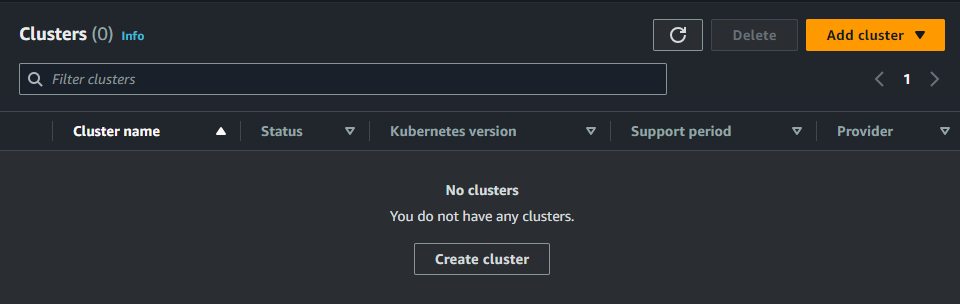
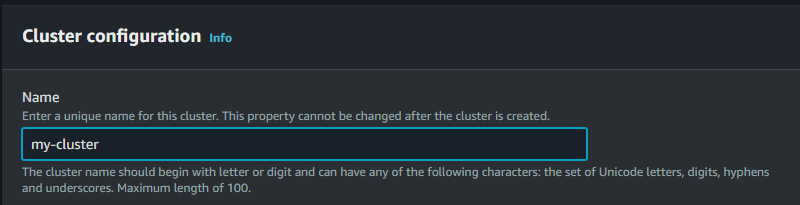
**EKS – Elastic Kubernetes Service**

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed Kubernetes service that makes it easy for you to run Kubernetes on AWS and on-premises. Amazon EKS automatically manages the availability and scalability of the Kubernetes control plane nodes responsible for scheduling containers, managing application availability, storing cluster data, and other key tasks.

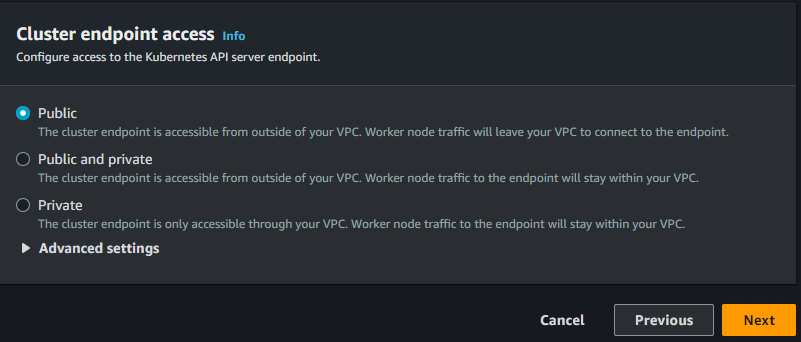
* Steps to Create Cluster.
* Search for EKS.
* Click on Clusters.
* Click Create Cluster.



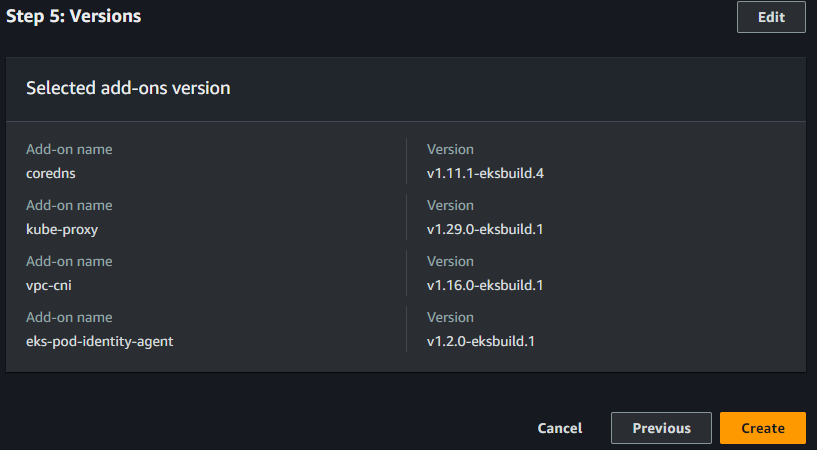
* Enter the name for the Cluster.



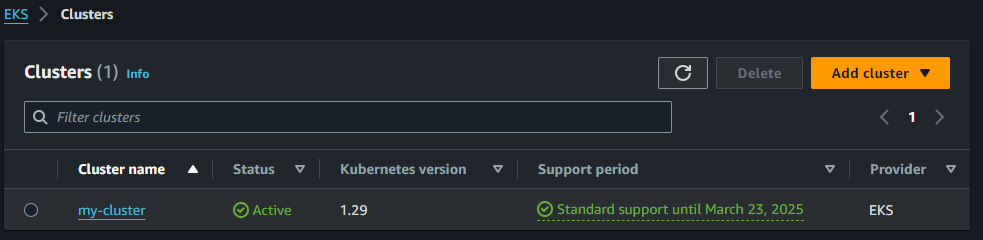
* Now we need a role to attach to the Cluster.
* Create a role for the EKS.
* Select the Role from the dropdown.
* Click Next.
* Add the Subnets and Security group.
* Set the Cluster endpoint access to public.
* Click Next.



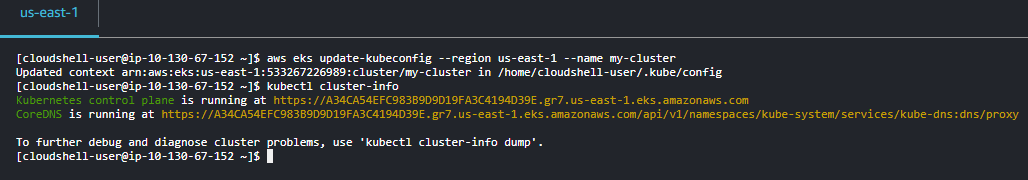
* Complete the remaining steps and Click Create.



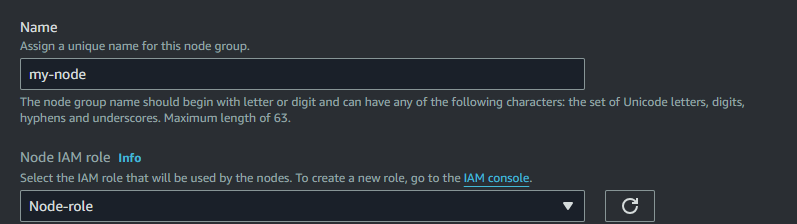
* Cluster Created.



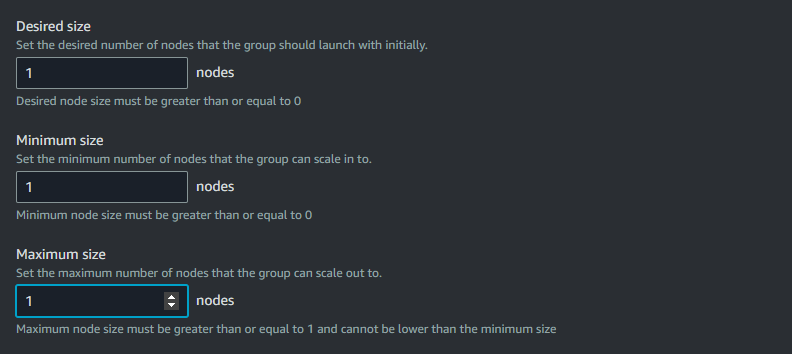
* Now Open the Cloud Shell to configure the region and Cluster.
* Hit command “aws eks update-kubeconfig --region us-east-1 --name cluster\_name” to configure the cluster.
* Hit command “kubectl cluster-info” to check the information about the cluster.



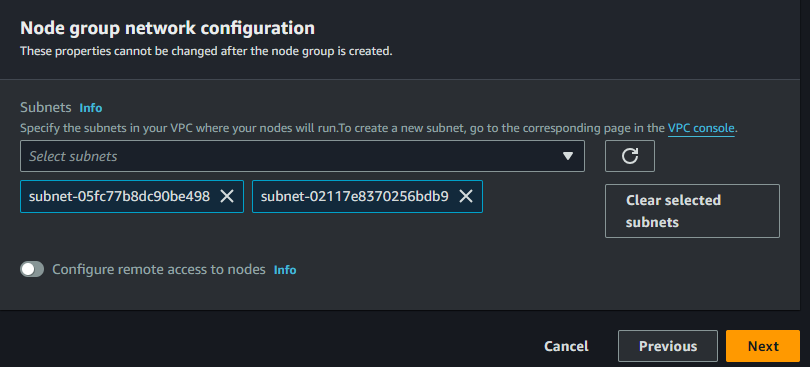
* Now we need to create node group.
* Go back to EKS and Click on Cluster-name.
* Click on Compute.
* Click Add node group.
* Enter the name for the Node group.



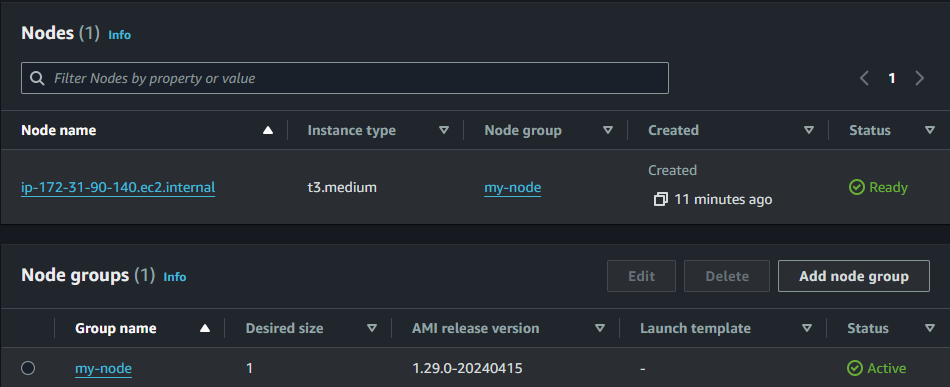
* Click Next.
* Now enter the desired size, maximum size and minimum size of nodes.
* Click Next.



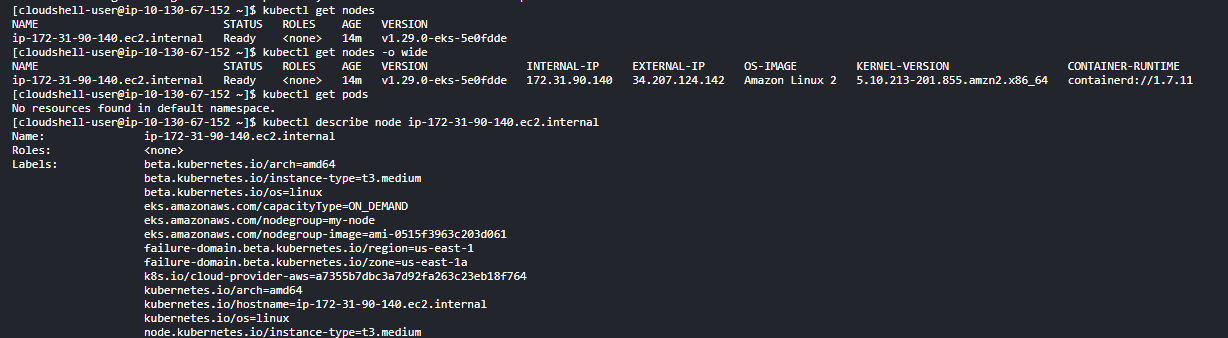
* Select subnets and click Next.



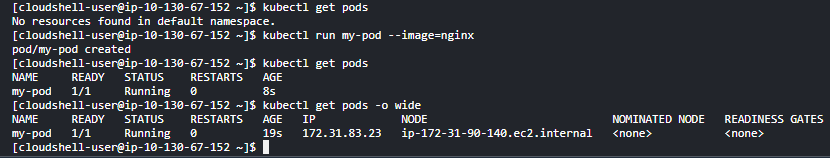
* Click Create.
* Node group and Node created.



* Hit command “kubectl get nodes” to see all the available nodes.
* We can also use “ -o wide “ for more information.
* If we want detailed information then we can use describe command.



* Now we need a Pod.
* Hit command “kubectl run my-pod --image=nginx” to create a pod and container from the nginx image.
* Hit command “kubectl get pods” to view all the pods.



* Exposing the pod port.
* Create a pod.
* Not hit command “kubectl expose pod pod\_name --port=80 --target-port=80 --type=NodePort”
* This command will expose the port of the pod externally on random port and internally on port 80.
* The --port is the listener and the --target-port is the port to which the traffic will be forwarded.
* Hit command “kubectl get services”
* Now allow the port in the security group.
* Hit the IP of the instance with the port number.
* You will be able to see the nginx welcome page.

**1. ClusterIP:**

* A ClusterIP service exposes the service on a cluster-internal IP.
* This type of service is only accessible from within the Kubernetes cluster.
* It assigns a virtual IP address to the service, which routes traffic to one or more pods that are selected by a label selector.
* It is the default type of service created when you use `kubectl expose`.
* Typically used for internal communication between different parts of your application.

**2. NodePort:**

* A NodePort service exposes the service on each node's IP at a static port.
* This means the service is accessible from outside the Kubernetes cluster by connecting to any node's IP address and the specified static port.
* It allocates a specific port on every node in the cluster, and any traffic sent to this port is forwarded to the service.
* It provides a way to expose a service externally without the need for a load balancer.
* Often used for accessing services from outside the cluster, such as exposing web applications to the internet.

**3. LoadBalancer :**

* A LoadBalancer service type in Kubernetes provides external access to the services running inside the cluster by allocating an external IP address and automatically setting up a load balancer in the cloud provider's infrastructure (such as AWS, GCP, Azure, etc.). Here's how it works:

1. **External IP:** When you create a LoadBalancer service, Kubernetes interacts with the cloud provider's API to provision a load balancer and obtain an external IP address.
2. **Traffic Distribution:** This external IP address is then used to distribute incoming traffic across the pods associated with the service. The load balancer intelligently routes traffic to healthy pods based on the configured load balancing algorithm (typically round-robin or least connections).
3. **Automatic Scaling:** The cloud provider's load balancer automatically scales to handle varying levels of traffic, ensuring your application remains available and responsive.
4. **Integration with Kubernetes Services**: From the perspective of Kubernetes, the LoadBalancer service is similar to a ClusterIP service. It creates a ClusterIP service to route traffic internally within the cluster, and then the cloud provider's load balancer routes traffic from outside the cluster to the ClusterIP service.
5. **Exposing Services Externally:** LoadBalancer services are commonly used when you need to expose your services to the internet or external clients securely. For example, you might use a LoadBalancer service to expose a web application to users on the internet.

* All Kubernetes commands.
* # aws eks update-kubeconfig --region region-code --name cluster\_name.
* # kubectl get nodes.
* # kubectl get pods.
* # kubectl describe node node\_name.
* # kubectl config view.
* # kubectl get services.
* #kubectl run pod\_name --image=image\_name.
* # kubectl version.
* # kubectl logs pod\_name.
* # kubectl logs --since=1h pod\_name.
* # kubectl delete pod pod\_name.
* # kubectl describe pod pod\_name.
* # kubectl expose pod pod\_name --port=80 --targer-group=80 --type=NodePort.