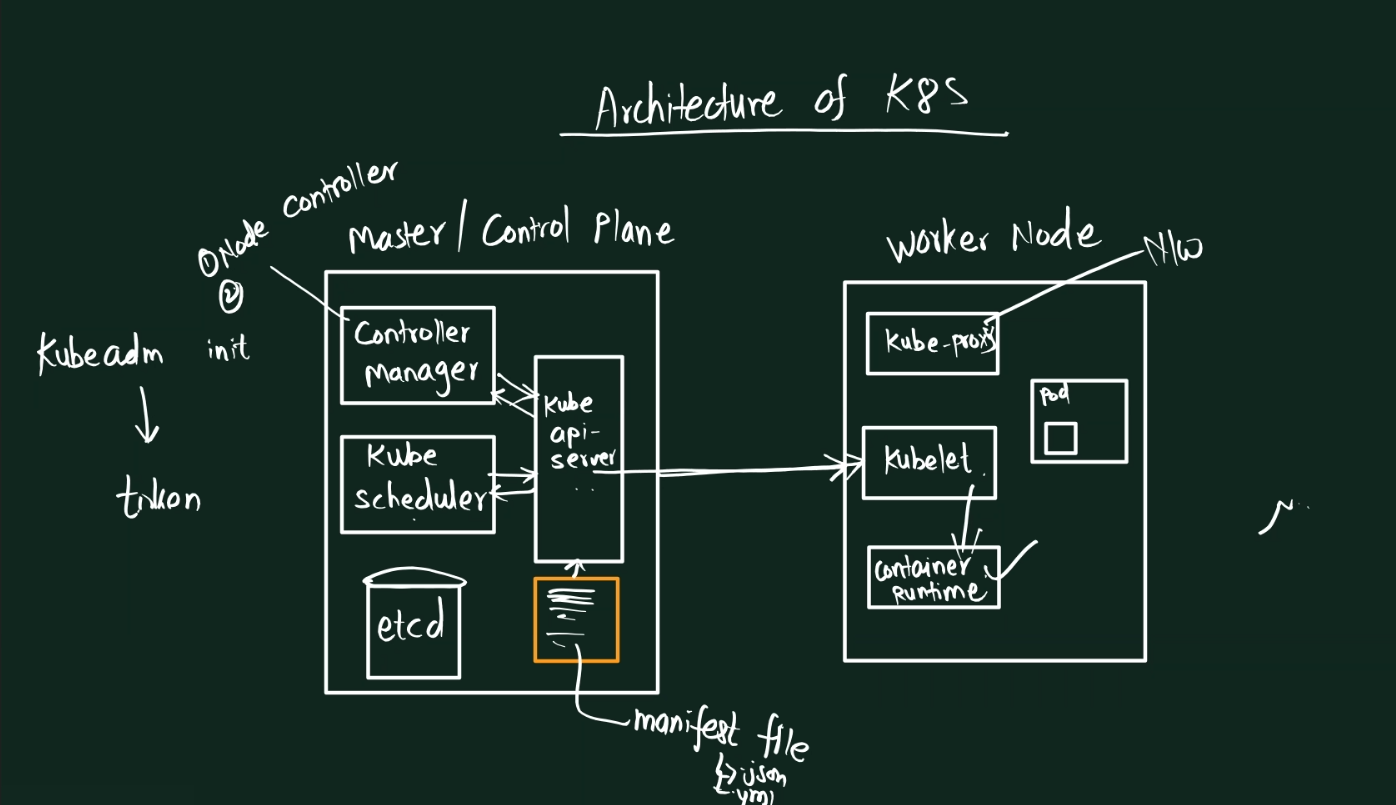
**Cheat sheet for Kubernetes**

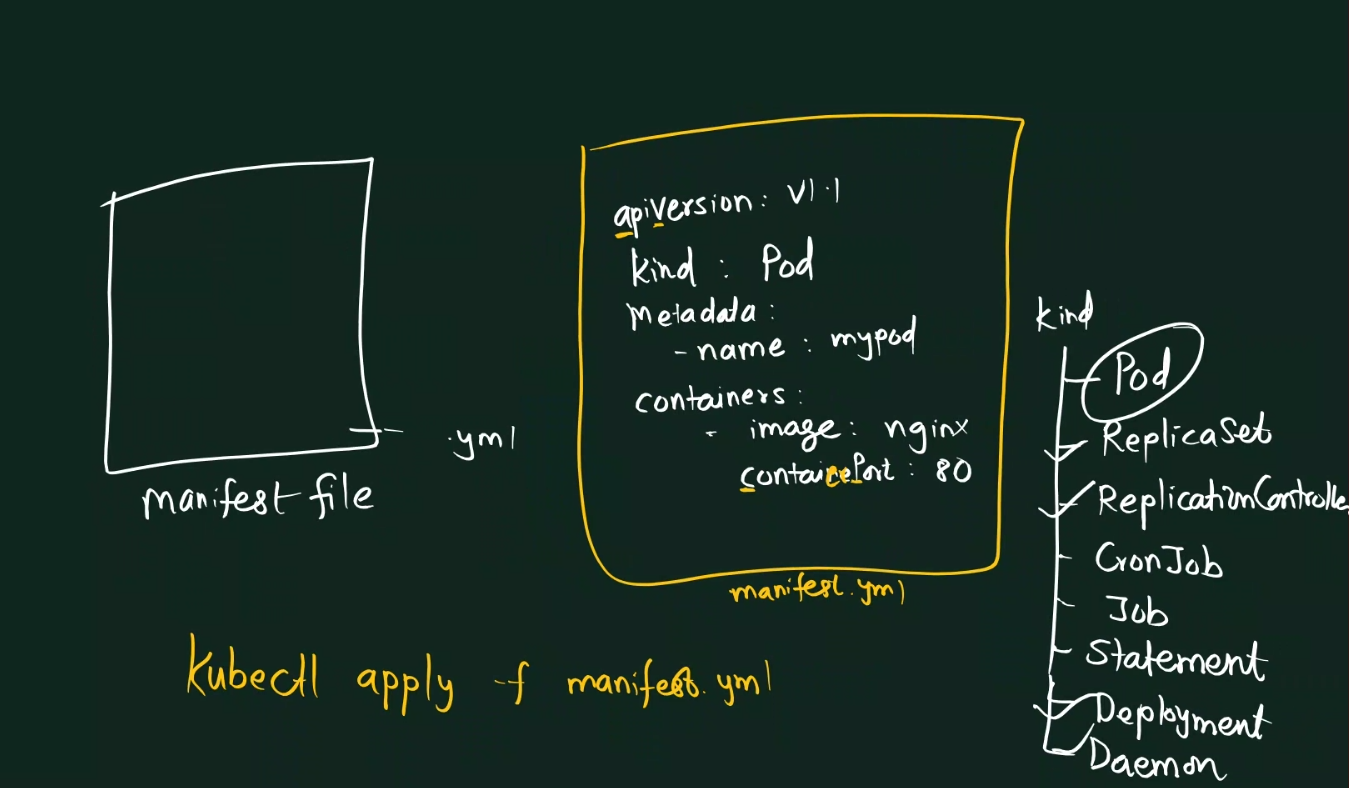
* It is a container management tool/Orchestration tool
* In K8s 8 stands for the 8 characters in between K & S
* Reasons to use K8s –

1. To restart or recreate container
2. Autoscaling of containers (vertical & horizontal)
3. Load balancing
4. Networking among container on different host
5. Fault tolerance
6. Create containers on different host
7. Deployment of different version
8. Rollback to previous version
9. Health monitoring of containers
10. To manage the containers

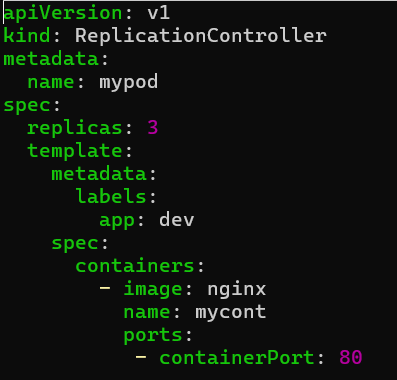
* It schedules, runs & manages isolated containers which are running on virtual/private/cloud machines
* In k8s there are minimum two nodes master node/control-panel and worker node
* Because of K8s the system will be highly available
* All top cloud providers support K8S
* There is different cloud based K8S services GKE (google), AKS (Azure), EKS(Amazon)
* Kubernetes is written in golang language
* Cluster 🡪node🡪 pod 🡪container 🡪application/microservice
* ARCHITECTURE OF K8S



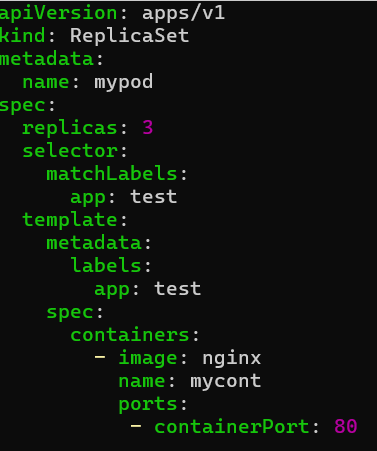
* Api server is medium through which all the communication will be done
* Controller manager looks up to the actual state is equal to desired state
* etcd is external entity but it has all the data about cluster, it can only be accessed by Api server
* kube scheduler does all the work according to Controller manager & Api server which schedules & do the tasks
* user will communicate with Api server with the help of manifest file which will be written in YML/JSON
* pod is a logical unit, if pod is failed it can’t be recreated, multiple containers can be created inside single pod
* kubelet manages pod (how many containers should be in pods)
* kube proxy does the networking in node, it assigns the IP
* Container runtime/container engine is where we use docker
* We can create multiple master for high availability
* MANIFEST FILE – To create pod with running container, apply this to cluster (to master) to bring into desired state



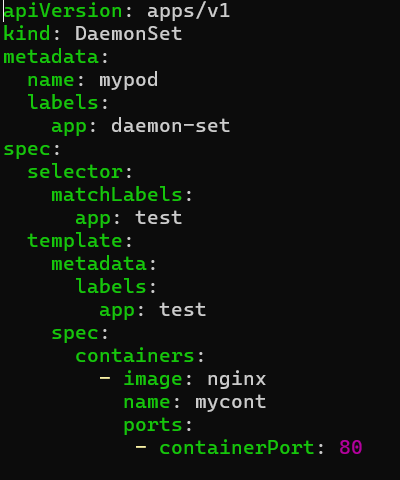
* Namespaces – It is logical separation of pods, group of pods with common namespace
* Replication Controller – To create multiple replicas of pod, if we delete one of pods it will create the same pods again. If a pod created using RC, it will be automatically replaced if they crash, failed or terminated



* Replica Set – It is a next generation of RC, Replica set is to create multiple replicas of pod, but we can search with the labels in this replica set



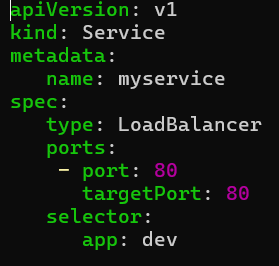
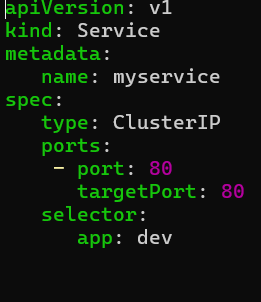
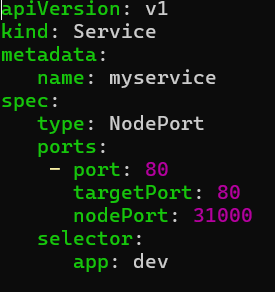
* Daemon Set – To create at least one replicas of pod in different nodes



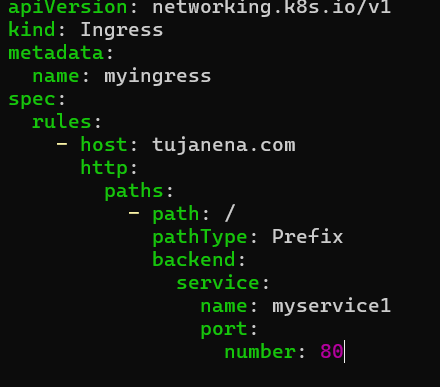
* Health check Probes –

1. Liveness Probe –
   1. http/https Get
   2. TCP socket connection
   3. Command execution
2. Readiness Probe - http/https Get
3. Startup Probe - http/https Get

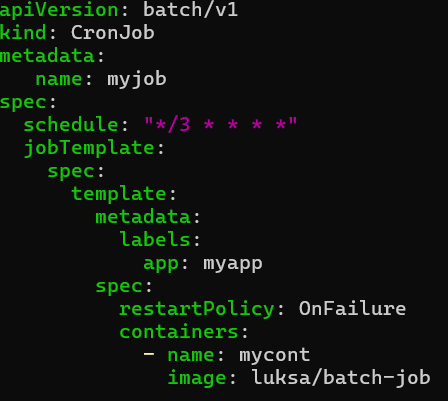
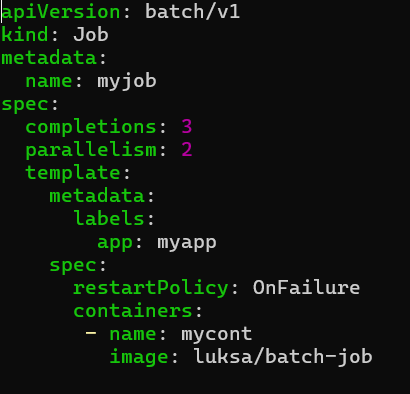
* Services –
  1. Cluster IP – In this service we can only do the internal communication
  2. Node Port – In this service we can do the external communication with the help of given node port
  3. Load Balancer – In this service we can do the external communication but we have to create ELB and map it to the load balancer service to do the load balancing

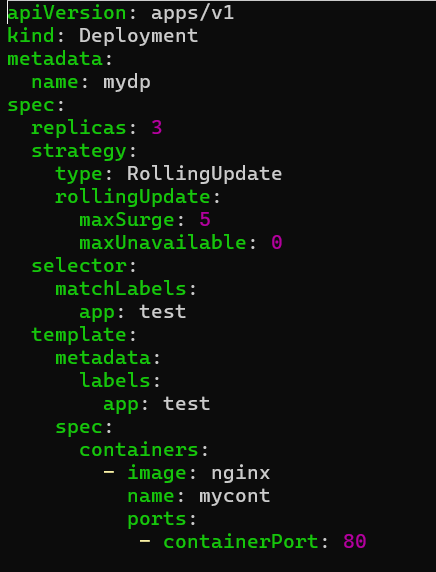
* Ingres – It provides a way to route external requests to different services inside the cluster based on rules, such as hostnames or paths.



* Jobs – Jobs are typically used for tasks that need to be run once or a finite number of times, such as data processing, batch tasks, or one-off operations.
  1. Single Run Job: Runs a task once and ensures it completes.
  2. Parallel Job: Runs multiple Pods in parallel.
  3. CronJob: A CronJob is a special type of Job that runs on a schedule.



* Pod (v1) 🡪replication controller (v1) 🡪replica set (app/v1) 🡪deployment (app/v1)
* DEPLOYMENT – A deployment object act as a supervisor for pods, giving you fine grained control over how and when a new pod is rolled out, updated or rolled back to a previous state. This provides a self-healing mechanism to address machine failure or maintenance.



* Stateful set – To maintain the same name of pod after recreation of pod (we just have to change the kind in replicasets yml file.)
* VOLUMES –
  + Empty directory – To share the data among multiple containers within a single pod.
  + Git repo – To use git-hub public repository as a volume.
  + Host path – To use directory on worker node as a volume.
  + NFS (network file system) – To create a EFS and map it to the multiple nodes/instances to share the data between two nodes.
  + Persistent volume – It is a piece of storage in the Kubernetes cluster that has been provisioned either statically or dynamically. A Persistent Volume Claim is a request for storage by a user or an application. It allows applications to claim a PV that matches the specified storage size, access modes, and other requirements.
* Config Map – A ConfigMap in Kubernetes is an object that lets you store configuration data as key-value pairs (without encoding the keys)
* Secret – A Secret in Kubernetes is an object that stores sensitive data, such as passwords, API keys.
  + There are three types of secret – Generic, Docker registry, TLS
* HPA (Horizontal pod Autoscaling) – It is a feature that automatically scales the number of pods in a deployment, replication controller, or stateful set based on observed CPU, memory, or custom metrics.
  + Metrics is used to monitor in K8s

**Commands**

* Kubectl get node – to get all nodes
* sudo systemctl restart kubelet – To restart the all nodes
* kubectl run mypod --image=image-name – to pull image in pod
* kubectl get pods/pod/po – to show pods
* kubectl get pod -A – To show all pods
* kubectl delete pod pod-name – To delete specific pod
* kubectl delete pod –all – To delete all pods
* kubectl get pod -o wide – to get detailed info about pods
* kubectl apply -f manifest.yml – To run the created manifest file
* kubectl exec -it pod-name -- bash – To get inside of pod
* kubectl label pod pod-name env=label-name --overwrite – To give labels to pod after creation
* kubectl get pods --show-labels – To show labels on pods
* kubectl get pods -l label-name – To get only env label pods
* kubectl get pods -l ‘!label-name’ – To not get only env label
* kubectl label node pri-ip of node label-name – To label the nodes
* kubectl get ns – To show all namespaces
* kubectl create ns ns-name – To create namespaces
* kubectl get pods -n ns-name – To show pods specific namespaces
* kubectl apply -f manifest.yml -n ns-name – To assign ns to pods after creation
* kubectl config set-context --current --namespace=default – To set default pods
* kubectl delete ns facebook – To delete ns
* kubectl delete all --all -n facebook --force – To delete all the pods inside of namespace
* kubectl scale rc pod-name –replicas=20 – To modify the number of replicas through command
* kubectl get rc – To show replicationcontrolled pods
* kubectl get svc/services – To show services
* kubectl get ingress – To show ingress
* kubectl delete ingress ingress-name – To delete ingress
* kubectl get job – To show ingress
* kubectl delete job myjob – To delete job
* kubectl get all – To show all the attributes
* kubectl get rs – to show replica set
* kubectl get deploy – to show deployment
* kubectl scale –replace=1 deploy deploy-name – To scale up or scale down by command
* kubectl logs -f <pod-name> - to check what is running inside containers
* kubectl set image deploy DP-name mycont=httpd – to change the image in deployment
* kubectl rollout status deployment DP-name – to check the status of deployment
* kubectl rollout history deployment DP-name – to check the history of deployment
* kubectl rollout undo deployment DP-name – to undo the rollout of deployment
* kubectl create configmap myconfig -MYSQL\_ROOT\_PASSWORD=pass@123 – To create config map
* kubectl create secret generic mysecret --from-literal MYSQL\_ROOT\_PASSWORD=sumit2606 – To create secret
* kubectl get object object-name – to show the object
* kubectl describe object object-name – to describe the object
* kubectl delete object object-name – to delete the object