Kubernetes

- Opensource container orchestration tool
- Helps us to manage containerized applications in diff deployment env.
  Eg:- Phy, virtual, cloud, Hybrid.

- Need of orchestration tool.
  - Trend from monolith to microservice.
  - So, increased usage of containers.
  - So management of this containers in single App, using scripts was difficult.

- Features of orchestration tool
  - High Availability or no downtime
  - Scalability or high performance
  - Disaster recovery or backup & restori.

→ Shortform of Kubernetes cluster

K8 components

Pod:
- Smallest unit of K8s
- Abstraction over container
- Usually 1 app per pod.
- new IP on re-creation.
- Each pod gets its IP addr.
- ∴ Pods have Pvt network.

Service:
- permanent IP addr
- Lifecycle of Pod & service is not connected.
- Can act as load balancer.
- External service:- service which can be accessed from browsers.
- Internal service:- service used inside App. Like DB.

Ingress:
- it converts simple http url to url for pod/service. (i.e. add IP & Port)
- first request is sent to ingress, & then it forwards it to service.

config map:
- external config of our app.
  - suppose we need to set url of DB, then, when we change it, we again build docker image, push it, pull it & run it.
- it saves us from above process.

Secret:
- used to store ~~secte~~ secret config data.
- like user name & pwd for DB., certificates.
- it is base64 encoded.

volume:
- way to persist data by attaching volume/path/storage.

deployment:
- Blue Print to scale up/down pods for stateless apps.

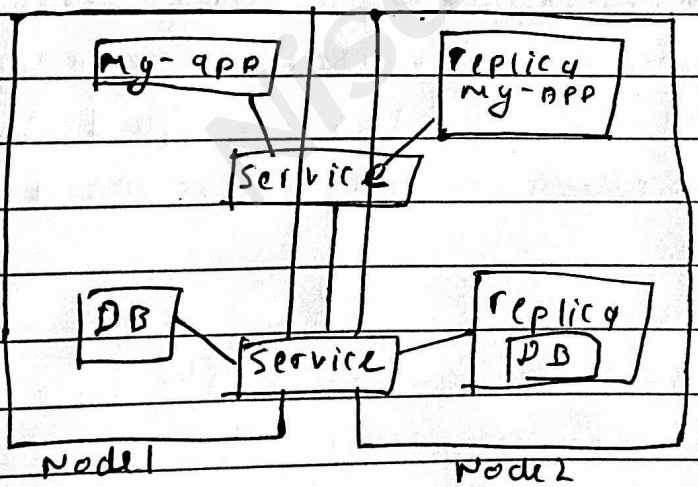
StatefulSet:
- used to scale stateful apps.

Data storage

- Volume → storage to store data generated by Pod.
  - → Local.
  - → Remote.

Deployment:

- Suppose a Pod crashes, in so to avoid downtime we will replicate Pods and connect it to common pods with service. (so that IP stays same.)
- Deployment is a blueprint for Pods.
- it is an abstraction over Pods.
- We work with deployment.
- Using This deployment we can scale up or down.



- But here we cannot create replica of DB Pod as it has a State i.e. it generates data & stores it.

- So we need a shared datastorage which can be accessed by replica of DB. & maintain consistency.
- this service is provided by statefulset.

Kubernetes cluster (K8s) Architecture

- worker machine / Node
  - Actual machine, doing computations.
  - Each Node has multiple Pods.
  - 3 Processes must be installed on it.
    - container runtime (Eg:- docker)
    - Kubelet (used to schedule Pods & interacts with container & Node)
    - Kubeproxy (used to forward incoming request)

How to interact with cluster?
- How to schedule Pod?
- monitor it?
- Reschedule / restarts Pod?
- join / Add new node?

=> it is Managed by Master Node.

- Master Node
  - it controls All worker nodes & manage Above Tasks.
  - services:
    - API server -> cluster gate way.
      - when deploying new app we interact with it using client (like kube.. dashboard, kubectl etc)
        - Responsible for Authentication.
        - Receives ~~Req~~ Initial Request.

L) Scheduler:
  L) After request is Authenticated, it is sent to scheduler to schedule execution of pod on a worker node.
  L) checks for resources required, & Allocates pod to node with less resource utilize.

L) Controller manager:
  L) Detect cluster state change like death of pod.
    L) then request it to scheduler & restarts pod.

L) etcd:
  L) Key value store of cluster state.
  L) cluster brain, as every thing is logged in key value format & other master node services depends on data in etcd.

- in small K8s
  - 2 master
  - 3 worker
    - API server is load Balanced
    - etcd is stored in distributed manner.

minikube

- Open source tool, used to test our APP.
- it creates 1 Node K8s cluster.
- Node runs in virtual env.

- Kubectl
  - way to interact with K8s.
  - it communicates with 'API server' process.

K8s namespaces

-> What is namespace?
- A cluster organizes resources in a namespaces
- It is like virtual cluster inside a cluster.

-> There are 4 namespaces per Default.
- ~~Kubernetes-dashboard~~
  - only with minikube
- Kube-system
  - Do Not create or modify it.
  - Not for our usage
  - System processes, master/management and kubectl processes.
- Kube-Public
  - Publically accessible data.
  - A config map which contains cluster info.
    - Kubectl cluster-info
- Kube-node-lease
  - Each node has lease obj associated in namespace.
    - determines the availability of a node.
- default
  - resource created by us are located Here. if we do not create separate namespace.

Need For Namespace

- Group Resources in namespace. (Eg: logging in one, monitoring in other)
- To remove conflicts among teams.
  - suppose 2 teams deploy APP with same name in a cluster, Then one will override other.

- Resource sharing
  - i.e. duplog common resource only once.
  - in Blue/Green deployment
    - Blue → App currently in use (Production)
    - Green → App which will be in future Production.
    - Both of this can use common resources.

- Access and Resource Limits
  - Provide a team Access to its namespace with cluster & no other N.S.
  - Limit Resources used by namespace
    - Like CPU, RAM, storage

- Config map, secrets cannot be shared even if common.
- Service is a sharable resource.
  - Eg:- "db-url: service-name.namespace-name"
- Volume & Node Lives globally i.e. you can not keep it in N.S.

"Kubectl api-resources --namespaced = false [true"

Namespace can be created by command & can be added to component by "-n" option.

OR

Can write it in Yaml. in metadata.

By default Active namespace is "default."

To change Active n.s.

- winget install kubectx
- kubens
- kubens namespace-name

K8s Ingress

- To make app available to public/Browser, we create external service.
- Now app is accessible using IP:Port.
- but Instead it should be url with human readable name.
- so we use Ingress service.

- url -> Ingress -> app-service -> pod

:-

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: #-app-name
spec:
  rules:      # routing rules
  -host: url.com      # url entered by user
                      ↳ valid domain addr.
  http:
    paths:
    - backend:      # means, when entered above url
                    forward it to below service.
      serviceName: name-of-service
      servicePort: portNum
```

↳ for host, map domain name to node's IP which is entry point

- To run Ingress successfully, we need Implementation of Ingress called Ingress Controller.

- Install Ingress controller
  - ↳ evaluates & process routing rule.
- it is a collection of Pods which run on ~~ow~~ our node.
- manages redirection.
- Many third party impl.
  - ↳ k8s Nginx ingress controller.

Consider ENV where Ingress will be running.

- → Cloud
  - → AWS
  - → GCP
  - → Azure
  - → Linode
- → Baremetal
  - → config your own entry point
  - → sw/hw soln (Proxy server)