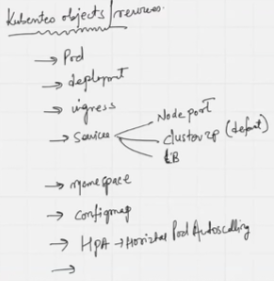
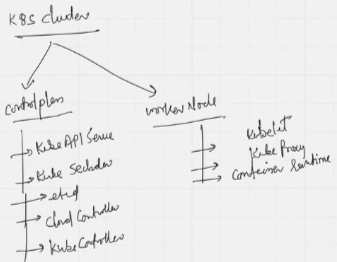
Summary in Kubernetes

>> To connect to cluster, user need Kubectl which is a CLI utility to connect with cluster

| Feature | Managed Kubernetes | Self-Managed Kubernetes |
| --- | --- | --- |
| Control | Less direct control, provider handles the control plane | Full control over all aspects of the cluster |
| Responsibility | Provider manages most infrastructure aspects | User manages the entire infrastructure |
| Ease of Use | Simplified setup, automated management | More complex setup, requires manual configuration |
| Cost | Potentially higher direct cost, lower operational overhead | Potentially lower initial cost, higher operational effort |
| Scalability | Often includes auto-scaling features | Requires manual scaling configurations |
| Monitoring | Often includes built-in monitoring tools | User responsible for setting up monitoring |
| Security | Provider handles security aspects | User responsible for implementing security measures |
| Expertise | Suitable for users with less Kubernetes expertise | Requires in-depth Kubernetes knowledge |
| Customization | Limited customization options | High degree of customization available |
| Availability | High availability often guaranteed by the provider | User responsible for ensuring high availability |
| Deployment Time | Faster deployment times | Longer deployment times due to manual configuration |

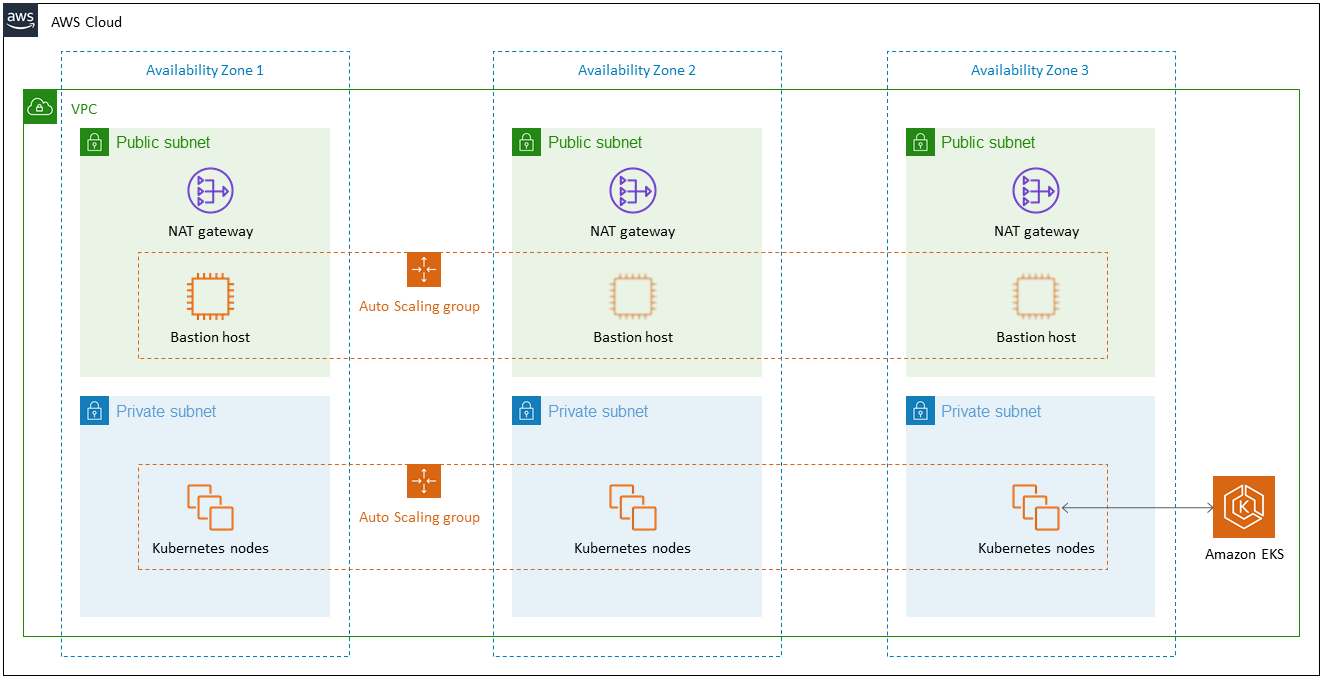
To create K8S

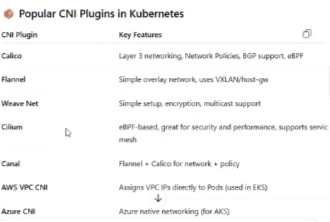
* Managed
* AWS EKS - GUI
* AWS EKS - Terraform
* Non-managed
* KOPS ……used when AWS EKS not existed before 2016 that made it managed cluster when EKS came in

To migrate the K8S cluster, Velero( <https://velero.io> ) is used to take backup on 1 cluster and take it to the new cluster. It is an open source tool.



<https://www.devopsschool.com/> ……well said K8S cluster image

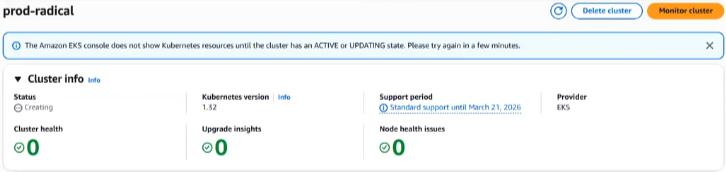




Steps to work with K8S

To Create a K8S cluster >> follow >> control-plane creation >> worker-node creation >> connect to cluster. Then POD creation >> Deploy deployment ……handle with care, Fragile

Goto AWS >> EKS >> Create cluster >> Custom configuration >> Disable EKS Auto Mode - new >> Name=mh-cluster-prod >> IAM Role >> Create Recommended Role >> point A >> Refresh & select >> Version=1.32 >> Cluster access >> EKS API and ConfigMap >> Next >> Networking >> Default VPC >> Subnets=1a,1b,1c >> Cluster endpoint access=public >> Next >> Skip Observability >> Select Add-Ons-Default >> Review & Create >> Create ……Control-plane creation initiated



Managing the Compute tab in the EKS K8S cluster is crucial. Let’s understand it here:

In Amazon Elastic Kubernetes Service (EKS), managed node groups, self-managed node groups, and AWS Fargate offer distinct approaches to managing worker nodes for your Kubernetes cluster. Managed node groups provide a balance of control and ease of management, where AWS handles node provisioning and lifecycle management, while you define the instance type. Self-managed node groups give you full control over the underlying infrastructure, including the choice of operating system and AMI, but require you to manage node scaling and updates. AWS Fargate, on the other hand, is a serverless compute engine that eliminates the need to manage any infrastructure, allowing you to focus solely on your application code.

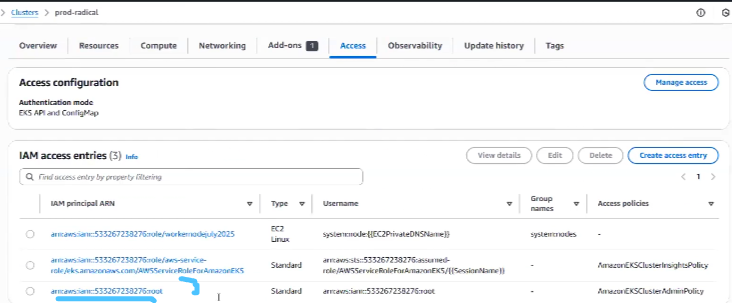
1. AWS service >> Use case=EKS >> Specific service=EKS cluster >> Next >> Role name=mh-cluster-controlplane-prod >> Add Role
2. Note on Add-Ons >> when installed individually, need to upgrade them. When installed with K8S cluster add-ons in EKS, get auto-upgrade with cluster upgrade

Goto K8S EKS >> Check whether creation completed >> Add Node Group >> Node group configuration >> Name=”workernode-airindia” >> Create Recommended Role >> point C >> Refresh & select >> Next >> Compute and Scaling configuration >> Instance types=t3.medium >> Node group scaling configuration >> Desired=2, Minimum=2(1 each subnet), Maximum=5 >> Node Group configuration=Value=1 Node >> Next >> Specify networking >> Next >> Review and Create >> Create ……Worker-node creation initiated



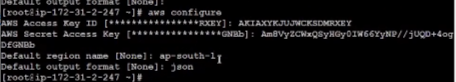
To connect to a worker-node, EC2(acts as a jump host) creation mandatory >> Create EC2=”k8s-jump-host” >> t2.micro >> Allow SSH

Make use of Satyam88/connecttok8s >> take git clone for using commands



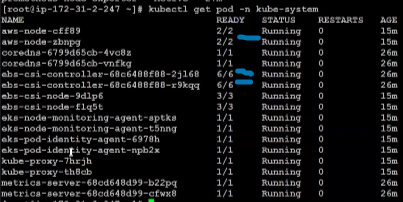
1. AWS service >> Use case=EC2 >> Specific service=EC2 >> Permissions >> AmazonEKS\_CNI\_Policy, AmazonEC2ContainerRegistryReadOnly, AmazonEKSWorkerNodePolicy >> Next >> Role name=workernodejuly2025 >> Create Role

Create an access key and secret access key for Root user although not recommended



Reach to after adding the kube config and running below: kubectl get node -o wide 

* Kubectl get node
* Kubectl get node -o wide
* Kubectl get ns
* Kubectl get namespace ……Namespace is virtual isolation
* Get pod -n kube-system

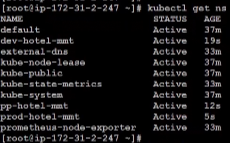
 ……zbnpg has 2 container in POD and 2 are healthy

When you see READY as ½ or anything not maximum in the READY column, it requires it to be said that the entire POD is unhealthy. I.e. 5 out of 6 denote that only 5 containers are healthy, but the entire POD is unhealthy as 1 out of 6 is unhealthy. At industry level, kube-system POD is not visible to terminal user, as highly-critical

* Kubectl create ns dev-hotel-mmt; Kubectl create ns pp-hotel-mmt; kubectl create ns prod-hotel-mmt;



* Kubectl get ns



* Kubectl run myhotel –image=nginx –namespace=dev-hotel-mmt; Kubectl run myhotel –image=nginx –namespace=pp-hotel-mmt; Kubectl run myhotel –image=nginx –namespace=prod-hotel-mmt;



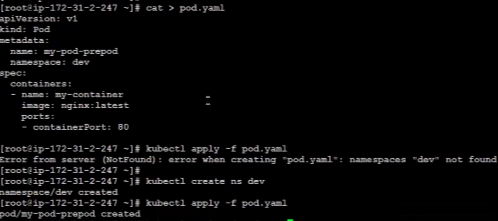
* Kubectl get pod -n dev-hotel-mmt; Kubectl get pod -n pp-hotel-mmt; Kubectl get pod -n prod-hotel-mmt;



* Kubectl delete pod myhotel -n prod-hotel-mmt



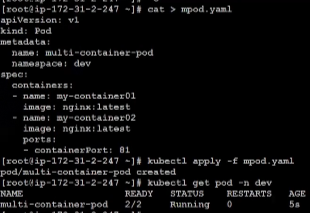
* Cat > pod.yaml
* Kubectl apply -f pod.yaml ……displays error of ns
* Kubectl create ns dev
* Kubctl apply -f pod.yaml ……kubectl uses YAML but K8S takes it as input as a JSON. YAML language is case sensitive



* Kubectl delete -f pod.yaml

Working with Multi-container POD:

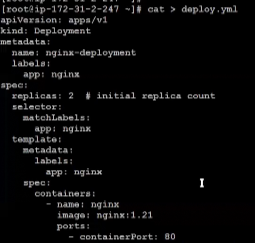
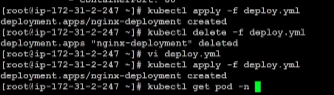
* Cat > mpod.yaml
* Kubectl apply -f mpod.yaml
* Kubectl get pod -n dev



* Kubectl logs multi-container-pod -n dev ……unhealthy POD logs

Working with Deployment:

* Cat > deploy.yaml ……adding a yaml file without ns is a big no-no in industry, so add line (namespace: dev) after (name: nginx-deployment) and it’s mandatory

* Kubectl get node
* Kubectl get pod -n dev -o wide ……update pods from 2 to 4
* Kubectl get deploy -n dev
* Kubectl delete pod -n dev PODNAME ……re-updates from 3 to 4
* Kubectl get replicaset -n dev

>> Replicaset shows monitor on deployment which tells us that deployment has POD, POD has containers, containers can be one or more.

* Vi deploy.yaml >> Update name: airindia-deployment
* Kubectl apply -f deploy.yaml
* Kubectl get replicaset -n dev ……displays 2 deployment with 4 POD

Goto AWS >> EKS >> Delete Cluster and worker-node

Auto-scale in K8S

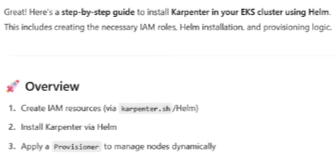
For pod-level scaling Cluster auto-scaling, Horizontal auto-scaling, Vertical auto-scaling. For node-level scaling use **Karpenter which is node auto-scaler.**

Karpeter auto-scaler is a tool that creates a new instance depending on the requirement whereas auto-scaler is fixed instance type

Can we give load on K8S cluster so that I can witness autoscaling of node in chatgpt?



I want to install Karpenter in my cluster via helm



List of CSI plugins in chatgpt