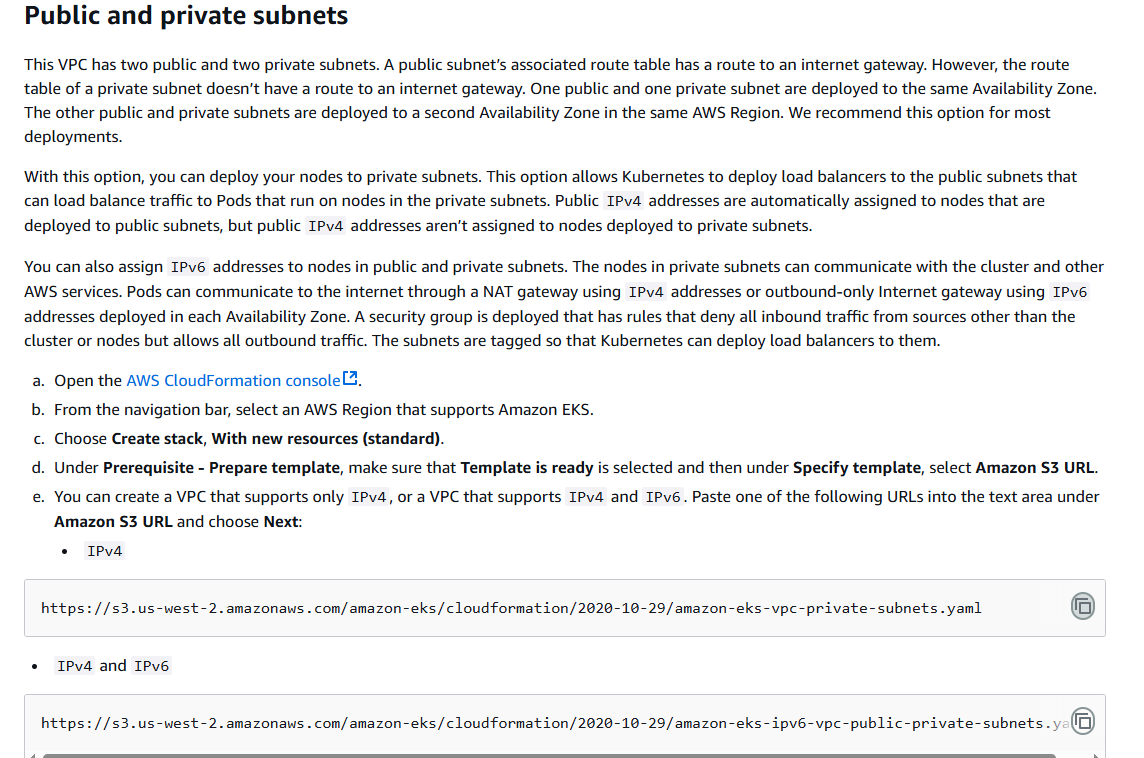


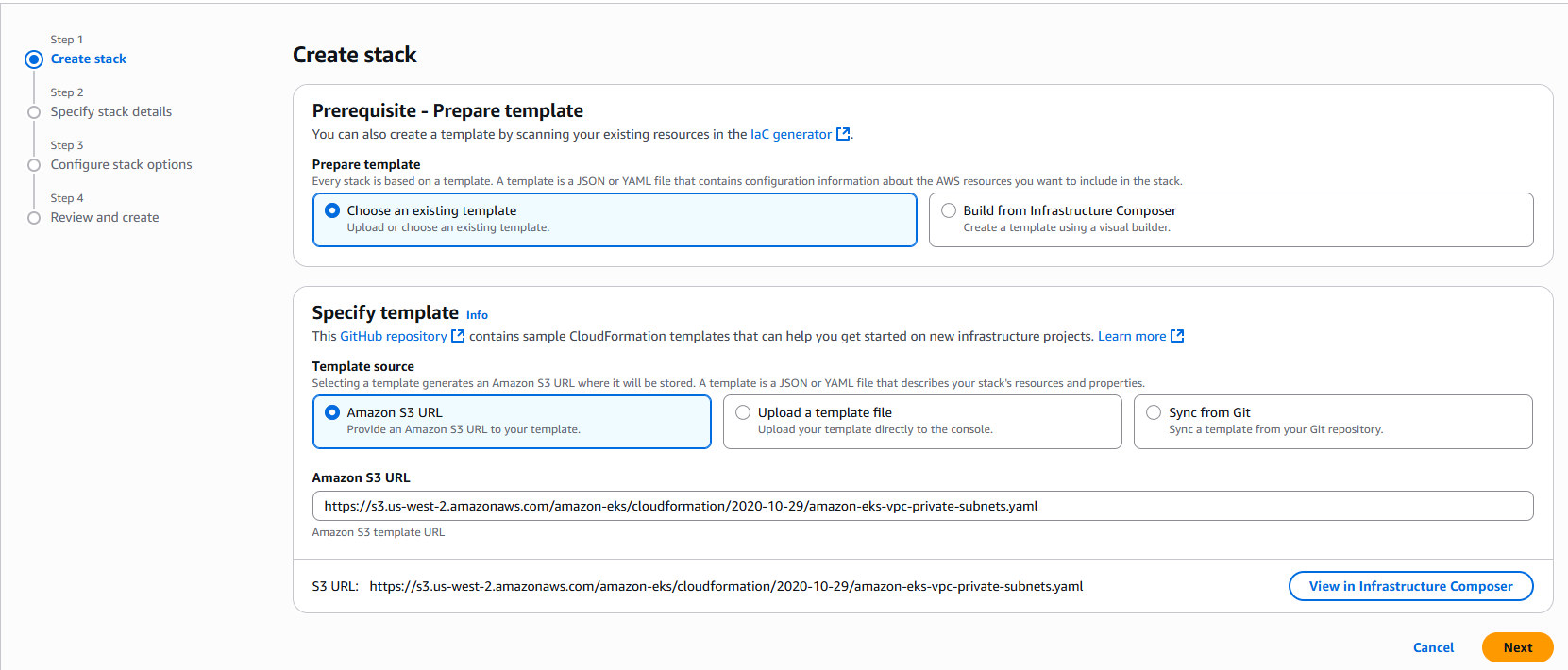
2**. CREATE A VPC**

We have to create a separate VPC for out EKS worker nodes since they require special firewall rules in order to run based on best-practice. Configure PUBLIC subnet and PRIVATE subnet.

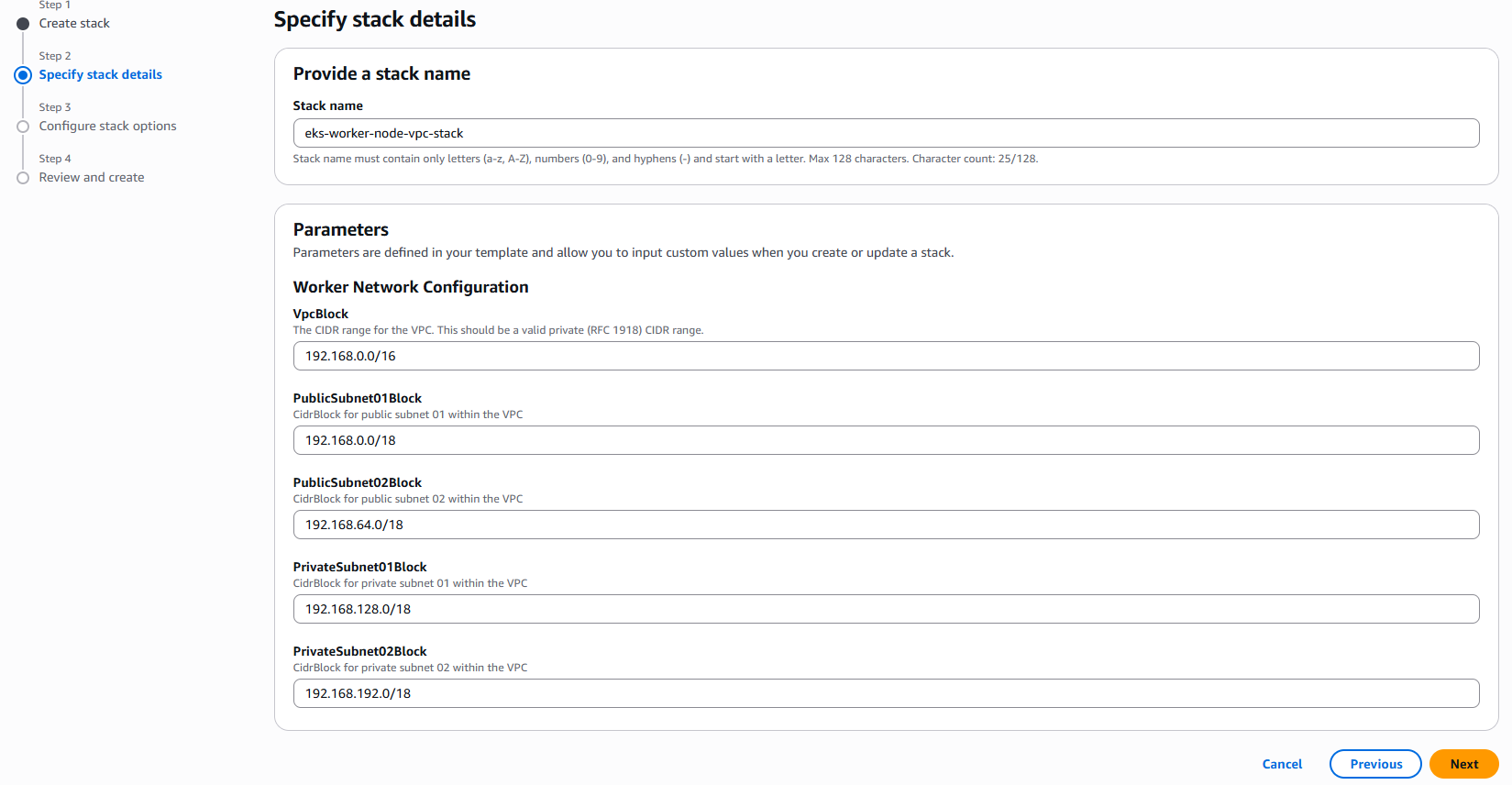
We are going to use CLOUD FORMATION template so as to ease the building of all these configurations.

Copy the URL from AWS documentation and paste on cloud formation AWS console

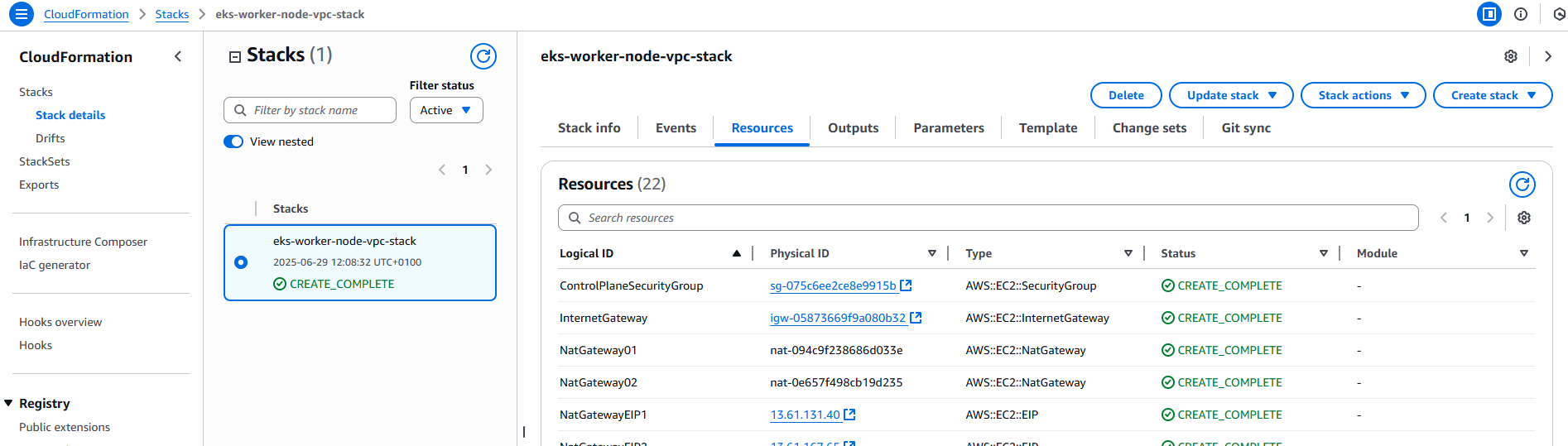




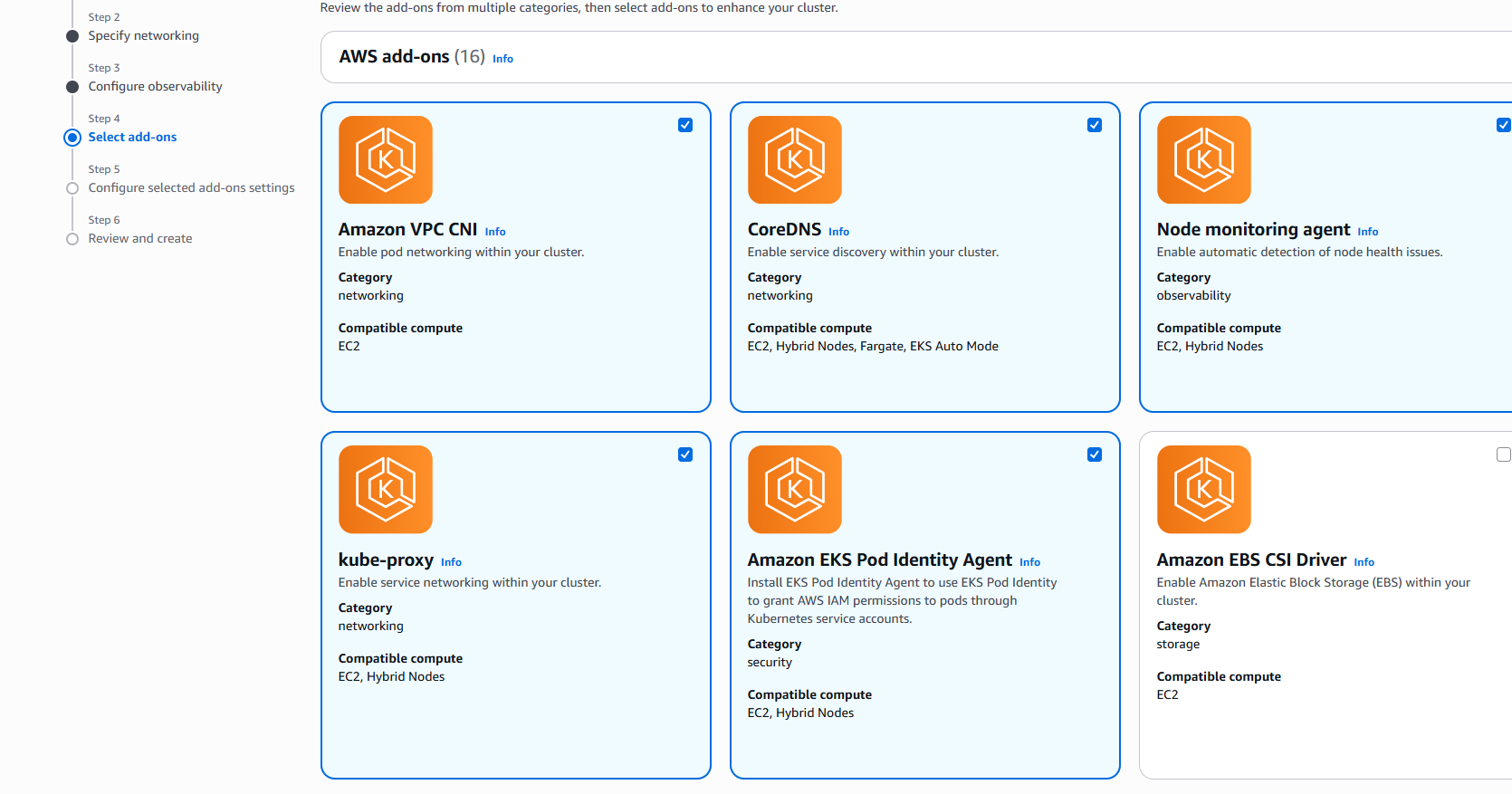
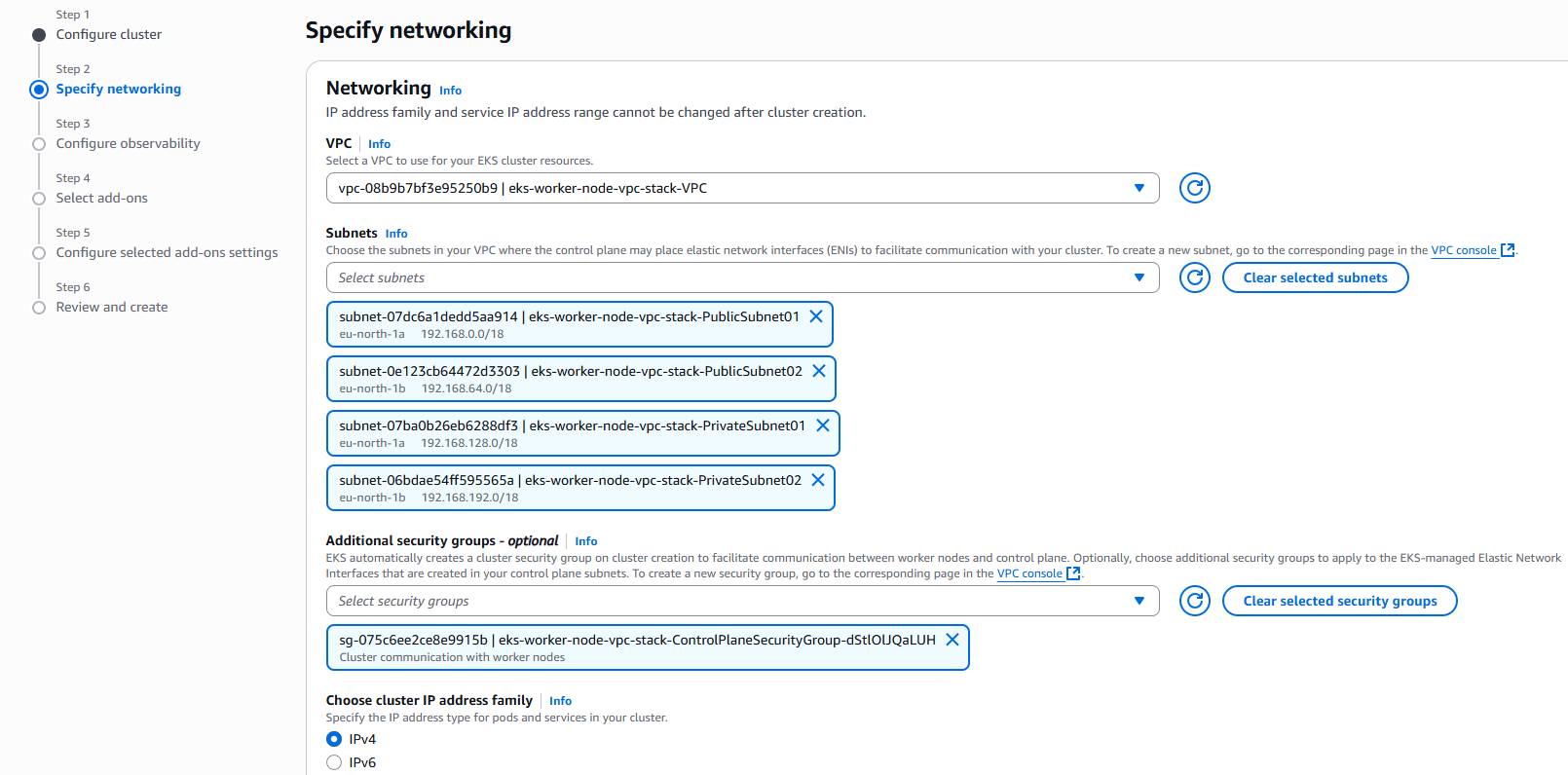
Click “NEXT”



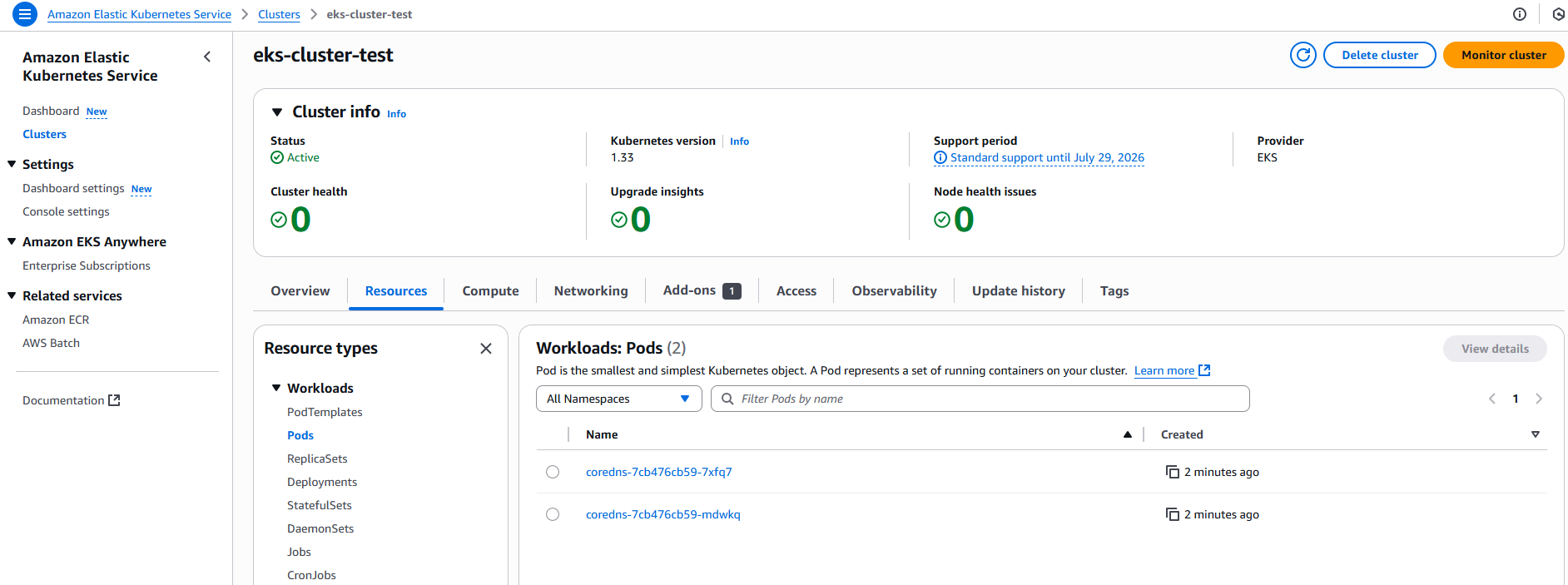
22 RESOURCES WERE SUCCESSFULLY CREATED AS SHOWN BELOW



**3. CREATE EKS CLUSTER**



The Cluster has been successfully created



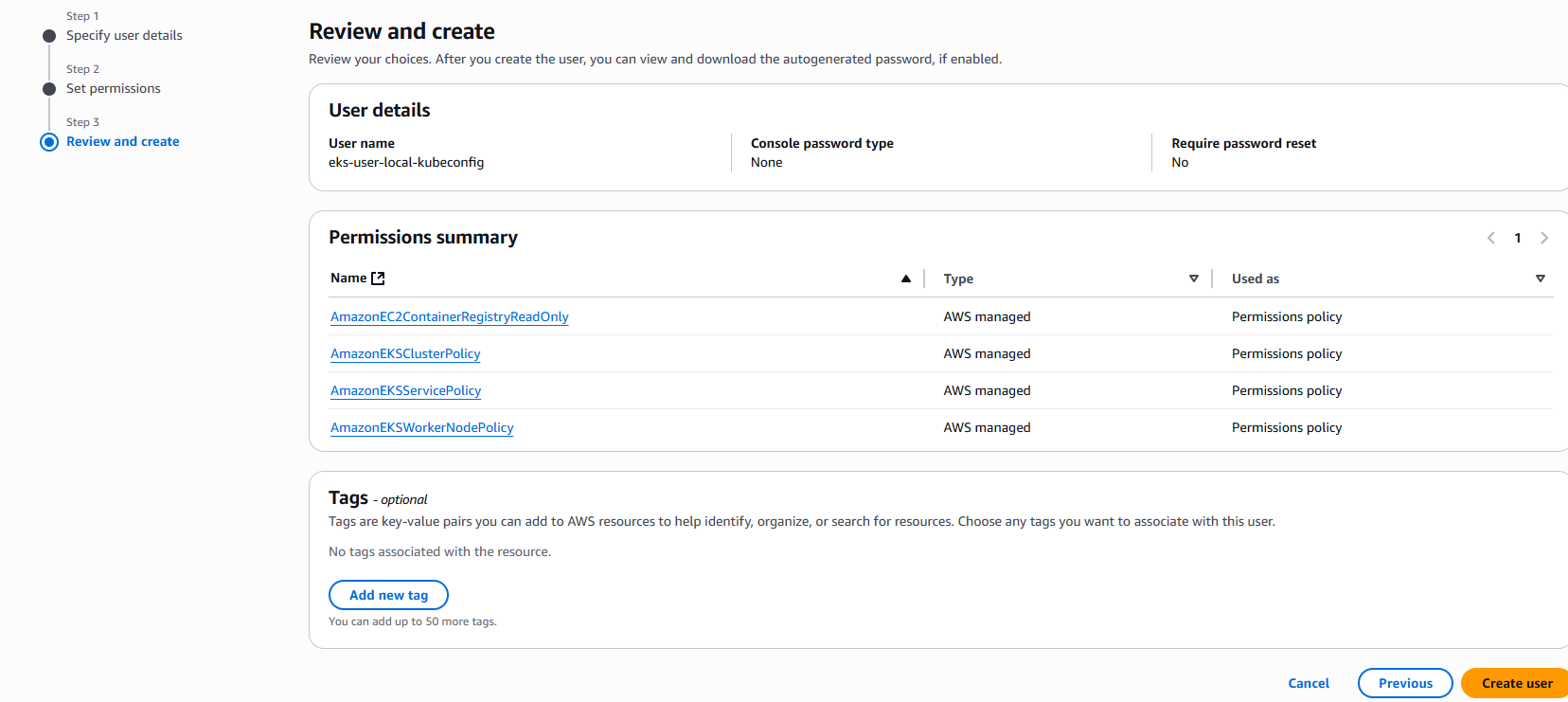
**4. CONNECT KUBECTL WITH EKS CLUSTER**

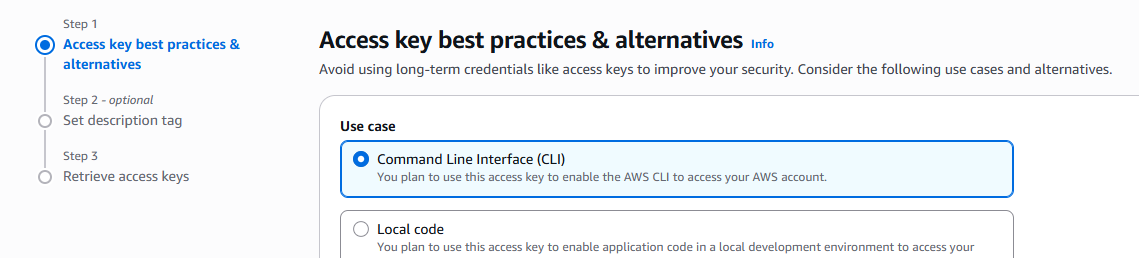
This can be done by creating a Kubeconfig file for the newly created cluster.

Ensure you have AWS CLI installed if now, download from the official documentation and install based on your OS.

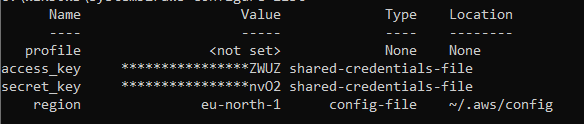
For windows, run **aws --version** to ensure successful installation

Then connect your AWS ACCOUNT to the terminal by running **aws configure** command. since I don’t have an IAM user yet, I would create one with programmatic access.





Access key and secret keys have been added successfully using the **aws configure** command.

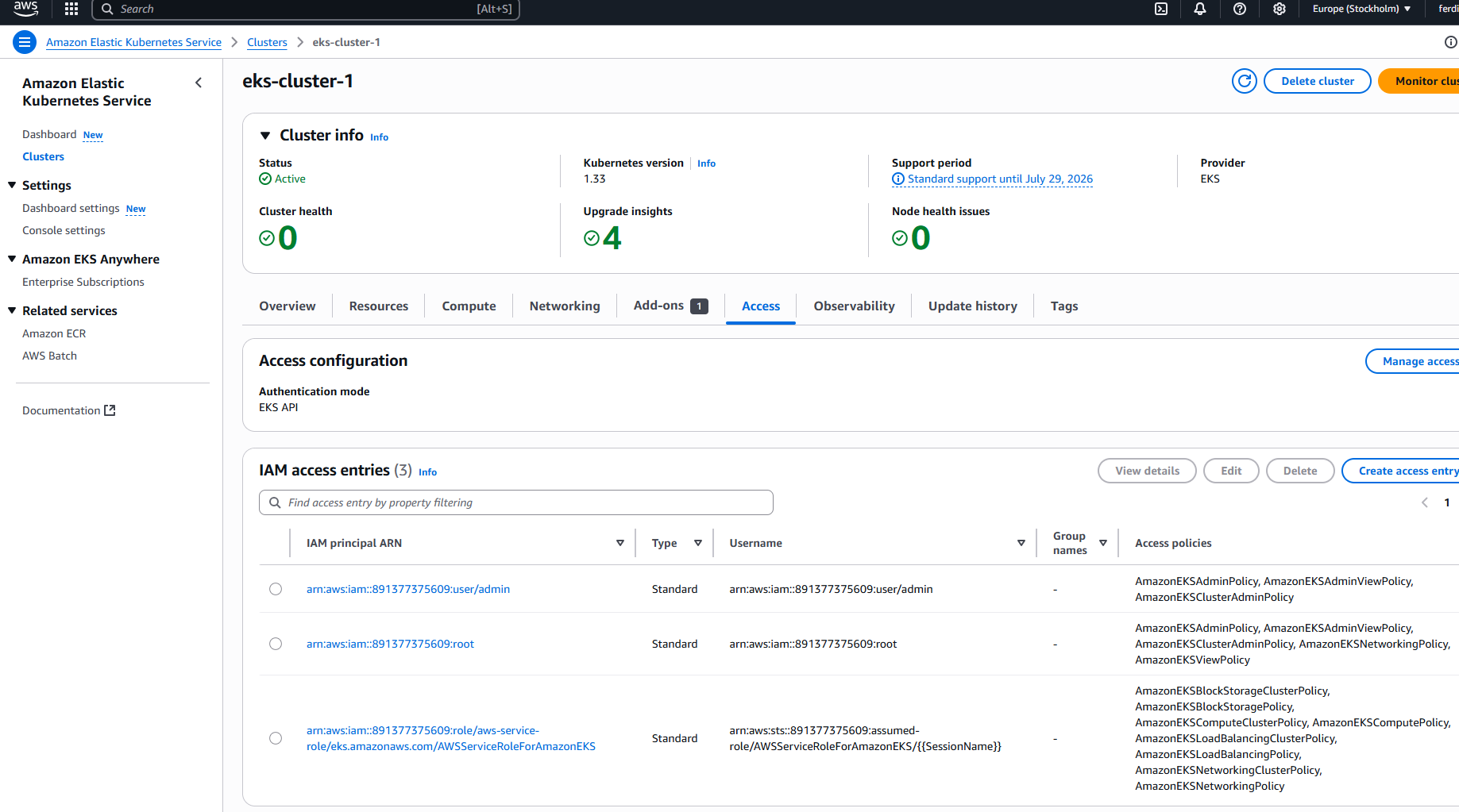


Create **kubeconfig**  to connect to the EKS CLUSTER

|  |
| --- |
| **$ aws eks update-kubeconfig --name eks-cluster-test** |

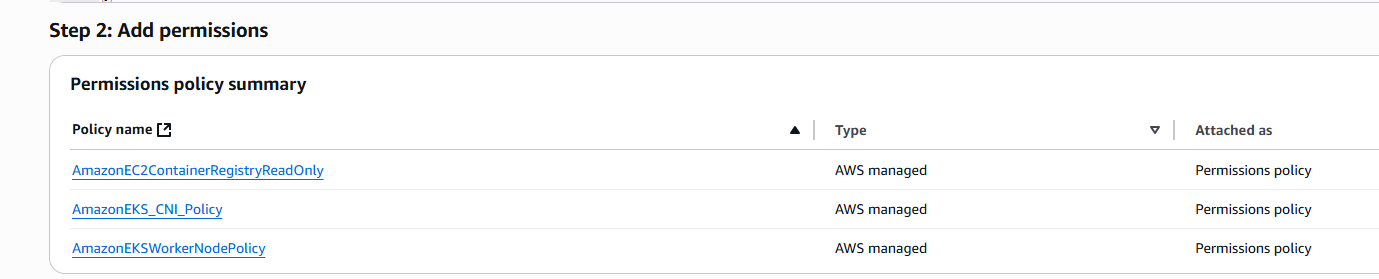


Make sure you add required cluster access for the user ACCESS TOKEN/ACCESS ID which was pre-generated.

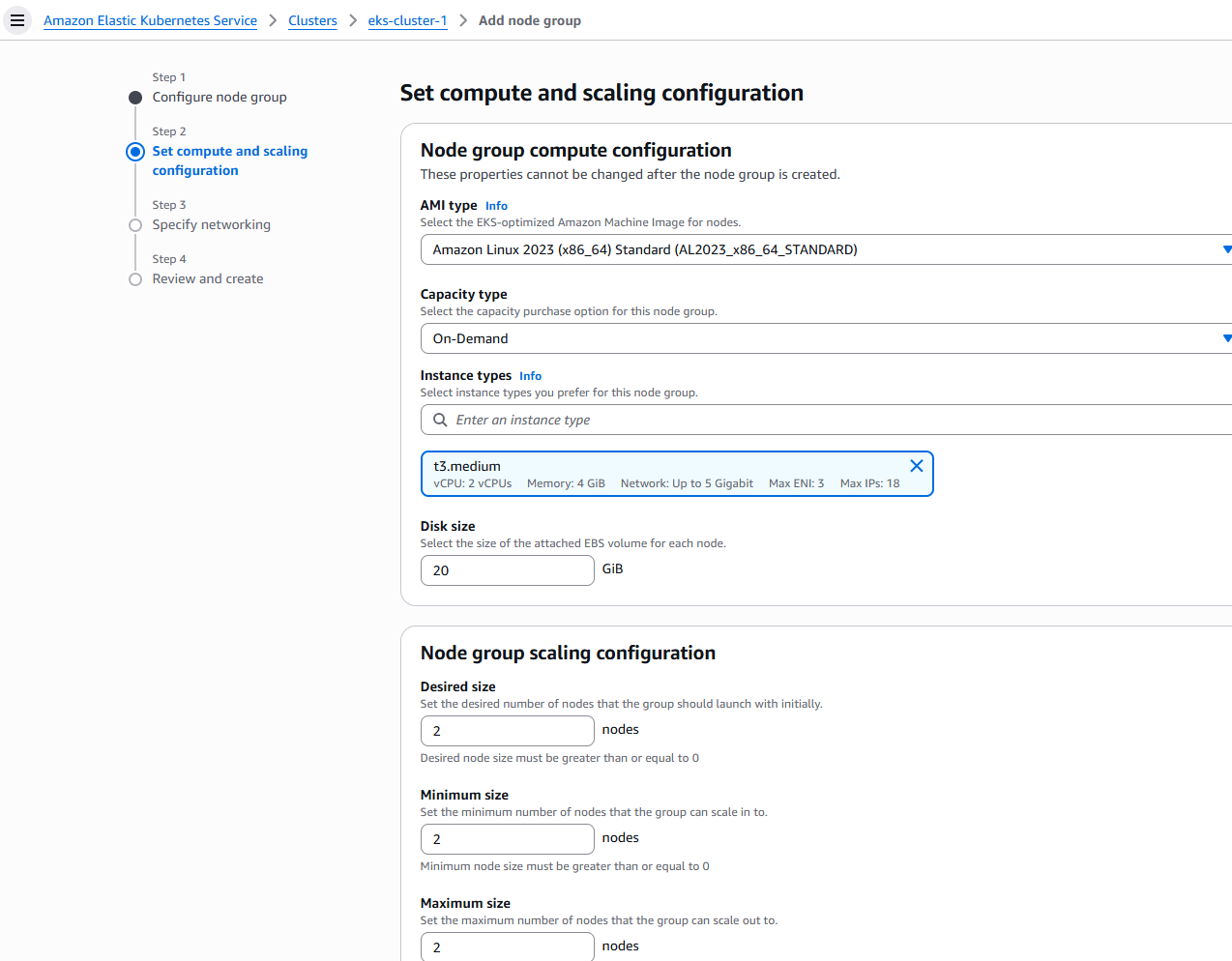


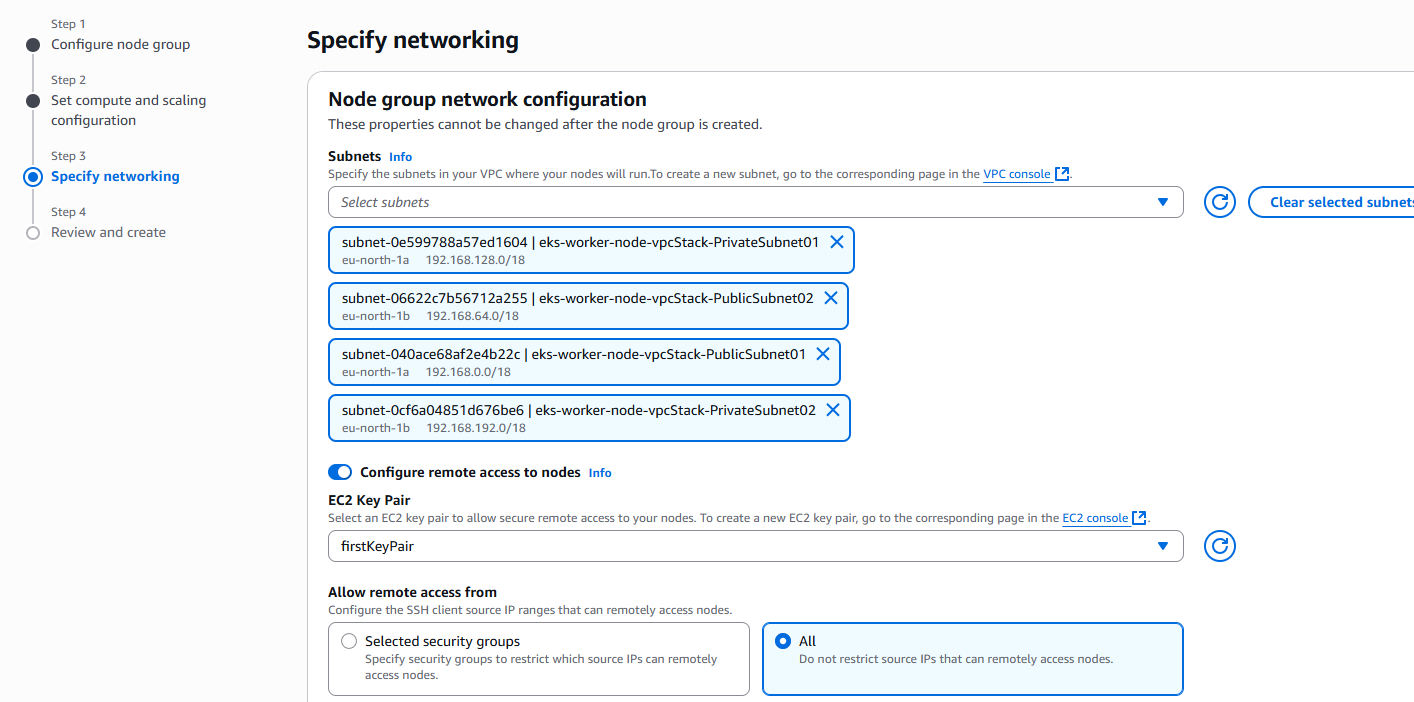
Create EC2 IAM Role for our Node group (worker nodes)

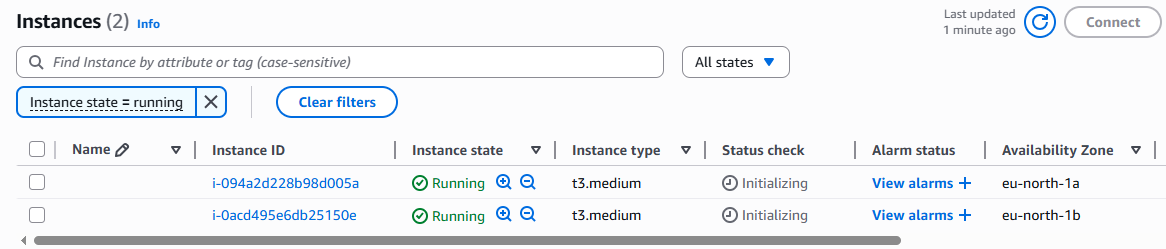
* Worker nodes run worker processes
* Kubelet is the **main process –** scheduling, managing pods and **communicate with other AWS services**
* Kubelet on worker Node(EC2 Instance) needs **permission**
* Create a **Role for Node Group**

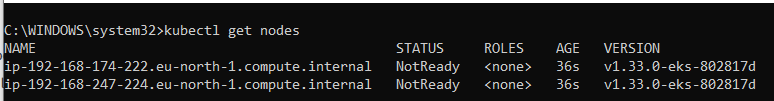
now create nodeGroup

EKS => Compute => Node Group









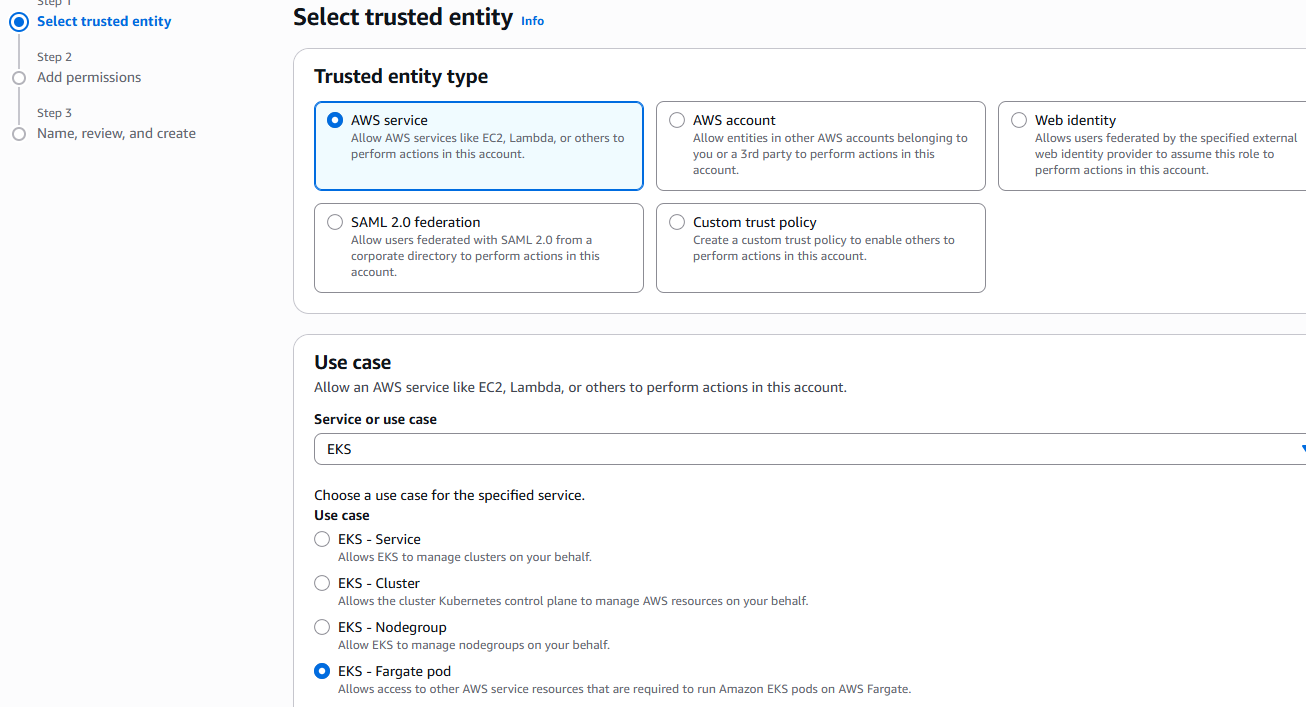
In order to run pods on these EC2 instances ( to make them K8s worker nodes), we need

* Container runtime (e.g. containerD)
* Kubelet
* Kubeproxy

All these processes get installed on the K8S servers when we create them using nodeGroup.

**DEPLOYING EKS CLUSTER WITH AWS FARGATE**

Create **Fargate Role** since Pods/kubelet on servers provisioned by Fargate needs **permission.**

****

**Then** create Fargate profile.

**What is a Fargate profile?**

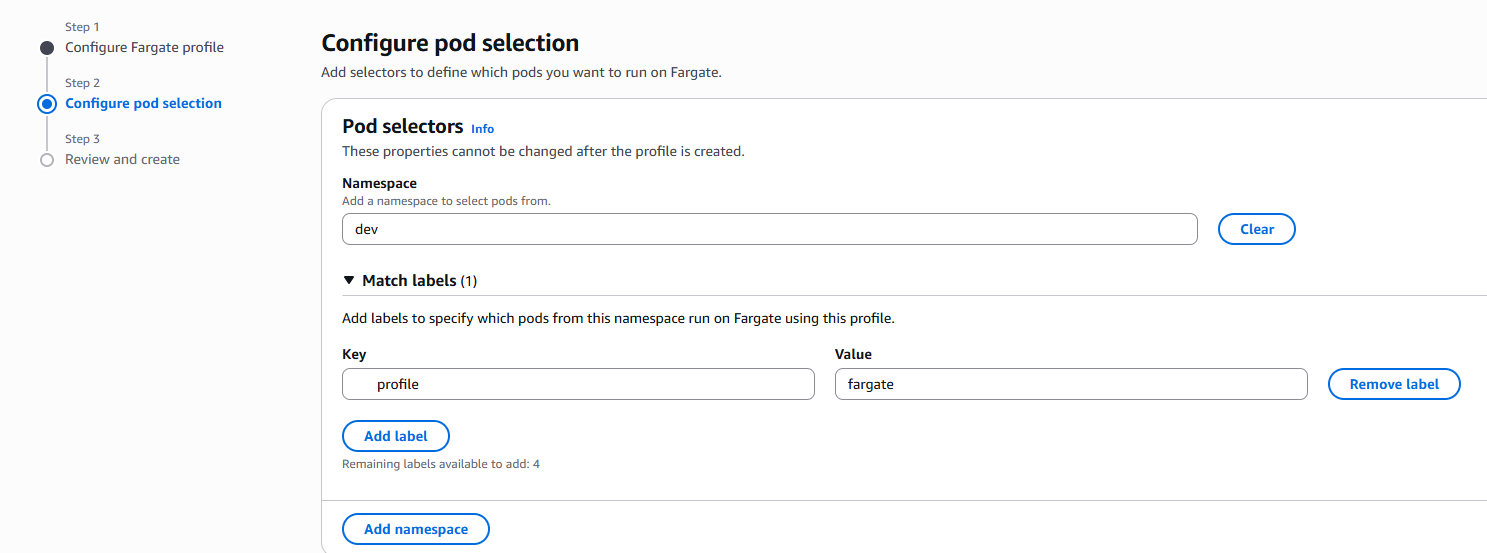
**-**it determines pod selection rule/how the pod should be scheduled.

- specifies which pods should use Fargate when they are launched.

E**.g. Use Cases – having both Node Group and Fargate**

**Y**ou can use both NodeGroup and Fargate in one cluster especially for projects that have **stateful applications** since Fargate alone can’t run stateful applications like databases. So for these type of applications, use Fargate + Node Groups

Continuation with FargateProfile creation.



**DEPLOY A POD THROUGH FARGATE**

Create sample YAML to deploy nginx and configure ;

* Namespace: dev
* Key: profile
* Value: fargate

Go to terminal, create the namespace called **dev**

|  |
| --- |
| **$ kubectl create ns dev** |

**DEPLOYING A KUBERNETES CLUSTER WITH eksctl**

It is a command line tool for working with EKS Clusters that **automates** many individual tasks.

* Executes just one command
* Necessary components gets created and configured in the background
* Cluster will be created with default parameters
* With more cli options, you can customize your cluster

We can install **eksctl**  on windows using **chocolatey** package manager

|  |
| --- |
| **$ choco install eksctl** |

Once the installation is complete, create an eks cluster using eksctl

|  |
| --- |
| $ eksctl create cluster \  > --name demo-cluster \  > --version 1.33 \  > --region eu-north-1 \  > --nodegroup-name demo-nodes \  > --node-type t3.micro \  > --nodes 2 \  > --nodes-min 1 \  > --nodes-max 3 |

**Eksctl documentation:** https://eksctl.io/