engine

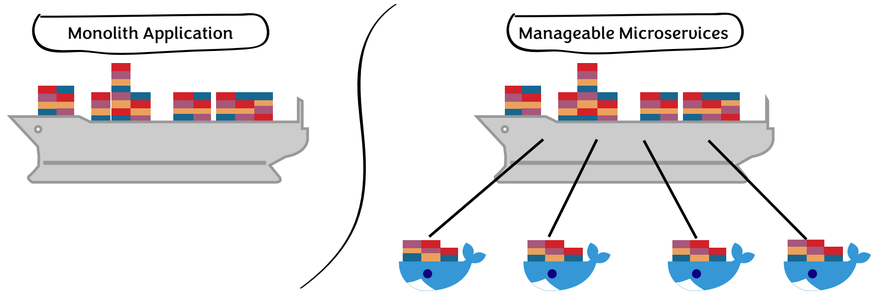
udemy GCP Gurus

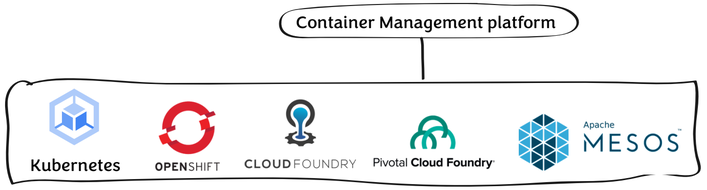
Kubernetes : Kubernetes (K8s) is an open source system to deploy, scale, and manage containerized applications anywhere.



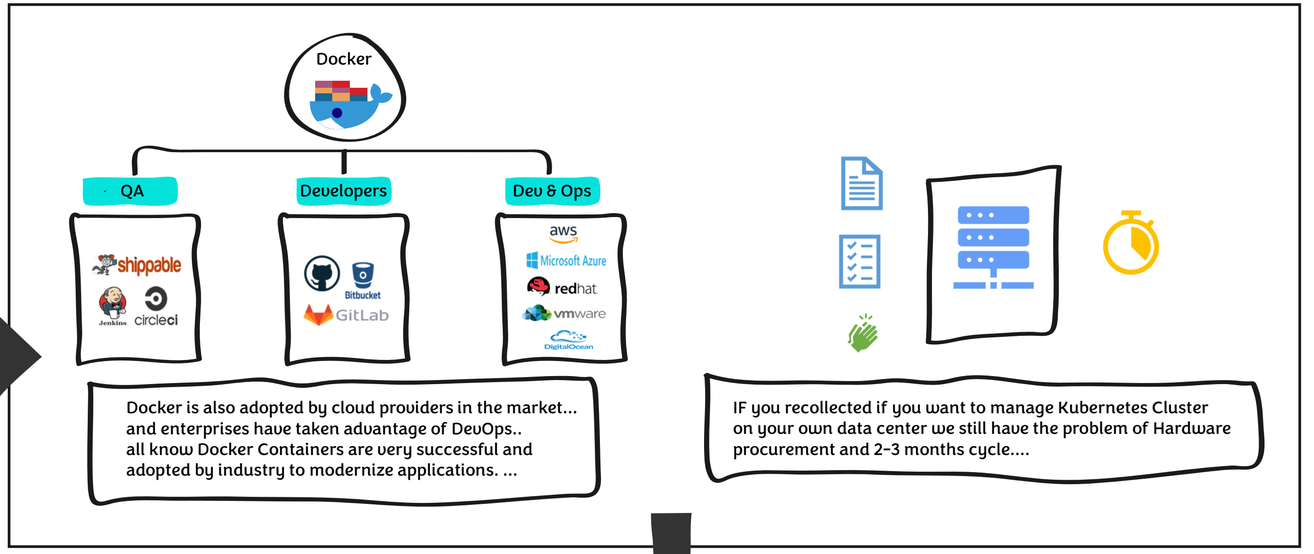


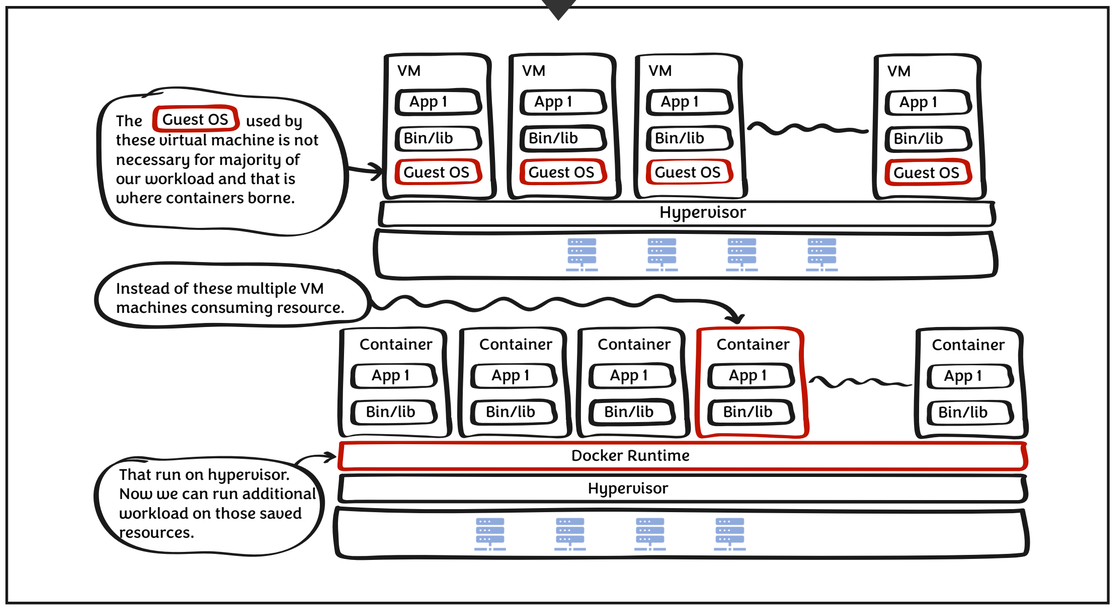




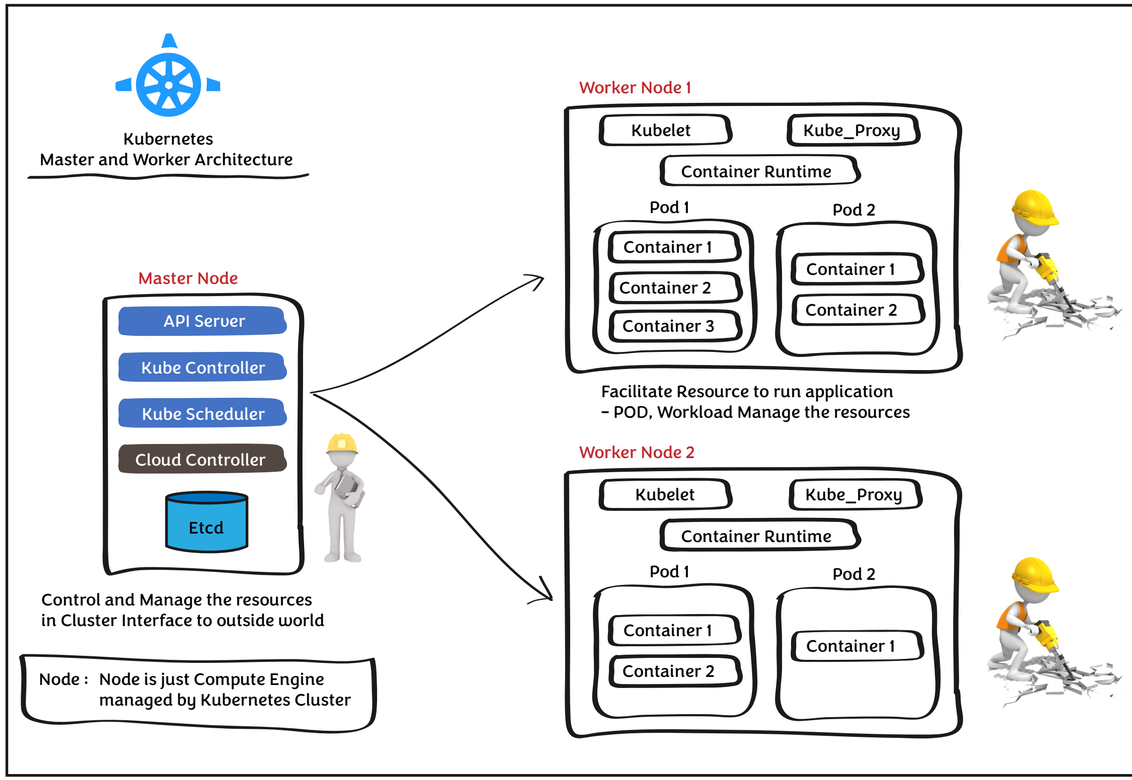


Today Kubernetes is most successful container management platform in ecosystems along with others.





Kubernetes
Master and Worker Architecture



API Server
1. Interface to Cluster
2. Inter Cluster Component communicate
3. Restful APIs Communications
   (POST, PUT, GET, DELETE, PATCH)
4. CLI and Direct API connect

Kube Controller
1. Workload, App, Pod Schedular
2. Watches for newly created POD
3. Identify Node that can run the workload
4. Run POD/s with help of Kubelet.

Kube Scheduler
1. Controller manage State of Cluster using API server.
2. Contains Multiple Controller Components in one Binary
3. There are build in controllers and you can build custom too..

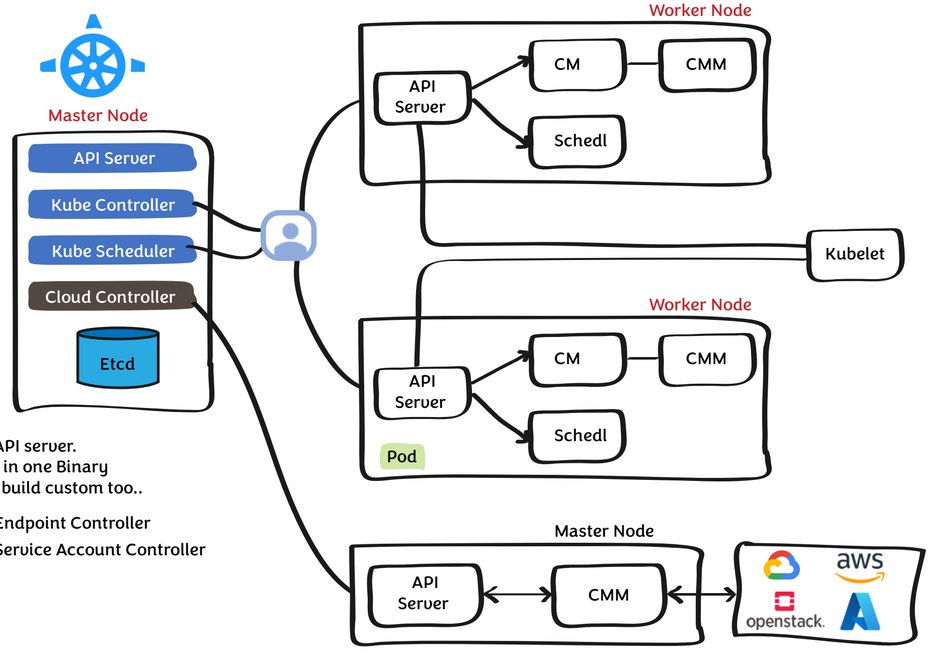
Examples :
- Node Controller
- Replication Controller
- Job Controller
- Endpoint Controller
- Service Account Controller

Many more...Build in

Cloud Controller
CCM Interacts with cloud API's (Public/Private) to manage Kubernetes Cluster

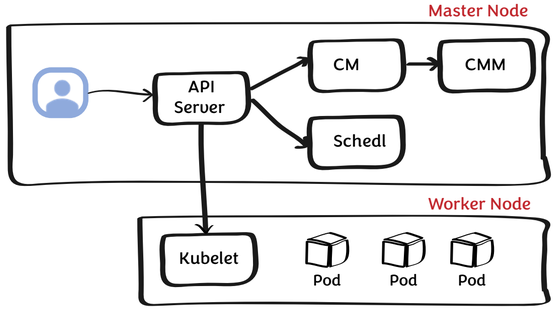
CCM helps to manage Kubernetes seamlessly with underline cloud provider for infra requirements

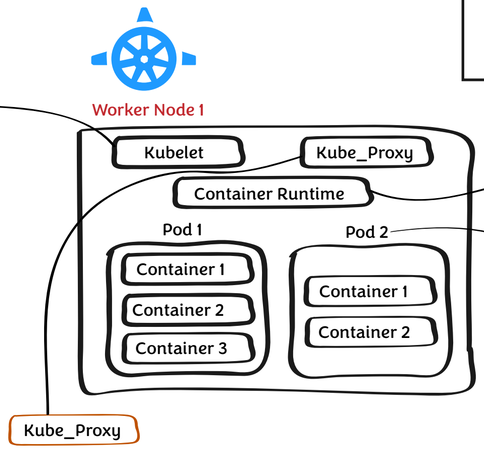
Examples :
- Node Controller
- Route Controller
- Service Controller
- Etc.



Etcd is a key-value data store used to store and manage the critical information that distributed systems need. It provides a reliable way of storing the configuration data. In this post, we will see a close look at etcd, why it is needed, and how to access its contents in Kubernetes.

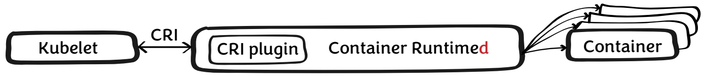
Kubelet
Kubelet manages nodes resources and works as co-Ordinator for API server.
It make sure Pod and Container inside Pod is running. It checks the Pod specifications to validate how container should be running make sure its healthy.





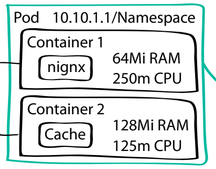
Kube_Proxy
Kube-proxy manages networking components for Pod (workload/app
- Kube-proxy manages network rules on Node
- Helps POP communication between inside and outside Kubernetes cluster.

Container Runtime and CRI
You can think of – Software needed to run Containers ... Applications.
CRI provides interface between Kubelet and Runtime.



Kubernetes run Pod as application or workload
i.e. Pod is smallest unit that is managed in Cluster

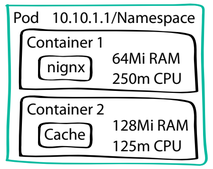
Pod Contains
1. One or more Containers
2. Has unique IP address



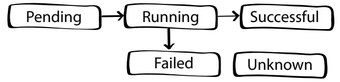
Containers in Pod
1. Share IP address
2. Namespace like localhost
3. Volume if any

Pods, Deployments & Replicaset

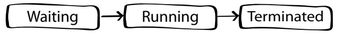
Pod and Container Lifecycles



Pod Lifecycles



Container Lifecycles

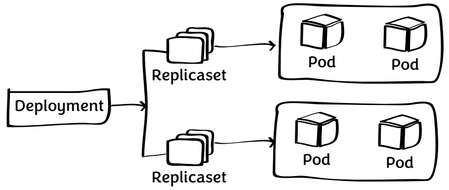


Deployment

Deployment in Kubernetes describe “Desired state of object”.

Deployment controller try to achieve desired state from current state with controlled rate

Deployment is “Way to manage your application”

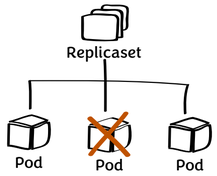


Application Management

1. Create Deployment to create rollout Replicaset.
2. Declare new state of Object.
3. Rollout new release or rollback to earlier version.
4. Pause rollout.
5. Scale up and Scale down.
6. Cleanup Replicaset.

Replicaset

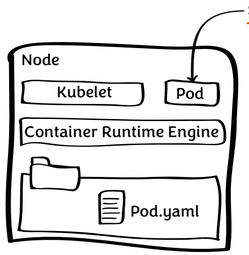
You also “Link” and “unlink” Pod based on need.



If this happens – Replicaset always try to maintain desired replicas of Pod.

Static Pod

Static Pods are managed by Kubelet Daemon without API server.



Static Pod running on the Node are visible to the API server but cant be controlled by API server

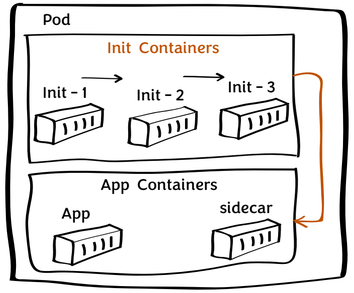
There is Mirror Pod for each static Pod

Init Containers

Containers that run before application containers.

You can setup init containers for utilities or initial setup scripts if can not be included in app image.

Resource Requests are handled differently.

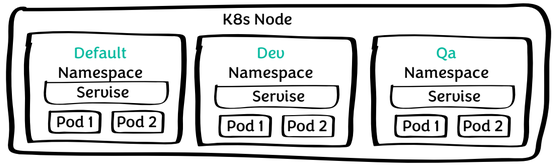


Objects & Namespace

Namespace

Namespaces are used in Kubernetes to isolate objects in single cluster

Name of the resources should be unique in NS but not across clusters



There are system NS & Default NS

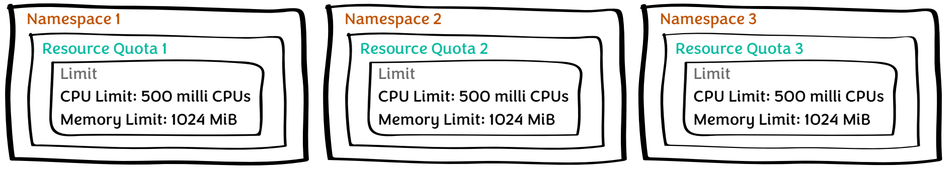
- Applicable only to namespaced objects e.g. Deployments, seruices
- Not applicable for cluster scoped resources e.g. Storage Class, Node, PV

Resource Quota :

Kubernetes allows us to configure resource Quota.

Limit :

Limit the "testing" namespace to using 1 core and 1GiB RAM.
Let the "production" namespace use any amount.



Request : Minimum committed by cluster

Limit : Max Allowed by namespace resources.

Resources can be : Memory Size , Number or % of CPUs, Number of objects etc

Example:

Namespace Resource Quota

Namespace: production

Quota Limits:

- Maximum Pods: 10
- CPU Requests: 2 CPU cores
- Memory Requests: 4GiB
- CPU Limits: 4 CPU cores
- Memory Limits: 8GiB

Objects

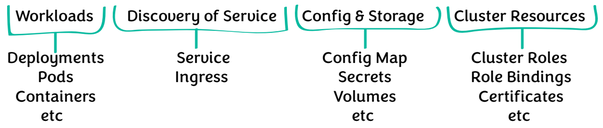
Pod is not the only objects in Kubernetes ...
there are many that helps Kubernetes cluster run itself as well as workload.

Kubernetes objects are persistent entities in the Kubernetes system !!!

Kubernetes Objects describes

1. Applications are running (and on which nodes)
2. The resources available to those applications
3. The policies around how those applications behave,
4. such as restart policies, upgrades, and fault-tolerance

Objects Types



Objects Managements Commands

Imperative Commands

```
kubectl create deployment nginx
-image nginx
```

Imperative Object Config

```
kubectl create
-f nginx.yaml
```

Declarative Object Config

```
kubectl apply
-f configs.yaml
```

Labels & Selectors

Label

Labels are key/value pairs that are attached to objects (Deployment, POD, Namespace etc)

Selectors

- Using labels Clients or Users can identify a set of objects
- Core Grouping of objects in Kubernetes
- You can have Equality based Label Selectors as well as Set Based

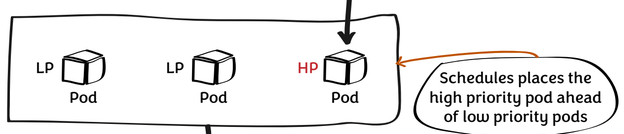
Application Lifecycle

POD Lifecycle and Priority

Priority Values

Pod Priority - 1,000,000,000 or Lower
system-node-critical- 2000001000
system-cluster-critical- 2000001000

Scheduling Queue



Schedules places the high priority pod ahead of low priority pods

LP - Low Priority
HP - High Priority

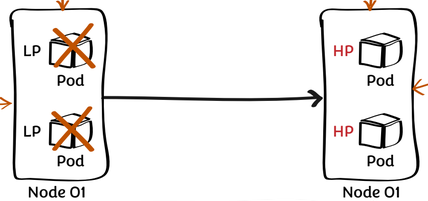
Scheduler starts preemptina low priority pode to schedule high priority pode

Does Cluster Noad Have Resources

No

Yes

Schedules pod without preemptina

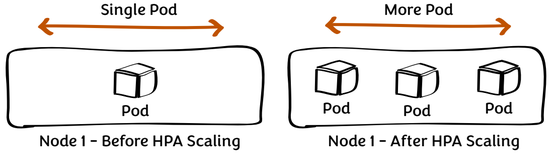


After evictina low priority pods, the high priority pods in the scheduling gets scheduled

Horizontal Pod Autoscalar

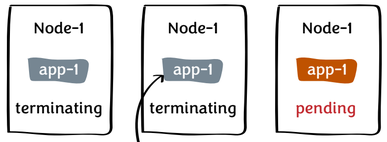
HorizontalPodAutoscaler automatically updates a workload resource to match the demand.

HPA is implemented as Kubernetes APIs and Controller resource



Pod Disruption Budget

Pod Eviction and termination of Pod happens voluntary or involuntary



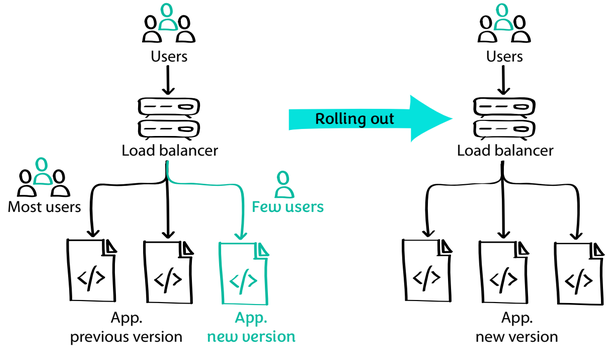
block termination until app-3 becomes redy

Rolling updates and constraints

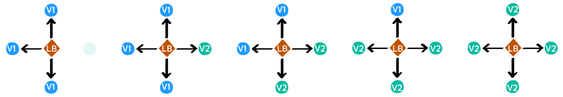
Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones.

Rollback policy:

1. Recreate : Recreate will kill existing Pods and create new pods.
2. Rolling Updates : Rolling updates will update Pod one by one - Means update one at a time.



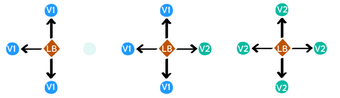
Rolling Updates:



Blue Green Deployment

Canary Deployment:

A canary deployment consists of routing a subset of users to a new functionality, Based on confidence or applications telemetry, new versions will gradually take complete load.



We creates two deployments with same labels. Replicasets are used to controls number of Pods of each version running.

A/B Testing - Canary Application

A/B Testing really checking business decisions on statistics rather than deployment strategy. canary deployment consists of routing a subset of users to a new functionality

Scaling Deployment

There are multiple ways you can scale Pods / Deployment / Application if cluster has resources.

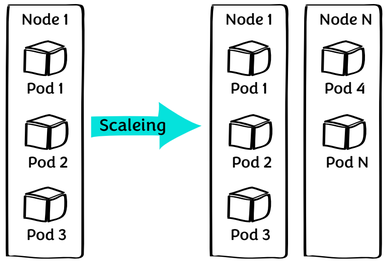
High level - There are two methods -> Manual and Automatic

Manual Pod Scaling	Automatic (HPA)
kubectl scale deployment/nginx-deployment--replicas=10	kubectl autoscale deployment/nginx-deployment--min=10 --max=15 --cpu-percent=80

Cluster Autoscalar

GCP Kubernetes Engine use cluster autoscaler to add or remove nodes based on workload demand.

How aggressively you want Kubernetes Engine to scale up and scale down and rearrange pods?

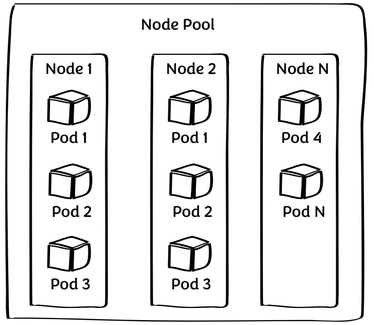


Autoscaling Profiles :

Balanced: The default profile.
Optimize-utilization: Prioritize optimizing utilization over keeping spare resources in the cluster. When selected, the cluster autoscaler scales down the cluster more aggressively: it can remove more nodes, and remove nodes faster.

Node Pool

Node Pool in Kubernetes Engine on GCP configures group of nodes (as part of managed instance group) to managed it together. It allows us to configure autoscaling properties
e.g. min and max size of Node pool.



Services

Clients want to connect directly to one or two pod

Nodeport exposes the service on each node IP at static port.

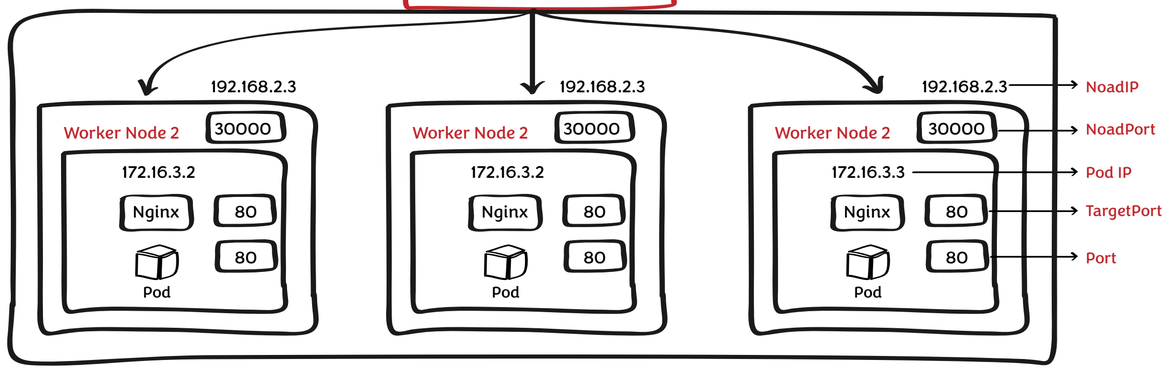
Cluster IP is default Access only available within cluster

Service is accessible via LoadBalancer and outside cluster.

This type of Service maps a Service to a DNS name by using the contents of the externalName field by returning a CNAME record with its value

HeadLess | NodePort | ClusterIP | LoadBalancer | ExternalName

Service



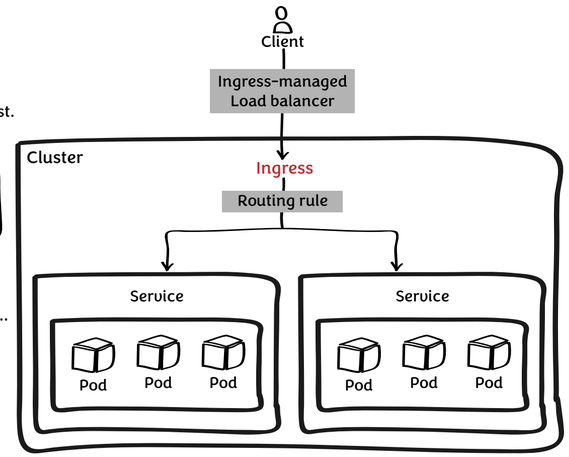
Ingress

Loadbalancer is associated to IP address and if you need to expose multiple services – you need to have multiple load balancer and hence increase the cost.

We can not use Nodeport in productions.

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster.

Traffic routing is controlled by rules defined on the Ingress resource.

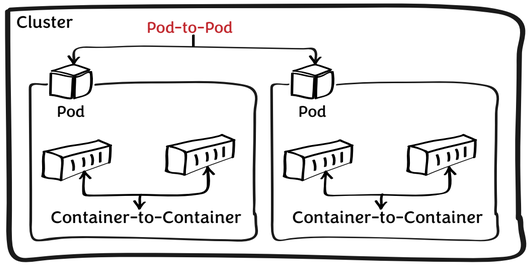


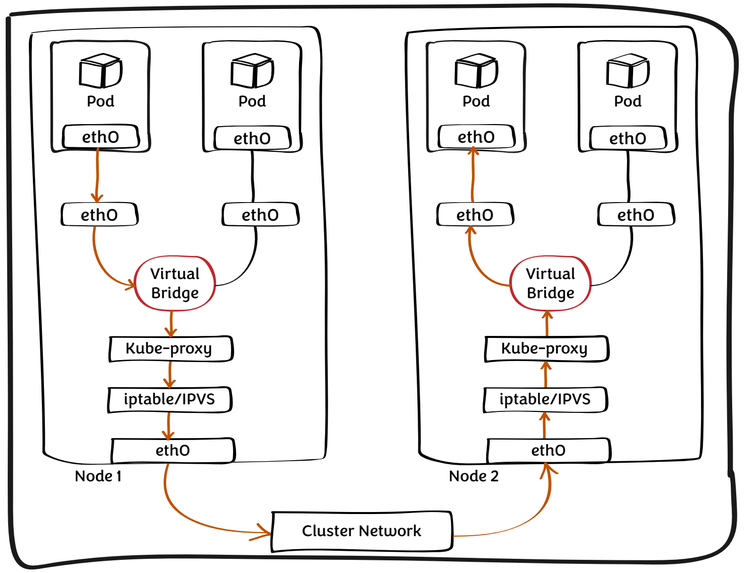
Ingress Controller

- Ingress Rerouce does not do any request forwarding...
- Ingress Controller does all the heavy lifting.. .
- Watches API Sever for change in ingress resources
- Deployed as Pod and daemonset or normal deployment is used to deploy ingress object
- Not Started automatically with cluster.
- GCP uses nginx controller in GKE

Networking

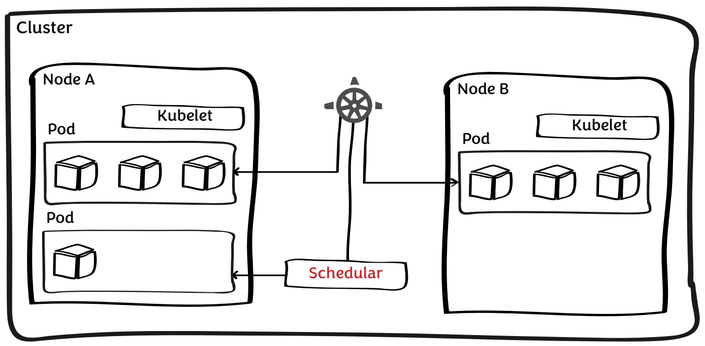
- Highly-coupled container-to-container
- Pod-to-Pod
- Pod-to-Service - Already Covered
- External-to-internal - Already Covered





Schedular

The Kubernetes scheduler is a control plane process which assigns Pods to Nodes .



Kube Scheduler

1. Identifi Unscheduled pods

Pod
Panding state

Node 1 cpu - 8 | Node 2 cpu - 10 | Node 3 cpu - 15

Pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
    name: <pod_name>
          :
          :
spec :
    containers:
      - name: <container_name>
        image: <image>
        resources:
            requests:
              cpu: 10
```

2. Node Selection is based on 2 step operation

1. Filtering :

Filtering out Node-1 cpu < 10)

Node 1 cpu - 8 | Node 2 cpu - 10 | Node 3 cpu - 15

Filtering

2. Scoring :

Node 1 cpu - 8 | Node 2 cpu - 10

Win

3. Binding :

Choose the Node with the highest score

Node 2
Pod

Pod.yaml

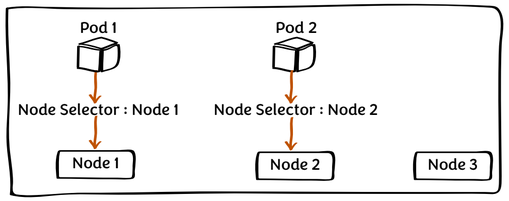
pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
    name: <pod_name>
          :
          :
    containers:
      - name: <container_name>
        image: <image>
        resources:
            requests:
              cpu: 10
```

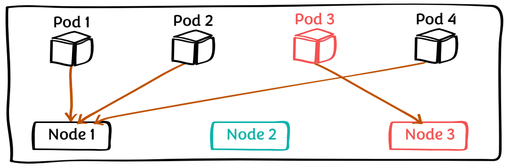
update nodename with the selected node → nodename: node 2

Kube_schedular does scoring and filtering provides other criteria are met ?

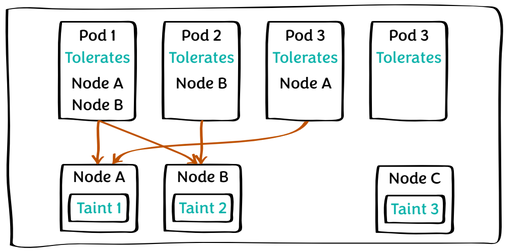
1. Node Selector :



2. Affinity:



2. Taint/Toleration:



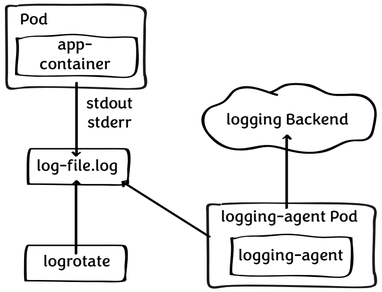
Logging and Monitoring

1. POD/Container Logging

Logging and Monitoring is essential of activity to keep application running in production

Applications container runs in POD as well as System components also runs in POD(container)

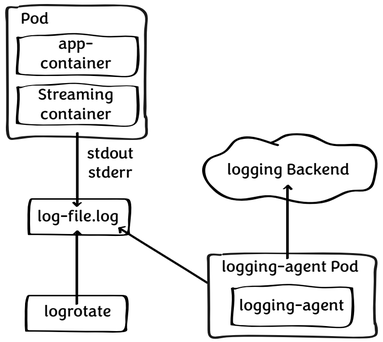
Kubelet and Container runtime writes to journalD



2. Node Logging

All stdout and stderr is handled and redirected by container engine

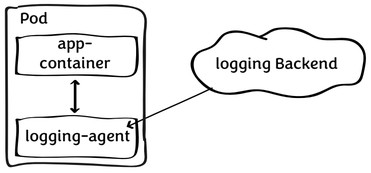
Kublete on node handles request for all the logs on node



GCP - Kubernetes Engine

Nodes and Pods are ephemeral and we may loose logs if node died by autoscaler or any other unplanned activity

Enable Logging and Monitoring of Kubernetes Engine Cluster.



StatefulSets, Jobs and DaemonSets

1. StatefulSets

Not all the workload that you run is stateless and have flexibility to scale up and scale down and Schedule as and when we need.

StatefulSets

Node 1 Pod 1 Pod 2
Node 2 Pod 3
Node 3

Not all the workload that you run is stateless and have flexibility to scale up and scale down and Schedule as and when we need.

2. DaemonSet

What we can do - if we need to run one application on all nodes.. ?

- 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.
- While Creating - Define highest priority so that Pods are not evicted.

Uses of a DaemonSet are:

- Running a cluster storage daemon on every node
- Running a logs collection daemon on every node
- Running a node monitoring daemon on every node