Kubernetes

- Kubernetes is an Open-Source Container management tool that automates container.
- It schedules, runs and manages isolated Container that are running on virtual/ physical/cloud machines.
- All top cloud providers support K8s i.e GKE (Google K8s engine)
  AKS (Azure K8s Service)
  Amazon EKS (Elastic K8s Service)

- Open Source Container Orchestration tool By Google used to Multi Environment, Multi Container Deployment

Why do we use kubernetes?

- For production ready deployment of Micro-services and Small apps.
- Having less failures & Downtimes
- Backups and restores :-

DD MM YY

Problem with scaling up the Container:

- Container can't communicate with each other.
- Autoscaling and load balancing was not possible.
- Containers had to be managed carefully.

Features of k8s:

- Orchestration (clustring of ang number of containers running on a different network.
- Autoscaling (vertical & Horizontal)
- Autohealing
- Load Balancing
- platform Independent (cloud/virtual/physical)
- Fault Tolerance (Node/POD/Failure)
- Rollback (Going back to the previous Version)
- Health monitoring of containers
- Batch Execution (:one time, Sequential, parallel).

Kubernetes Installation tod:

(i) Minicube
(ii) Kubeadm

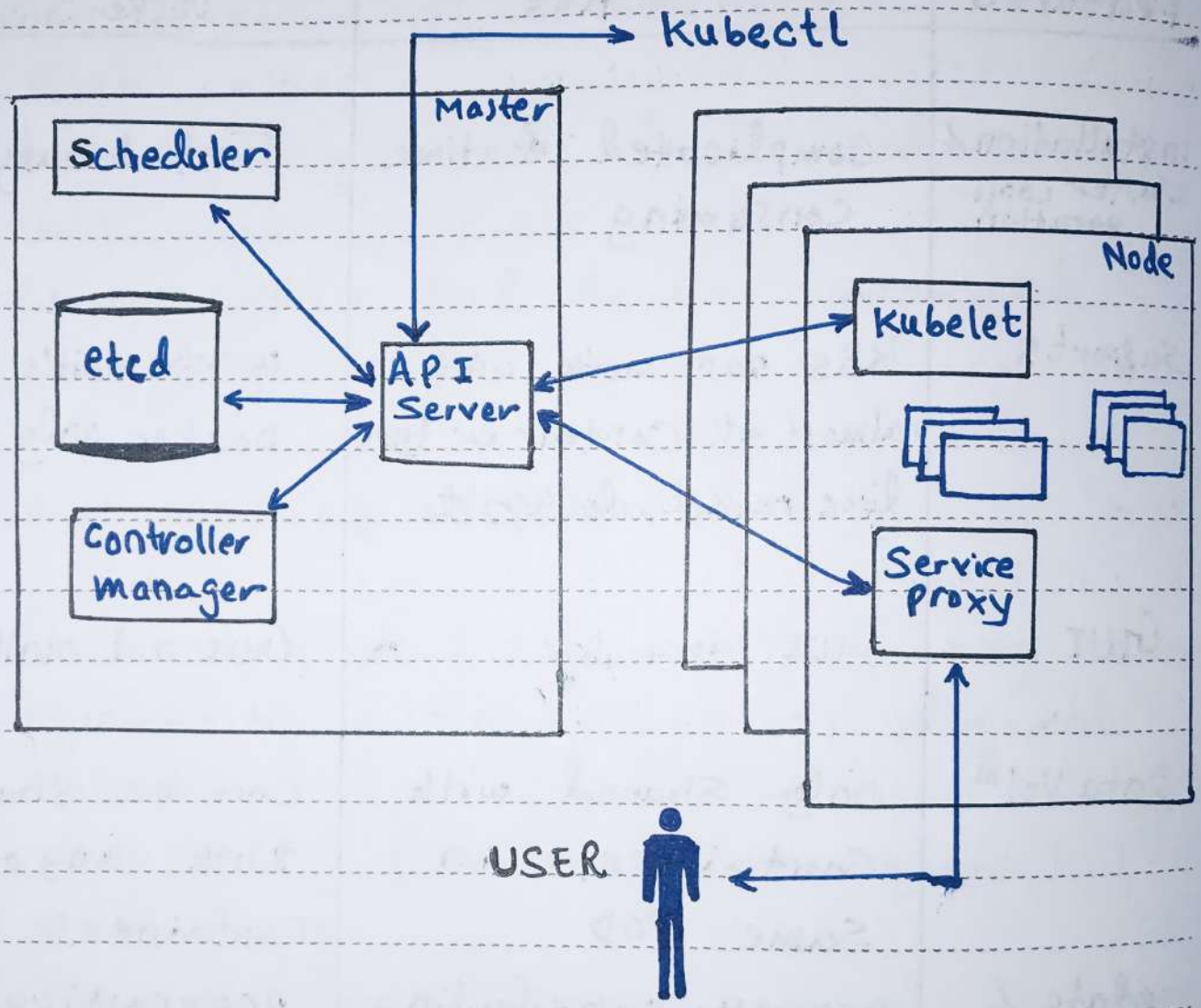
DD / MM / YY

TVS ELECTRONICS

Features	K8s	Docker Swarm
Installation & cluster configuration	Complicated & time consuming	Fast & Easy
Supports	K8s can work with almost all Container type like rocket, docker, etc	work with Docker only
GUI	GUI available	GUI not available
Data Volm	only shared with Containers in same POD	Can be shared With any other Containers
Update & Rollback	process scheduling to maintain Services while updating	progressive update & service helth monitoring throughDut update
Autoscaling	Support vertical & horizontal autoscaling	Not Support autoscaling
logging & monitoring	Inbuilt tool present for monitoring ✓	used 3rd party tool like Splunk

DD MM YY

Architecture of Kubernetes:



Working with Kubernetes:

- we create a Manifest (.yml) file
- Apply those to cluster (to master) to bring it into the desired state.
- POD runs on a node, which is controlled by the master.

DD / MM / YY

TVS ELECTRONICS

Role of Master Node:

- Kubernetes cluster contains containers running or Bare Metal / VM instances/ cloud all mix
- Kubernetes designates one or more of these as master and all other as workers.
- ~~Kubernetes can be~~ .
- The master is now going to run a set of k8s process . These process will ensure the smooth functioning of cluster. These process are called "Control plane"

- Can be Multi - Master for high availability.
- Master runs control plane to run cluster ~~availability~~ Smoothly.

Components of Control plane (master):

1. Kube- API Server
2. Etcd
3. kube-Scheduler
4. Controller manager

Akhlaqur R.

/ /
DD MM YY

TVS ELECTRONICS

1. kube-API Server:

- It is used to authenticates user, validate requests, Retrieve Data, update ETCD communicate with other Components of Cluster.
- API-Server interacts directly with user (i.e we apply .yml or json manifest to kube-apiserver)
- It is meant to scale automatically as per load.
- It is front-end of Control-plane

2. Etcd:

- Contains all the information related to Node, PODS, configs, Secrets, accounts, Rolls, Bindings etc.
- It Store meta data and Status of Cluster
- Etcd is Consistent and high available Store (key-value store)

DD MM YY

Etcd has following features:

1. Fully replicated:
   The entire state is available on every node in the cluster.
2. Secure:
   Implements automatic TLS with optional client-certificate authentication.
3. Fast:
   Benchmark at 10,000 writes per second.

3. Kube Scheduler: (action)

- Responsible for scheduling the pods on the nodes.
- It just decides which node based on the CPU, RAM, resources on the node
- when users make request for the creation and management of PODs, Kube scheduler is going to take action on these requests
- Handles POD creation and management

DD MM YY

- Kube scheduler match/assign any node to create and run PODS

- A scheduler watches for newly created PODs that have no node assigned. For every POD that the scheduler discovers the Scheduler becomes responsible for finding best node for that POD to run on

- Scheduler gets the information for hardware configuration from configuration file and schedules the PODs on Nodes accordingly.

4. Controller Manager:

- Continously monitors various Components of cluster and works towards managing / restoring to the desired state.
- Make sure that actual state of cluster matches the desired state

DD MM YY

- Two possible choice for Controller manager:

(a) If k8s on cloud, then it will be Cloud-controller-manager

(b) If k8s on non-cloud then it will be kube-controller manager.

Components on master that runs Controller:

(a) Node Controller:

- For checking the cloud providers to determine of a node has been detected in the cloud after id stop responding.

- Communicates with kube api server and manages node [every 5 sec]

- Checks again for 40 seconds then marks as "unreachable"

DD MM YY

(b) Route controller:
Responsible for setting up networks, routes on cloud.

(c) service Controller:
Responsible for load balancers on your cloud against services of type load balancers.

(d) Volume Controller:
For creating, attacbing and monitoring volumes and interacting with the Cloud provider to orchestrate volume.

(e) Replication:
- Responsible for monitoring Status for replica set.
- Ensures that desired no. of pods are available at the required time.

DD / MM / YY

TVS ELECTRONICS

Node is going to run 3 important pieces of Software/process:

1. Kublet
2. Container engine
3. Kubeproxy

1. Kublet:

- is on the worker node and registers the Node with the pod.
- Monitors the Status of Pods and reports to the Kube api-servers.

- Need to install Kubelet on worker Node:
- Agent running on the node
- Use Port 10255
- Send success/fail reports to manifest

2. Container Engine:

- works with Kublet
- Pilling images
- Start/stop Containers
- Exposing container on ports specified in manifest:

DD MM YY

3. kubeproxy:

- Pod network allows to connect/ communicates pods for each other by Pod IP
- Kube proxy runs on each node, using IP Tables rules so that any Service can connect to pod from outside:-

* Kubectl → Single cloud
  Kubeadm → on premise
  Kubefed → federated.

DD MM YY

TVS ELECTRONICS

POD

- All containers in Kubernetes are contained within pods. The simplest component of the kubernetes architecture is a pod.

- POD is a group of one or more containers that are deployed together on the same host.

- A cluster is a group of nodes.
- A cluster has at least one worker node and master node.

- In kubernetes the control unit is the POD, not Containers.

- In consists of one or more tightly coupled containers.

- POD runs on node which is control by master.
- Can't start containers without a POD.
- One POD Contains one Container →

DD MM YY

TVS ELECTRONICS

Multi Container PODs:

- Share access to memory space
- Connect to each other using local host <Container Port>
- Share access to the Same Volume
- Containers within POD are deployed in an all-or-nothing manner.
- Entire POD is hosted on the same node.

POD Limitations:

- No auto healing or auto Scaling
- POD crashes.

Higher level Kubernetes objects:

(a) Replication Set:
autoscaling & auto healing

(b) Deployment:
version & rollback

(c) Service

(d) Static (non-ephemeral) IP & networking

(e) volume: non-ephemeral storage.

DD MM YY

Each manifest has four necessary parts:

1. The version of the API in use
2. The kind of resource you'd like to define.
3. Metadata about the resource
4. Through not required by all objects, a spec, which describes the desired behavior of the resource.

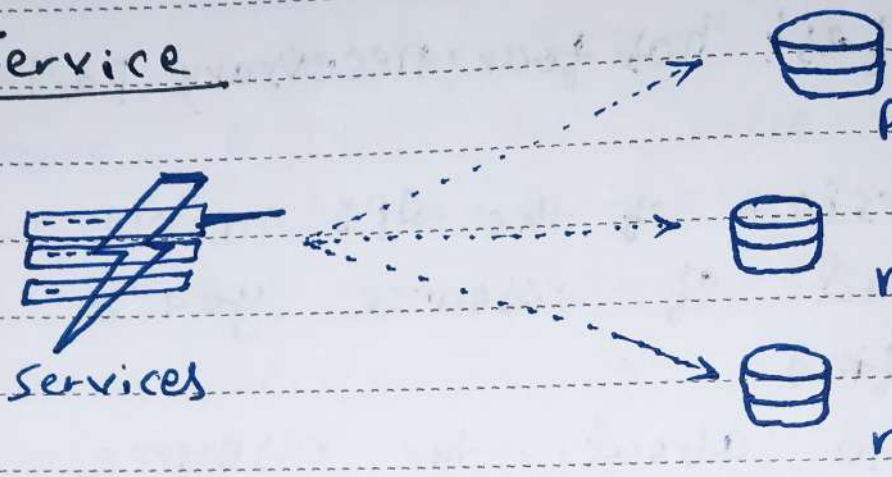
Pods 

Pods can have one or more Containers coupled together.
They are basic unit of kubernetes.
To increase High availability, we always prefer pods to be in replicas.

DD MM YY

TVS ELECTRONICS

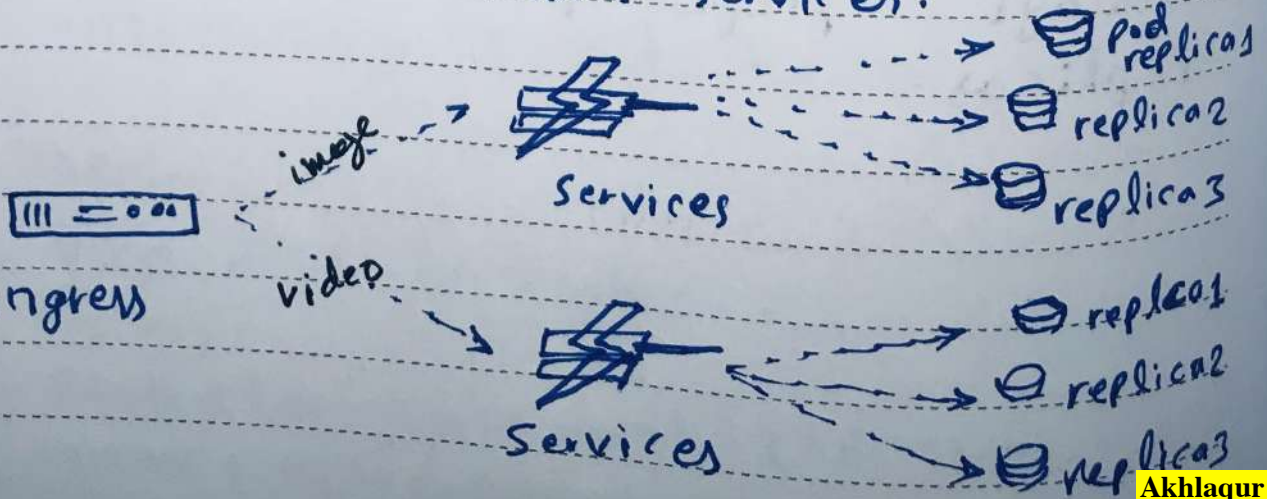
Service



are used to load balance the traffic among the pods.
is round-robin load balancer for all the pods

Ingress

is an object that allows access to your Kubernetes Services from outside the Kubernetes cluster. You configure access by creating a collection of rules that define which inbound connection reach which services.

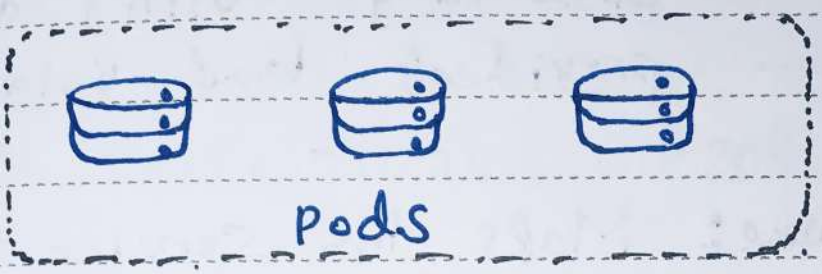


Akhlaqur R.

DD / MM / YY

TVS ELECTRONICS

Deployments in Kubernetes:



Deployment in k8s is a Controller which helps your applications reach the 'desired state', the desired State is defined inside the deployment file.

Creating a deployment
↳ kubectl create -f nginx.yaml

Services Types:

1. ClusterIP: exposes the service on cluster-internal IP

2. NodePort: Exposes the Service on each Node's IP at a static port

DD MM YY

TVS ELECTRONICS

3. LoadBalancer: Exposes the Service externally using a cloud provider's load Balancer.

4. ExternalName: Maps the Service to the contents of External Name:

To know the Port, on which the Service is being exposed.

↳ kubectl get svc nginx

DD / MM / YY

TVS ELECTRONICS

Kubernetes Objects:

- Kubernetes uses objects to represent the state of your cluster.
- what containerized applications are running.
- The policies around how those applications behave, such as restart policies, upgrades and fault tolerance:

- Once you create the object, the Kubernetes System will constly works to ensure that object exist and maintains cluster's desired state:

- Every Kubernets object includes two nested fields: that given the object config the object spec & the object status:

- All object are identified by a unique name and a UID

DD / MM / YY

TVS ELECTRONICS

Labels & selectors:

- Labels are the mechanism you use to organize kubernets objects.

- A label is a key-value pair without any predefined meaning that can be attached to the objects.

- Labels are similar to tags in AWS or git where you use a name for a quick reference.

- Multiple labels can be added to a single object.

Labels - selectors:

- unlike name/UIDs, labels do not provide uniqueness, as in general, we can expect many objects to carry the same label.

DD MM YY

- Once labels are attached to an objects, we would need filters to narrow down and these are called label selectors.

Node selector:

- One use case for selecting labels is to constrain the set of nodes onto which a pod can schedule.
i.e you can tell a pod to only be able to run on particular nodes.

- Generally such contstraints are unnecessary, as the scheduler will automatically do a reasonable placement, but in certain circumstances we might need it

- we can use labels to tag nodes.

DD / MM / YY

TVS ELECTRONICS

- you the nodes are tagged, So you can use the label selectors to specify the pods run only on specific moves :

- First we give a label to the node.
- Then use the node selector to the pod configuration :-

Scaling and Replication

- Kubernets was designed to Orchestrate Multiple constraints and replication.

- Need multiple Containers / replication helps us with these :-

Reliability :

- By having multiple versions of an application . you prevent problems if one or more falls :-

DD MM YY

TVS ELECTRONICS

Relationship betn these objects.

- Pod manages containers
- Replica set manage pods.
- services expose pod processes to the outside world.

- Configmaps and secrets help you configure pods.

Kubernetes object:

- It represents as JSON or YAML files.
- you create these and then push them to the kubernetes API with kubectl.

Kubernets objects Management

The kubectl command line tool supports several different ways to create and manage kubernets object.

DD MM YY

TVS ELECTRONICS

Load Balancing

Having multiple versions of a container enables you to easily send traffic to different instances to prevent overloading of a single instance or node.

Scaling

when the load does become too much for the number of existing instances, Kubernetes enables you to easily scale up your application, adding additional instances as needed.

Rolling updates:

- update to a service by replacing pods one by one.

DD / MM / YY

TVS ELECTRONICS

Replication Controller :

- A replication controller is an object that enables you to easily create multiple pods, then make sure that number of pods always exists :-

- If a pod is created using an RC - replication controller will be automatically replaced if they do crash, failed, or is terminated :-

- RC is recommended if you just want to make sure 1 pod is always running, even after the system reboots :-

- You can run the RC with 1 replica & the RC will make sure the pod is always running :-

DD MM YY

Replica Set :

- A replica set is a next-generation replication controller.

- The replication controller only supports equality-based selectors whereas the replica set supports set-based selectors.

- The replica set rather than the replication controller is used by other objects like deployments.