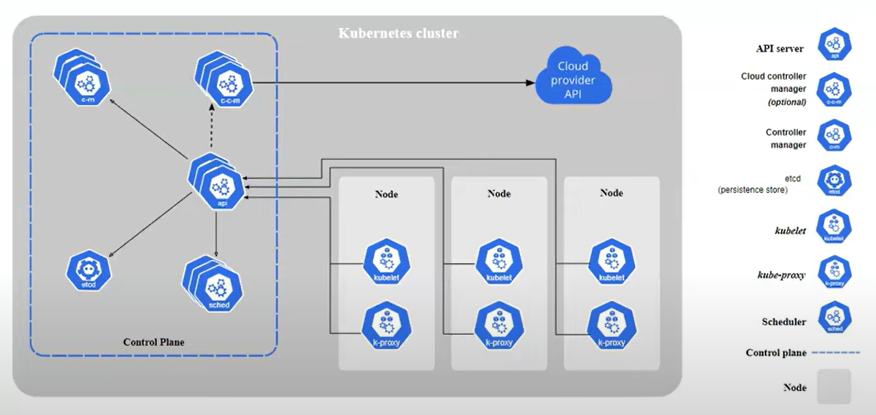
**KUBERNETES**

* Its also known as the K8’s its an open source container orchestration platform developed by google
* Kubernetes is a combination of multiple docker hosts forming a cluster
* It's designed to automate deployment, scaling,management of containerized apps across a cluster of nodes.
* Docker host = Node
* Pod is a thin layer over the container.
* Clustering = its a collection of multiple systems (Master and multiple worker nodes)
* It provides a consistent and reliable way to manage applications, regardless of whether they are running on-premises or in cloud or in a hybrid environment.
* Sidecar : any secondary container within the pod is called the sidecar or helper container
* Deployment : its an abstraction to the pods where we can create n number of pods using the replica set  
  

**worker nodes** : container runtime (its responsible for the container to run)

kubelet ( creation of pods and always makes the pods in running if not inform to API server)

kube-proxy ( (networking IPtables on linux machines) decides if the load is more it would add the replica to another pod)

**Master Node**  : control panel/API server (which takes all the requests from the external world)

Scheduler (it is used to schedule the pods on Wrk node1 or Wrk node2 based on the instructions from the API server.

ETCD ( its a key value store entire kubernetes

Controller manager ( replica set which controls )

Cloud controller Manger (

Deployment : its a yaml structure where we need to specify the number of replica sets and pods count  
Deployment>>> Replica Set (controller : which give autoscaling feature) >>> Pods

* **Master server component** : API server,Controller Manager, scheduler,etcd
* **API Server** : it's the main entry point to the cluster its the front end for the kubernetes control plane and it also handles the authentication and authorizations
* **Scheduler** : it watches for newly created pods with no assigned nodes and it selects the nodes for them to run.   
  It used to check the available node and will deploy the newly container deployment request
* **Controller Manager:** its a control plane component which runs the controller processes.
* **Etcd database:** its a key value store of consistent and high availability it stores the all cluster data.
* Worker node components : kubelet, kubelet-proxy , any container platform
* **Kublet** : its an agent which passes the info to the master node (transactions and resource utilization)
* **Kubelet-proxy** : its used to select the best path for the communication within the containers between them.

It is also used to update the IP tables

* SVC : load balancing (distributes the traffic), Discovery(labels & selector mechanism)Expose to external
* **Kubeshark**  : its a tool used to monitor how traffic is flowing within the kubernetes.
* Config Map; its used to store the data in the pod info that can be used by your application at later point of time.
* Secret : secrets always deals with the sensitive data, the data gets encrypted before the object saves in the ETCD. and also give the least privilege through the RBAC.
* Volumes : which are persistent storage and also we can mount inside the container also we can config the config map as a files so that any modification on the config file will not restart the container and will get autoupdated using the these.  
   **Workload Components**
* Pods, deployments,ingress,services,configmaps,secrets,Namespaces,persistent volume & claims, statefulsets.
* Pods : its a collection of containers
* Deployments ; will be used when they need to do the replica of the pods (its a collections of pods)
* Namespaces : it uses to partition the logical volume technology with in the clusters
* Gcloud auth login : its used to login to the google cloud the basic authentication
* Kubectl get nodes : it just shows the all the nodes with in the cluster
* Kubectl get all -A : which gives all the details of all pods
* Kubectl get pods : it just shows the all the pods with in the cluster
* Kubectl describe pod podname : it gives all the information about the pods (similar to inspect on docker)
* Kubectl get pods -0 wide : it just shows the all the pods with in the cluster along with the ip address and o/p
* Kubectl get deploy : it just shows the all the deployments within the cluster
* kubectl get replicasets : it just shows the all the replicasets within the cluster
* Kubectl get rs : it shows all the replica sets
* Kubectl get svc : it just shows the all services with in the cluster
* Kubectl get cm : it displays the available config maps
* Pod’s : its the smallest & basic unit of deployment it represent the 1 instance of a running process with in the cluster.  
  Its encapsulates the 1 or more containers storage resources & network config and other options required to run a specific set of container together.(here second container is called helper container or side car)
* If they are any discrepancies in kubernetes automatically take actions to reconcile the state creating or deleting pods if necessary its called the self healing capability.
* **Kubectl apply -f .\sample\_deploy.yml** : which runs and executes the yml file and creates the pods with the given image within the yaml file.
* kubectl delete -f .\sample\_deploy.yml : which runs and executes the yml file and creates the pods with the given image within the yaml file.
* Kubectl describe pod podname : To get the full details of the pod
* Kubectl edit deployment deploymentname : where its a command we can edit the replication count.
* Kubectl get all -n kube-system : it shows the default metrix server
* Services : its an abstraction layer on the pods (which has the static ip)which provides the consistent way to access and communicate.  
  These services can be deployed using the manifest yaml file.  
  **Types of services** : Cluster ip, Node port , Load Balancer, External Name:
* Cluster Ip : its the default service type which exposed on the service on cluster-internal IP address its accessible only with in the cluster.
* NodePort: Exposes the Service on a specific port on each node in the cluster. The Service is accessible using the cluster nodes' IP addresses. ( for testing purpose)
* LoadBalancer: Provision a cloud provider's load balancer to distribute traffic to the Service. External clients can access the Service using a publicly accessible IP address.
* ExternalName: Maps the Service to a DNS name, allowing the Service to redirect requests to an external endpoint outside the cluster.
* Ingress : its an API object that provides external access to services within a cluster.  
  Primarily used for the reverse proxy mechanism  
  Ingress controller : its responsible for fulfilling the ingress rules by config and managing the underlying LB or reverse proxy that handles the incoming traffic.  
  Ingress resource :
* Kubectl create namespace ingress-nginx : its to create the ingress controller
* Kubectl get ingress : it is used to show the ingress resources along with the LB ip
* Kubectl get svc -n ingress-nginx : it shows the complete details of the ingress controller and its resources
* **Configmap** : its an API object used to store configuration data which is consumed by the pods and other resources within the cluster  
  We can update the config settings from container app without modifying app code or container image  
  It stores the key-value pair and provides ability to mount config files as the data files
* Secrets : its an API object used to store the sensitive info like passwords,tokens & ssh keys  
  It provides a way to securly store and manage sensitive data within cluster.  
  It can be mounted as files or exposed as environment variables within a container.  
  Opaque : common type of secret storage key-value pair base64 encoded string  
  Docker-registry: it uses to store the credentials to authenticate private docker registry  
  TLS : to store the TLS certificate and private key it include the tls.crt and tls.key  
  Service Account : Automatically created secrets which provide cred for accesing kubernetes API  
   Which allows pod

**NAMESPACES**

* These are the way to create the virtual clusters within the physical cluster and they provide a way to divide cluster resource into logical group by enabling multiple teams to coexist and operate with in the kubernetes cluster.
* Each object in the kubernetes belongs to a specific namespace
* Isolation: Namespaces provide isolation between resources. Objects in one namespace are typically not aware of objects in other namespaces, unless explicitly configured to communicate.
* Resource Allocation: Resources like CPU, memory,storage, and network bandwidth can be allocated and managed at the namespace level. This allows for resource quotas and limits to be set for each namespace.
* Access Control: Kubernetes RBAC (Role-Based Access Control) can be used to define fine-grained access control policies at the namespace level. This enables different teams or users to have different permissions within their respective namespaces.
* Namespace Scopes:
* Kubernetes has four default namespaces: default,kube-system, kube-public, and kube-node-lease.The default namespace is where most objects are created by default.
* The kube-system namespace contains system-related objects created by Kubernetes itself.
* The kube-public namespace is publicly readable and contains cluster-wide resources.
* The kube-node-lease namespace is used by Kubernetes nodes to communicate their lease status.
* Kubectl get all -n kube-system : it use to show all the objects with in the kubesystem namespace.
* Kubectl create namespace namespacename : we can create our own custom namespace.
* Kubectl get namespace : which is used to show all the namespaces available within the cluster.

**Persistent volumes**

* Life cycle of pod and persistent volumes are totally isolated
* Its a piece of storage provisioned in a cluster that can be dynamically allocated and managed by admins
* It provides the way of decouple storage from lifecycle of pods and allows data to persist beyond the lifetime
* PV can be provisioned statically and dynamically static provisioning involves creation pv in advance while dynamic provisioning allows pv to be created on demand.  
  Access modes : Read write once, read only many , read write many  
  Persistent volume claims :   
  retain : the PV associated with the PVC is not automatically deleted or released when the PVC is deleted. Instead, the PV is retained and its contents are preserved.,   
  delete : defines the associated PV is automatically deleted when the pvc is deleted   
  Recycle : its a deprecated and no longer recommended for the use.  
  Kubectl get pvc : it used to show all the Persistent volume claims with in the cluster  
  Kubectl edit pv : which is used to edit the default written yaml file and we can make required modifications

**Manifest**

**Apiversion : v1**

**Kind : pods/replicaset/deployment**

**Metadata :**

**Name : Mydemoapp**

**Spec :**

**Replicas : 3**

**Selector :**

**matchlables :**

**app : demoapp**

**INGRESS(simillar to Application gateway)**

* Its used for reverse proxy machinism

L4 : it used to collect the traffic and transfer it to the backend pools (no changes will be applied)

L7 : Application gateway (intelligence) based on the Url hit by the user we can run multiple websites using the single LB.

**(routing rules)  
Users request >> ingress controller >> Ingress Resources>> service:app  
  
 Statefullsets**

* It will use some certain naming conventions while creation of pods ( it has sequential & standard naming convention and while deletion as well it uses the reverse order for the deletion LIFO It has the unique network identity for each pod
* Headless service : it automatically creates a headless service which allows each pod to have its own dns entry which enables the direct communication btw the pods and unique hostnames
* Data Replication : each pod in statefullset has its own pv and pvc data replication can be handled at storage level

**Daemonsets**

* These are type of controller used to ensure thata a copy of a specific pod is running on all nodes in cluster
* In replica sets we need to specify the replica count in daemonsets we dont need to
* These are commonly used to run system services or monitoring agents on each node by ensuring that these essential services are present and active throughout the cluster
* **Jobs :**  its a controller object that create one or more pods to perform a specific task and will marked as completed(successful)
* Horizontal pod autoscaller : if a pod reaches the threshold limit then HPA adds the extra replica of pods based on the conditioned threshold (depends on the inbuild metrix server plugin)
* cluster autoscaller : It automatically adjust the size of the kubernetes cluster by adding and removing nodes based on the resource demands of the pods
* Limit Range : we can specify the limits for the resources with in the namespace
* Resource quota : where it used to allocate few limits of ram and vcpu from the node to the namespace

**DAY to Day Activities**

* We manage kubernetes cluster for our org and we ensure that apps are deployed into the cluster and to maintain no issues on the apps so we have setup monitoring on the kubernetes cluster and we ensure that when ever they are bugs we support them ( if developer were unable to troubleshoot issue w.r.t pods or services we wil help them in solving) we also maintenance activities (3 masters with 10-12 Worker nodes) upgrading of the versions of the worker nodes and installing default mandatory packages  
    
  Along with this if any team is facing any issue with the kubernetes they used to create a ticket for us we used to work on those iteams