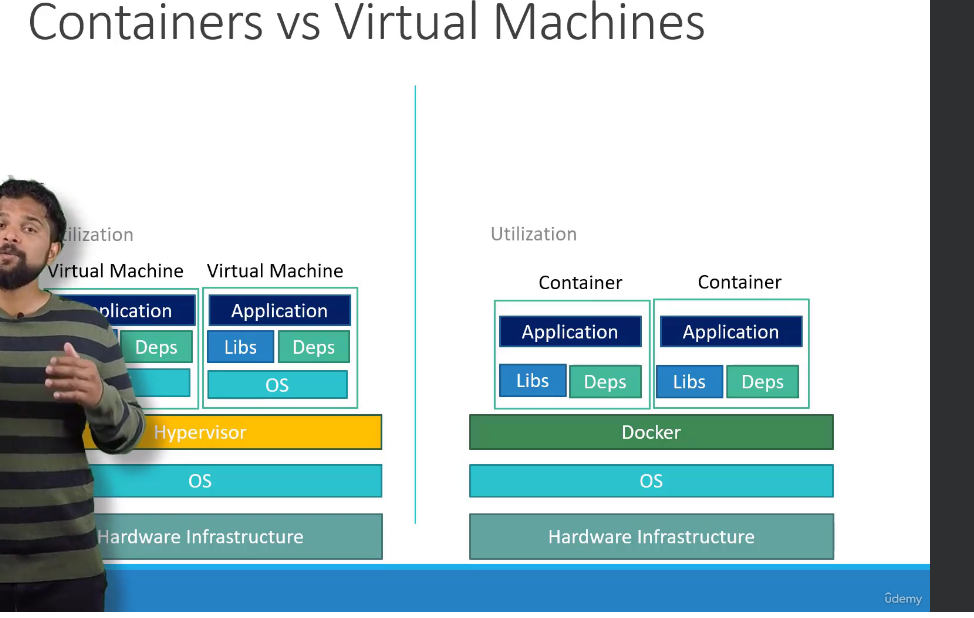
Docker

Containers are completely isolated environments, having there own processes/services, networking, memory(mounts).



Docker container are running instances of docker images

Docker registry

Container Orchestration – Automatically creating and managing containers

1. HA
2. Load balanced application
3. YAML based syntax

K8s Architecture

Node: Physical machine or VM on which k8s is installed , it is also called worker node.

Cluster : A set of nodes, if a node fails your application will still be available from other nodes

Master : It is another node with k8s installed on it, It is responsible for managing the cluster

1. It has information about members of the cluster stored
2. How nodes are monitored?
3. When a node fails how do you move the workload of the field to another worker node

Master node actual does the orchestration of the worker node.

K8s Components

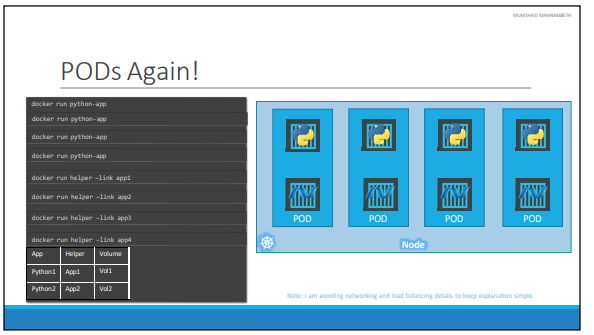
1. K8s API server – Act as frontend for k8s, all the users, CLI, managements devices connect to k8s.
2. Etcd service – Distributed key value store to store all data used to manage the cluster.
3. Kubelet service – Agent running on each node of the cluster.
4. Container run time – IT is used to run containers.
5. Controllers – Brain . The notice and respond when nodes goes down and bring up new containers.
6. Schedulers – Responsible for distributing tasks across multiple nodes.

A screenshot of a computer

Description automatically generated

Kubectl: Kube Command line tool

Kubectl tool is used to deploy and manage applications on a k8s cluster.



PODS: Single instance of an application, smallest object you can create in a k8s.

POD is 1:1 relationship with a container.

**Kubectl run nginx: Create a pod and deploys an instance of nginx docker image**

**Kubectl describe pod nginx**: Detailed information about the nginx pod.

Kubectl get pods -o wide: Details view of the running pods

**YAML:**

Key Value Pair

Fruit: Apple

Vegetable: Peas

Liquid: Water

Meat: Chicken

Array/Lists:

Fruits:

* Orange
* Apple
* Banana

Dictionary / Map

Banana:

Calories: 105

Fat: 0.4

Carbs: 27g

PODS Definition YAML components:

1. apiVersion: Version of k8s API used to create the k8s objects (v1, apps/v1)
2. Kind: Type of object we are trying to create (Pod, Deployment, Service, Replica set etc.)
3. Metadata: It has name and labels
4. Spec:

Replication Controllers: It help us to run multiple instances of pod in k8s cluster, thus providing HA.

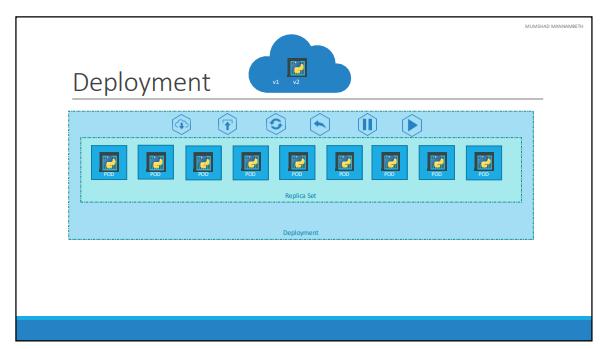
1. Load balancing – Across same and multiple nodes as well.
2. Scaling

Replication Controller: Old technology replaced by Replica set.

Selector key is available in replica set but not in replication controller.

**Replica set uses selector label to monitor the pods with matching label in the k8s Cluster**

Deployments : Rolling updates and rollbacks for the underlying pods, pausing and resuming changes.



When we first create a deployment, it triggers a rollout, a new rollout creates a new deployment revision

**Recreate Strategy:**  It will delete all the pods and then create the pods which application downtime



**Rolling Update: Default deployment strategy**

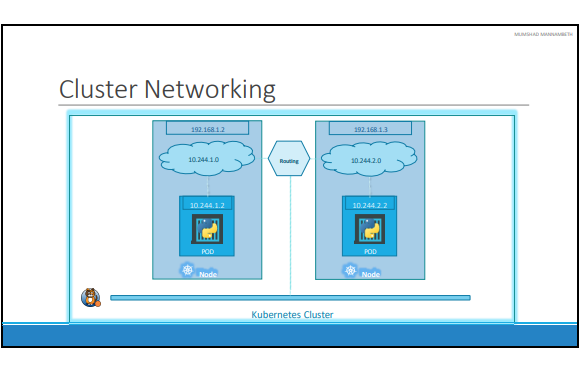
Recreate and Rolling Update logs has different messages

--record=true used to record the deployment history in kubectl rollout history command

IP address is assigned to POD rather than Docker container

All containers/PODS can communicate to one another without NAT.

All nodes can communicate with all containers and vice-versa without NAT.



K8s services enable communication between various components within and outside of the application. Services help us to achieve loosing coupling in microservices.

A computer screen shot of a diagram

Description automatically generated

Usecase of K8 Service:

1. To listen to a port on the node and forward requests on that port to a port on the port running the web application.
2. Above service is called Node Port Service.

* Node Port – Where the service make an internal port accessible on a port on the node.
* ClusterIP – The service creates a virtual IP inside the cluster to enable communication between different services such as frontend and backend servers.
* LoadBalancer – IT would distribute the load across the different web servers in your frontend.

A screenshot of a computer

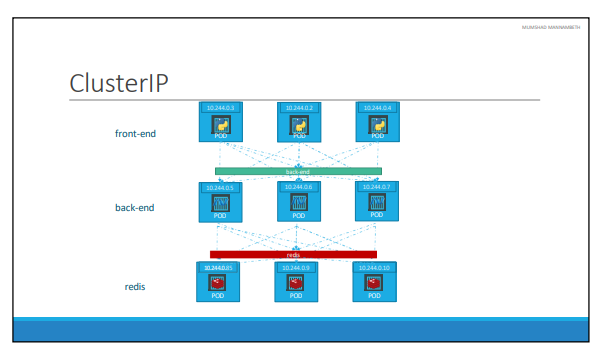
Description automatically generated

Nodeport range – 30000 to 32767

Nodeport – 30008

**IP address of PODS are dynamic in nature because pods can be deleted and created at any point of time**

**Cluster IP:**

****

[gagandeepp/example-voting-app-kubernetes (github.com)](https://github.com/gagandeepp/example-voting-app-kubernetes)

Config Maps: They are used to pass configuration data in the form of key value pairs in k8s.

1. Create Config Map
2. Inject into Pod

Secrets: Used to store the password and keys in k8s

1. Create Secret
2. Inject Secret

Objects store in secrets and etcd are just base64 encoded.

Service Accounts: Used by machines

1GB : 1000 MB

1GiB: 1024 MiB

1MiB: 1024 KiB

1KiB: 1024 bytes

Taint: To avoid placing all the pods on a node , we taint a node.

Toleration: The pod which can be scheduled on a node using k8s scheduler.

Taints are done on Node

Toleration are set on Pods.

Taint levels:

1. NoSchedule : Pods are Either nodes are not scheduled or evicted from existing nodes.
2. PreferNoSchedule
3. NoExecute

**Pods are not scheduled on Master node because Master node has No Schedule taint set on it.**

Node Selector: The mechanism by which we can schedule a pod on a particular node.

Node Affinity: We cannot provide advance expressions like or not with node selectors, to cater those we have node affinity.

Multicontainer Pods in K8s:

1. Adapter: Adapter container process logs from different pods in a common format
2. Sidecar: Deploying logging agent alongside a web server
3. Ambassador: Outsourcing proxy logic to separate container.

For example a process that pulls a code or binary from a repository that will be used by the main web application. That is a task that will be run only one time when the pod is first created. Or a process that waits for an external service or database to be up before the actual application starts. That's where **initContainers**comes in.

An **initContainer**is configured in a pod like all other containers, except that it is specified inside a initContainers section

PODS Lifecycle:

1. Pending
2. Running
3. Succeeded.
4. Failed.
5. Unknown.

PODS Conditions:

1. PodScheduled -When Pod is scheduled on a node.
2. Initialized – When Pod is initialized.
3. ContainerReady – When multiple containers of the Pod are ready
4. Ready –

Readiness Probe – When pod is up and running and ready to serve the traffic. If the readiness probe condition is met on the pod then only k8s will set the POD condition to Ready and sends the traffic to the POD otherwise it will not send the request to POD.

httpGet : For web app

tcpSocket: For DB connection

exec: To run some script

Liveness Probe: A liveness probe is used to periodically test check the application container is healthy or not. If the

Monitoring K8s Cluster: DataDog, Promethus, Metrics Server

Kubelet – Running on pod and it has component cAdvisor (Container Advisor) which send the

Performance metrics from pods and exposing them through Kubelet API to make the metrics available to metrics server.

Labels and Selectors are used to group things together.

Labels are like properties attached to k8s resources

Selectors are like filters which is used to filter the k8s resources.

Annotations: Are used to record build information, contact details etc.

Pod restartPolicy is set to Always by default.

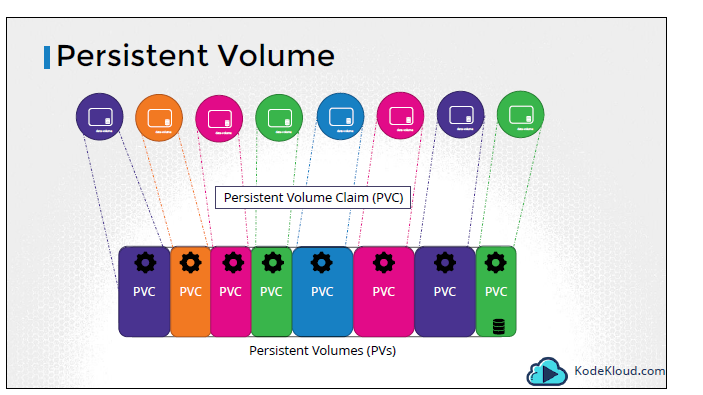
Jobs are recreated one after the another only when previous Job is success.

Volumes: To persist data of a container we attach volume to the container.

A screenshot of a computer

Description automatically generated

Persistent Volume: It is a cluster wide pool of storage volumes configured by an administrator to be used by users deploying applications on the cluster.



Persistent Volume Claims: Administrator create PV , user create PVC to use the storage.

Every PVC is bounded to single PV.

Static Provisioning: Manually provision on Cloud and then using the storage in PV.

Storage Class: We can automatically provision storage on GCP. Storage class will create PV for us in the backend.

Stateful Set: Pods are created in sequential order.They maintain sticky identity throughout their lifecycle. Ex: {pod\_name}-0

PodmanagementPolicy: Parallel to avoid sequential pod creation

Headless Service: It is a service without IP address and doesn’t do load balancing. IT create dns entries for each pod name and subdomain.

It is a normal service file with ClusterIP set to None.

Authentication and Authorization in K8s

Kube-apiserver: Controlling access to api server

Authentication: User ID and passwords/token stored in files, certificates , LDAP/Keberos, Service accounts

Define username,password and userid in CSV file and pass it to kubeapiserver with flag –basic-auth-file and restart the kubeapiserver

KubeConfig file: It will have server url, certificates and keys to connect to k8s cluster

Available in $HOME/.kube/config

It has 3 sections

1. Clusters: Dev, Prod, Testing Environment
2. Contexts: Which user account is used to access which cluster
3. Users: User accounts (Admin, Dev user etc)

API Groups: /metrics, /healthz, /version, /api, /apis, /logs

Core: /api

Named: /apis

Kubectl proxy: IT launches a proxy service on port 8001 and use credentials from kubeconfig file to access the cluster.

Authorization: RBAC, ABAC

Node Authorizer

ABAC: Attribute based Access control is defined by Policy in json format and we associate a user with set of permissions

{“kind”: “Policy”, “spec”: {“user”: “dev-user”, “namespace”: “\*”, “resource”: “pods”, “apiGroup”: “\*”}}

RBAC: Role based access control we associate a role with set of permissions.

Open Policy Agent: 3rd party tool that Helps with admission control and authorization.

Default mode is Always allow.

Resources in K8s are namespaced or cluster scoped.

Admission controllers are used to add restrictions like:

* Use docker image from internal repository
* Do not run docker container as admin
* Do not use latest tag on docker images

Examples of admission controllers:

1. AlwaysPullImages
2. DefaultStorageClass
3. EventRateLimit
4. NamespaceAutoProvision (Deprecated)- NamespaceLifecycle is new controller
5. NamespaceExists(Deprecated)- NamespaeLifecycle is new controller

Custom Resource Definitions (CRD): Used to define the custom kind value

Operator: Package Custom Resource and Custom controller as an extension so that it can be reused.