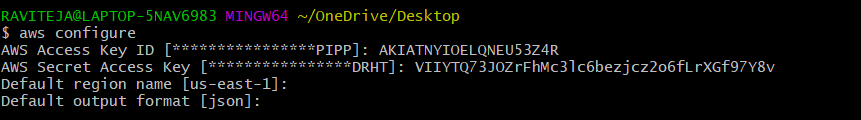
1. Setup eks cluster using eksctl



 **Download eksctl for Windows (AMD64)**

bash

CopyEdit

curl -LO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl\_windows\_amd64.zip"

 **Unzip it**

bash

CopyEdit

unzip eksctl\_windows\_amd64.zip

 **Move it to a folder in your PATH**  
For example, if you have C:\tools in your PATH:

bash

CopyEdit

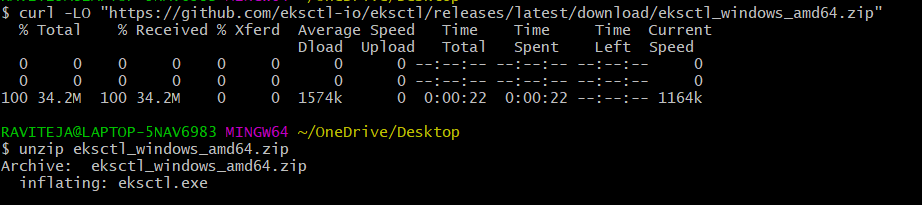
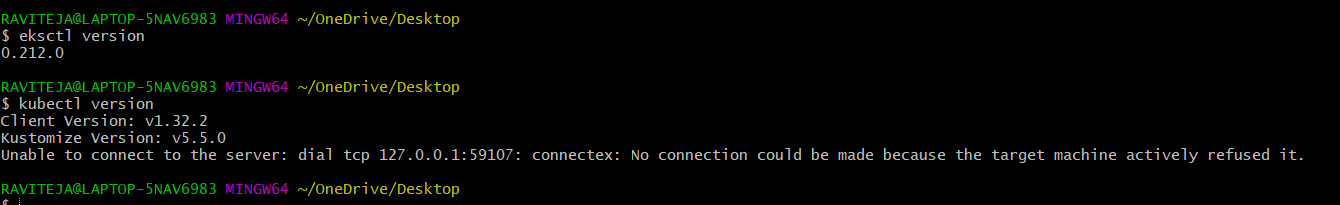
mv eksctl.exe /c/tools/

 **Verify installation**

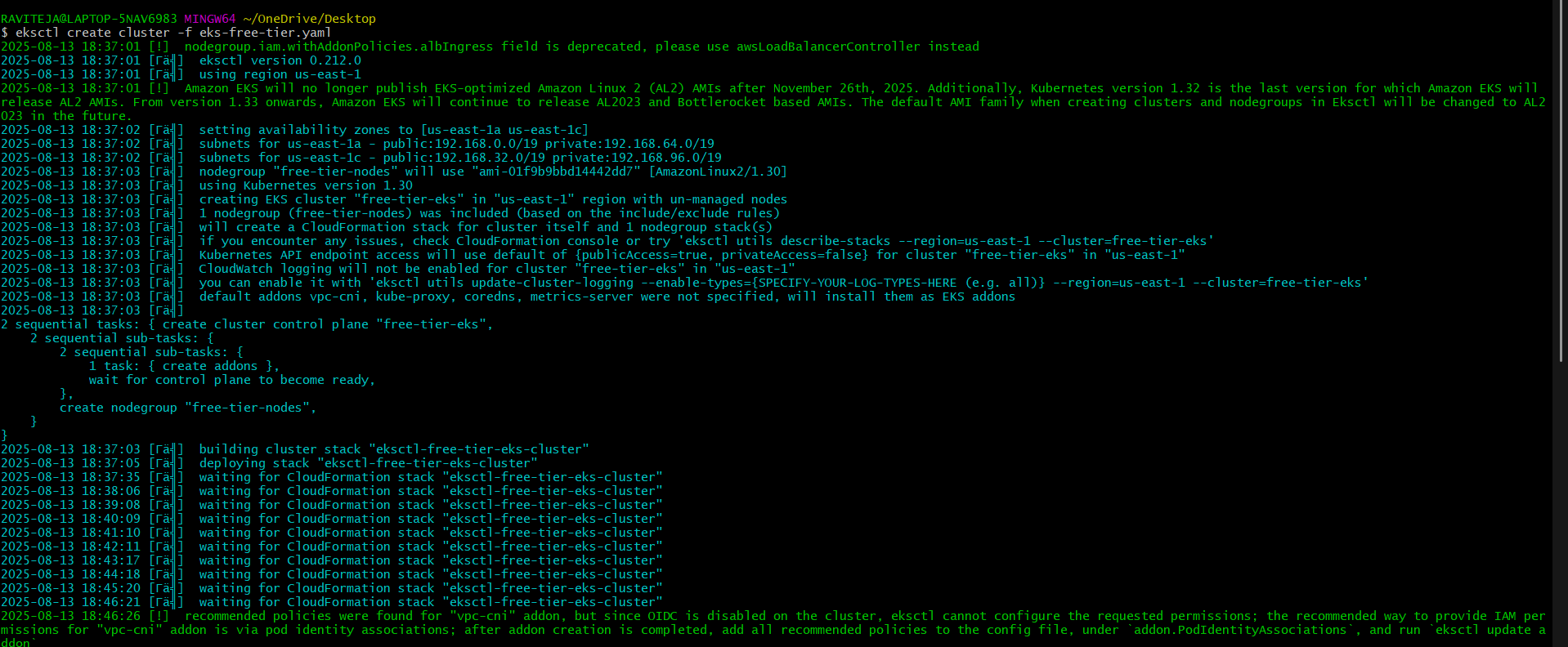
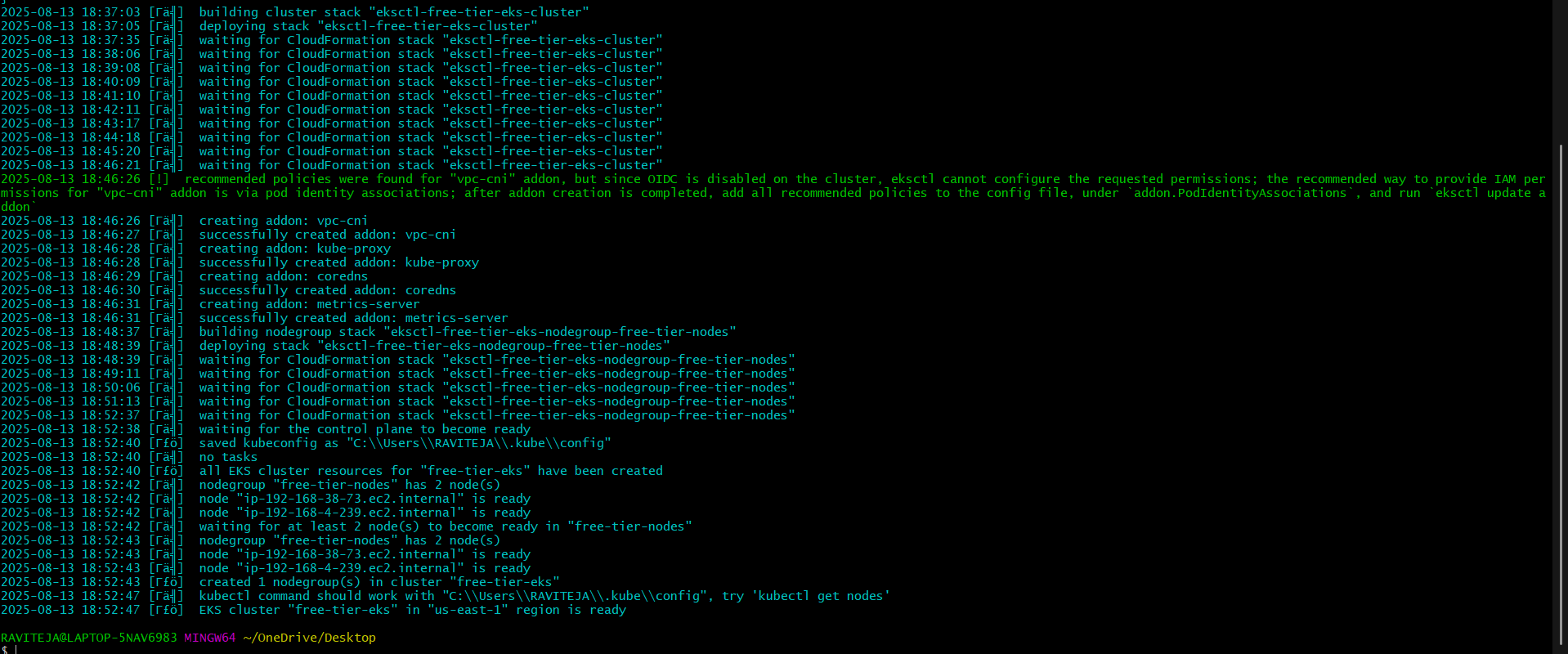
bash

CopyEdit

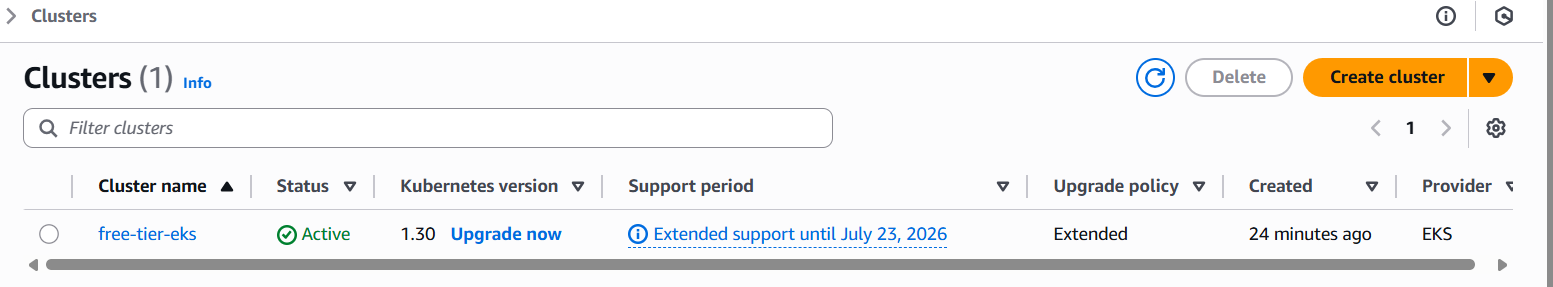
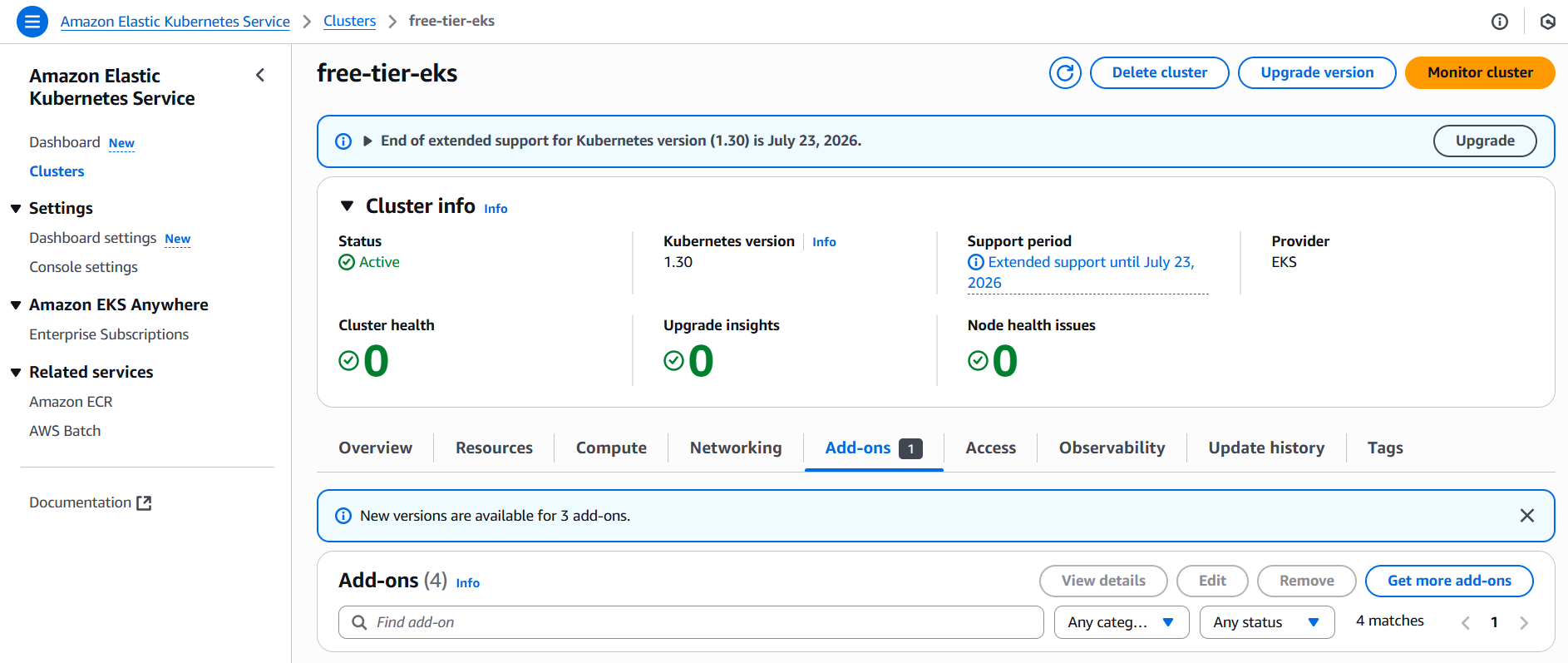
eksctl version

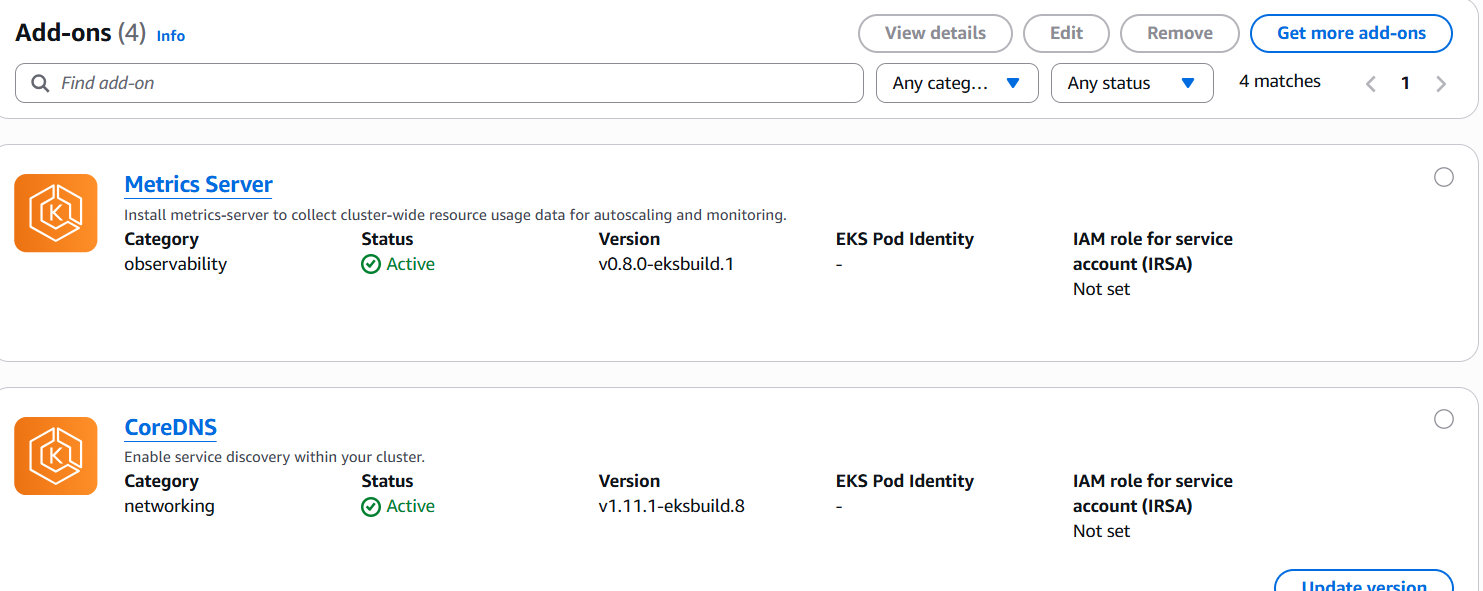
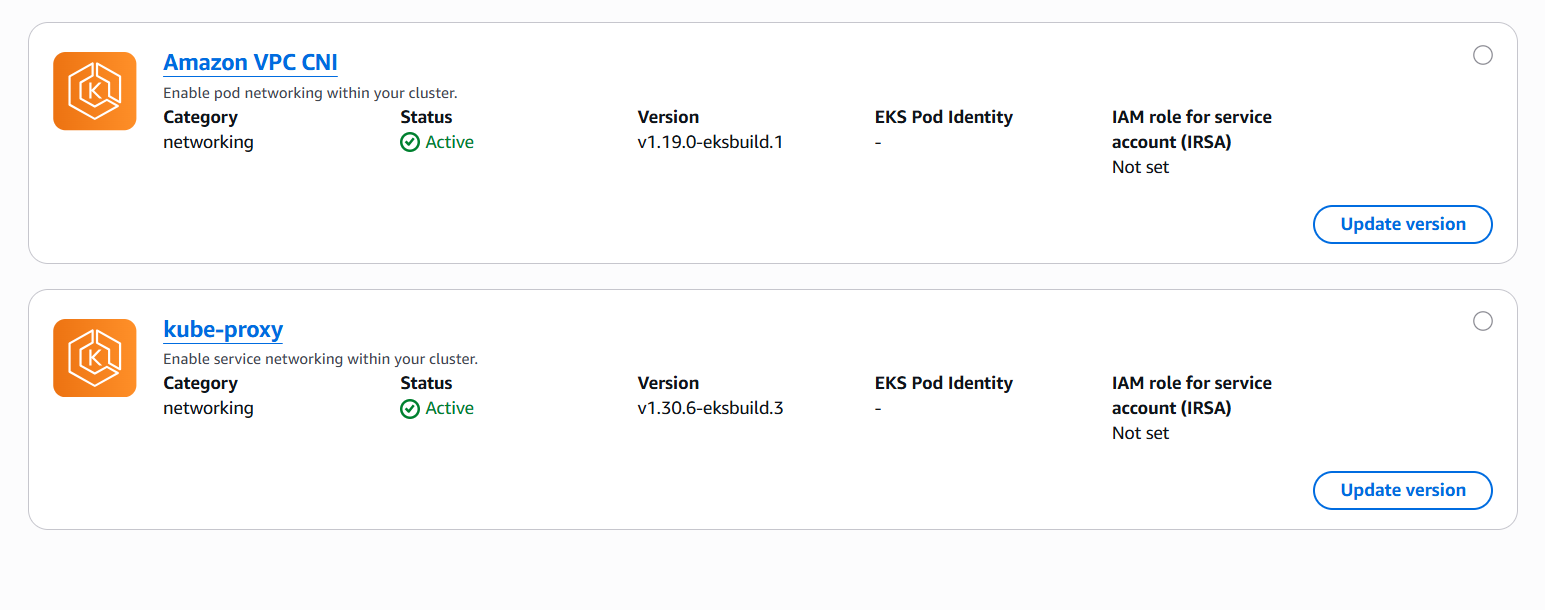
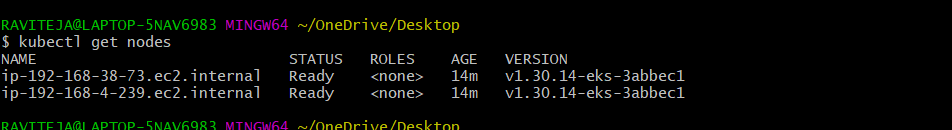
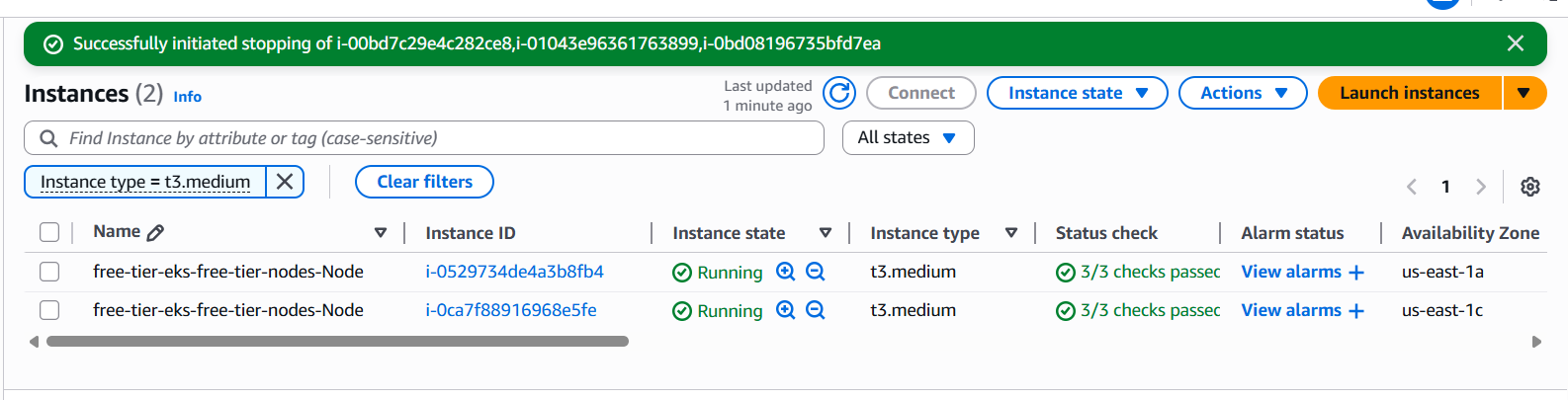
Created the cluster:

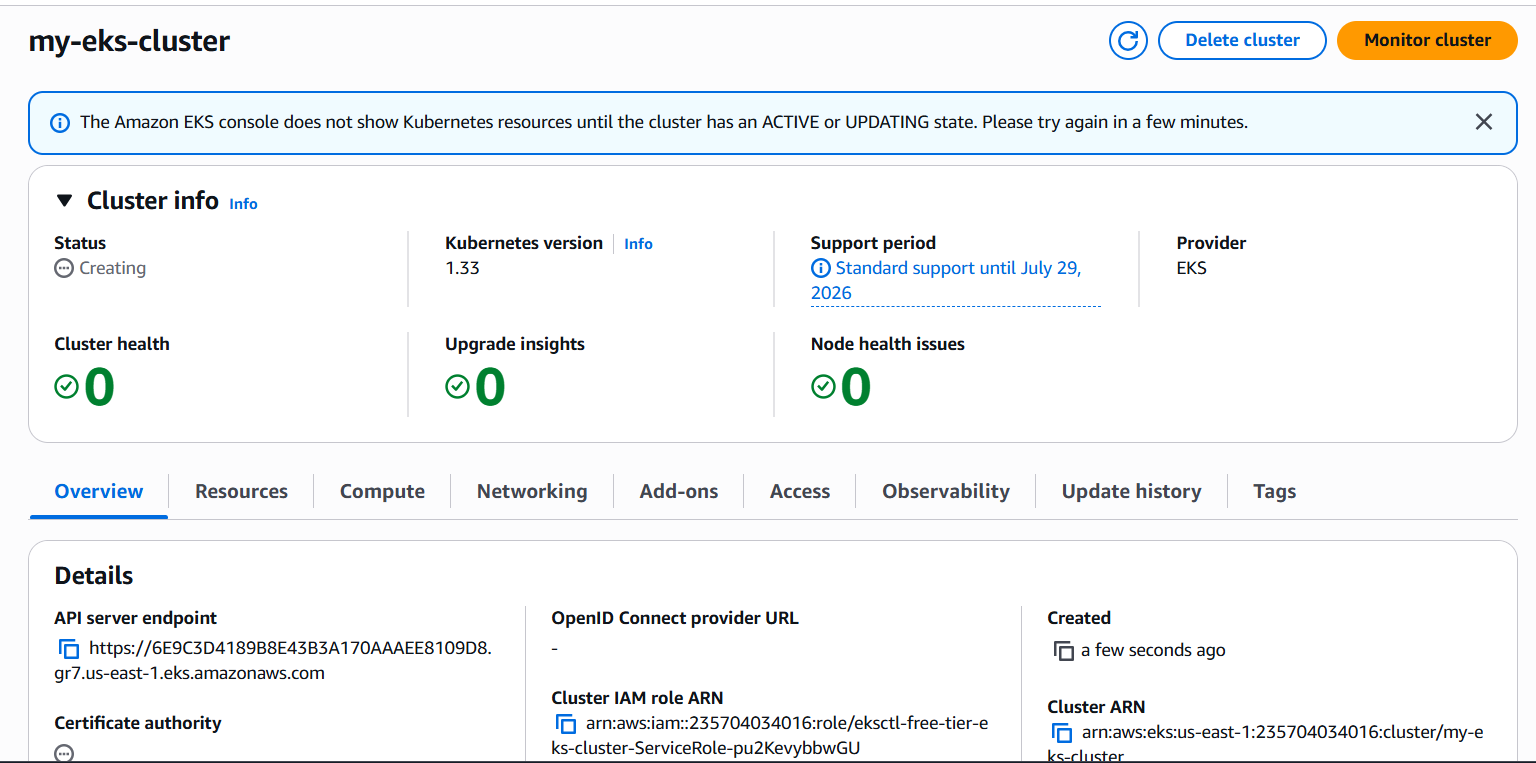
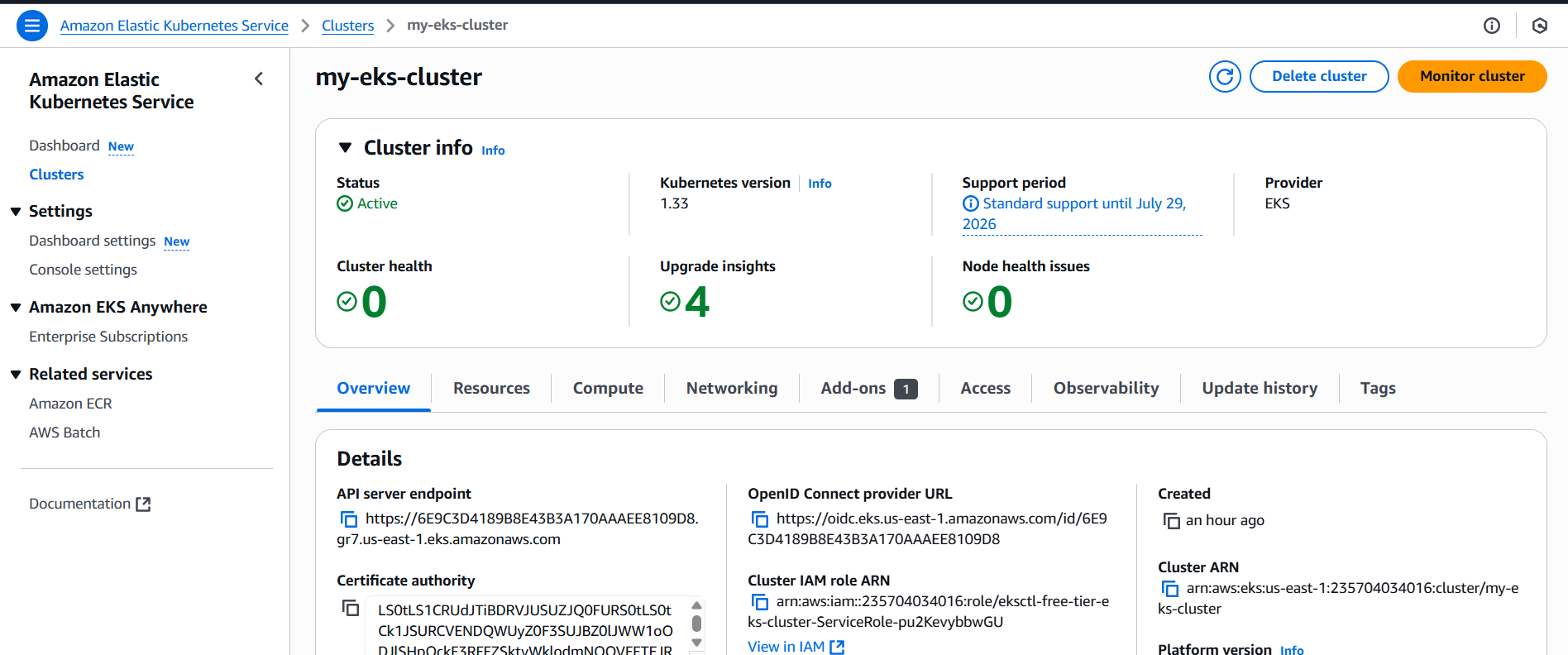
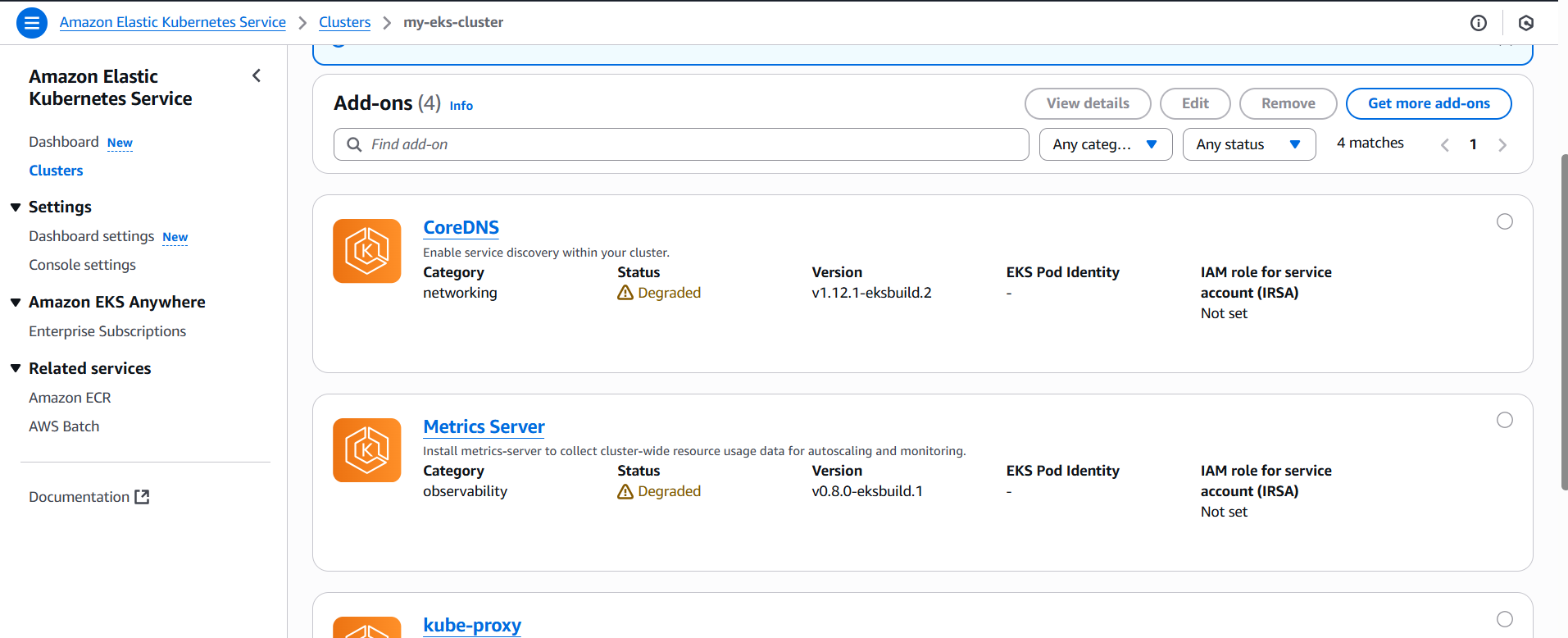
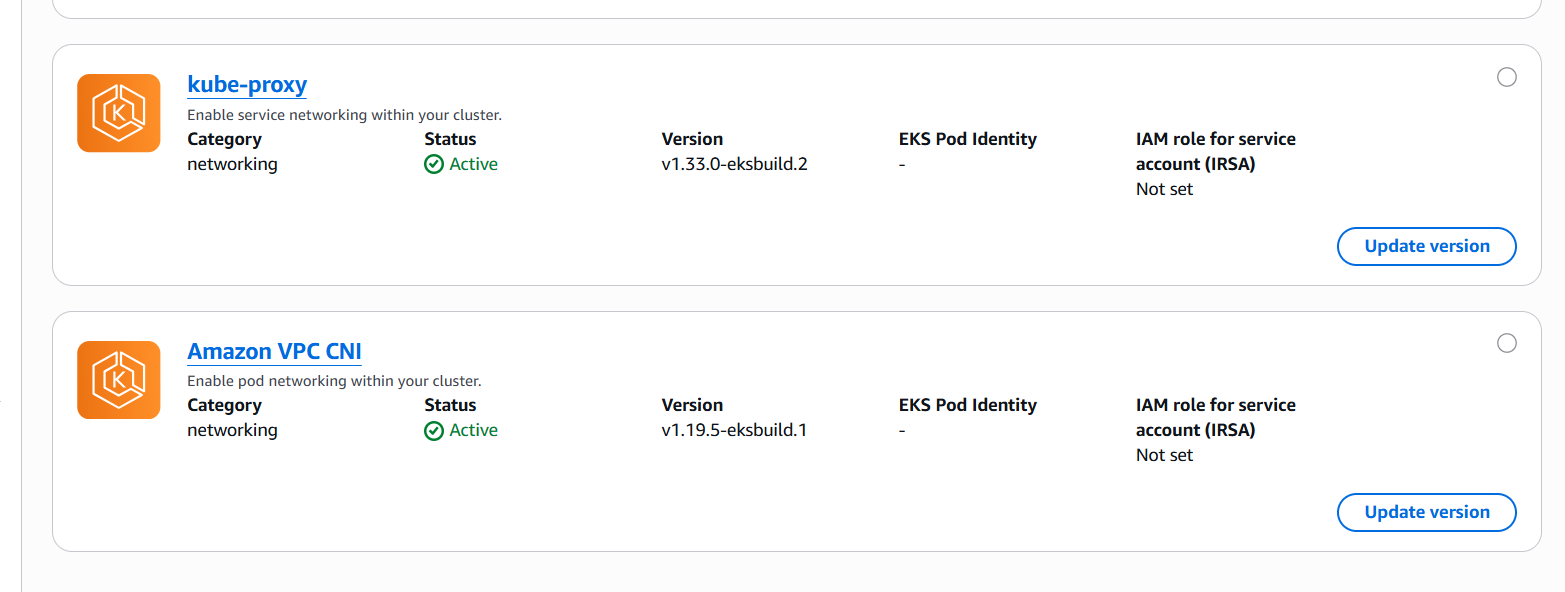
Cluster

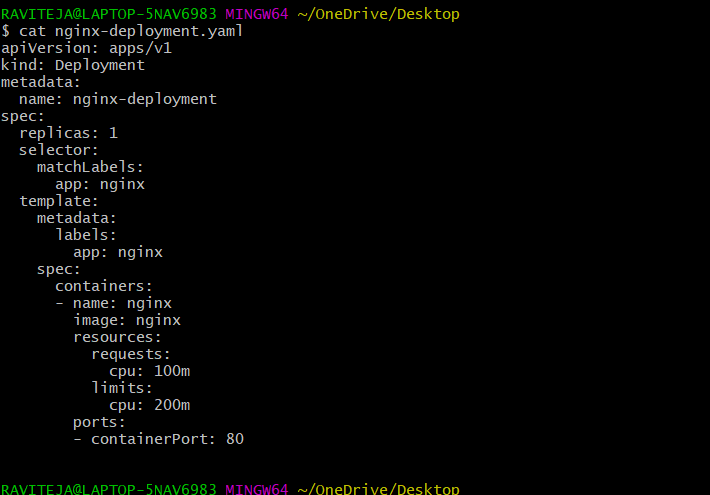
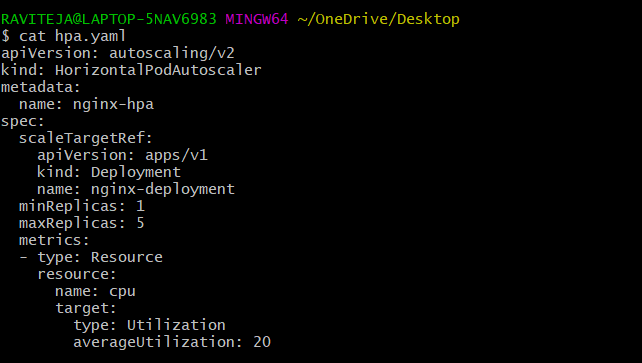
ADD ONS:

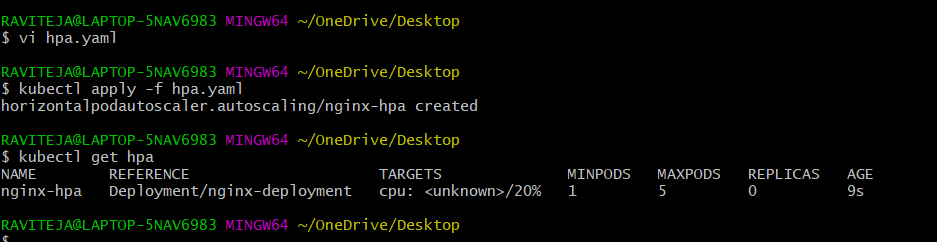
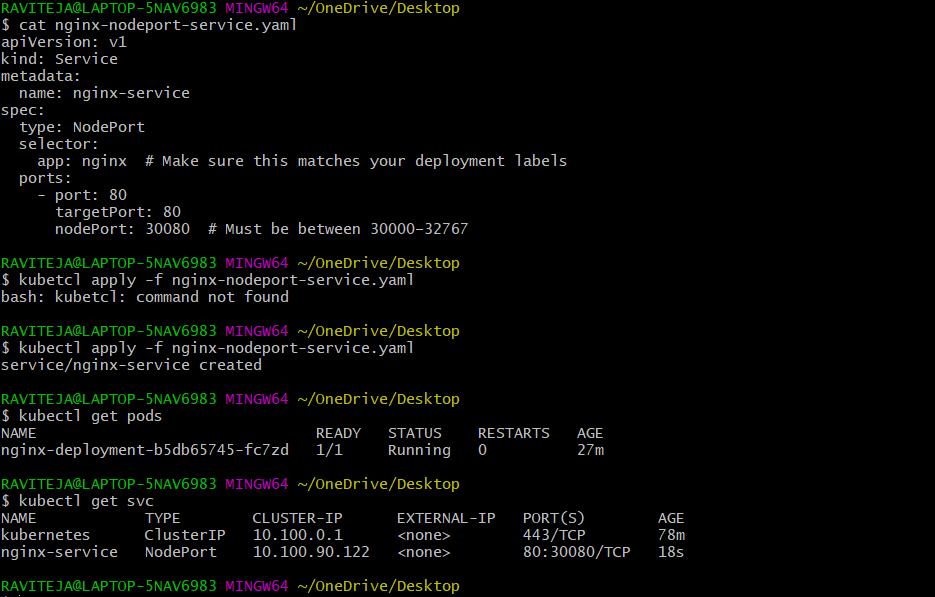
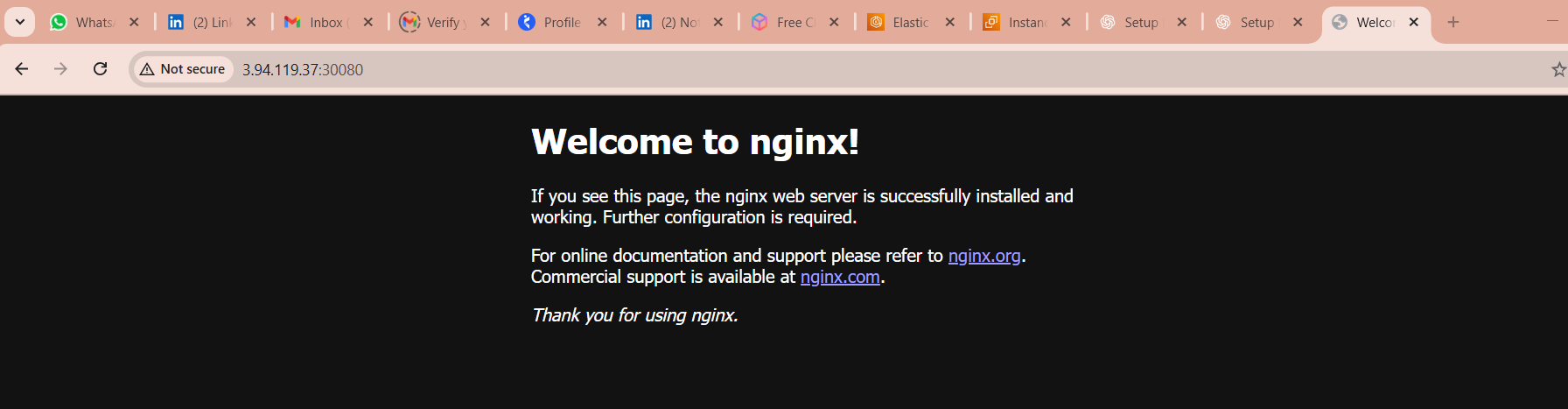
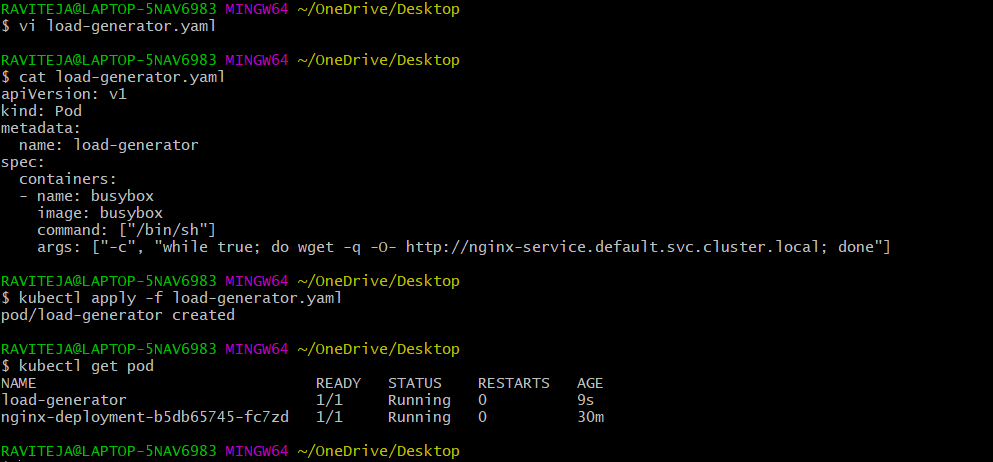
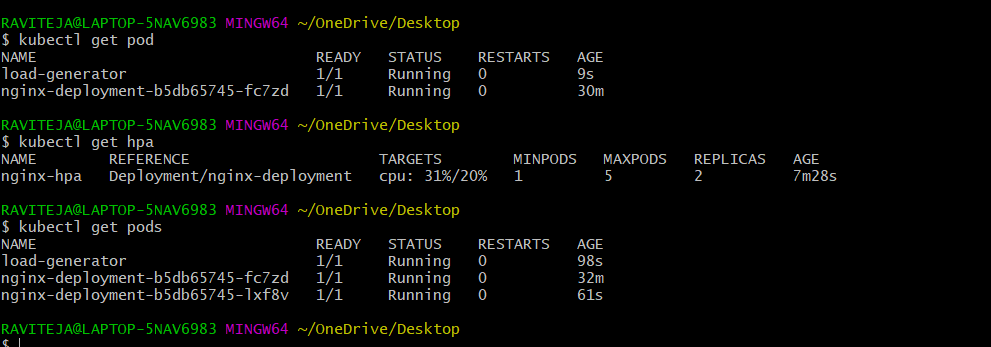
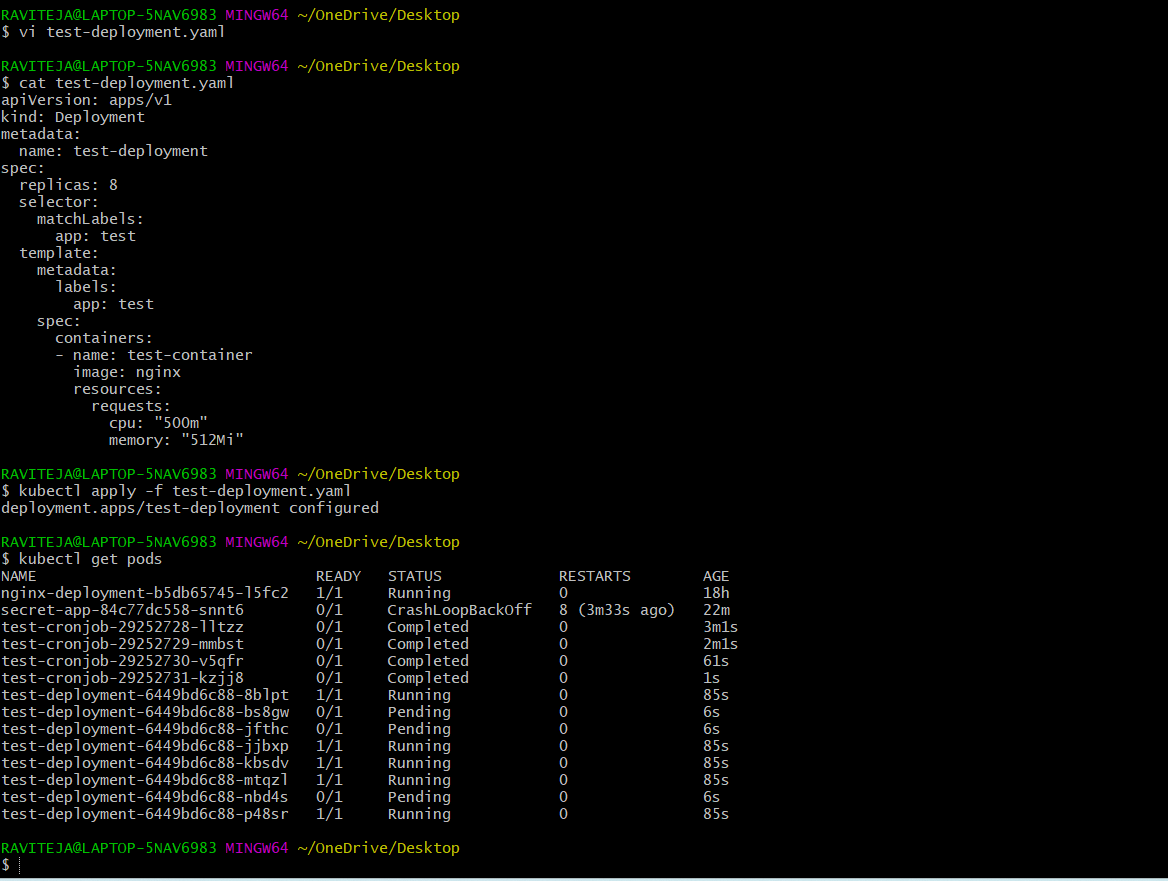
   

2) setup eks cluster using console

3) Setup HPA

      
4) Setup cluster autoscale  


Get the name of autoscaling group.

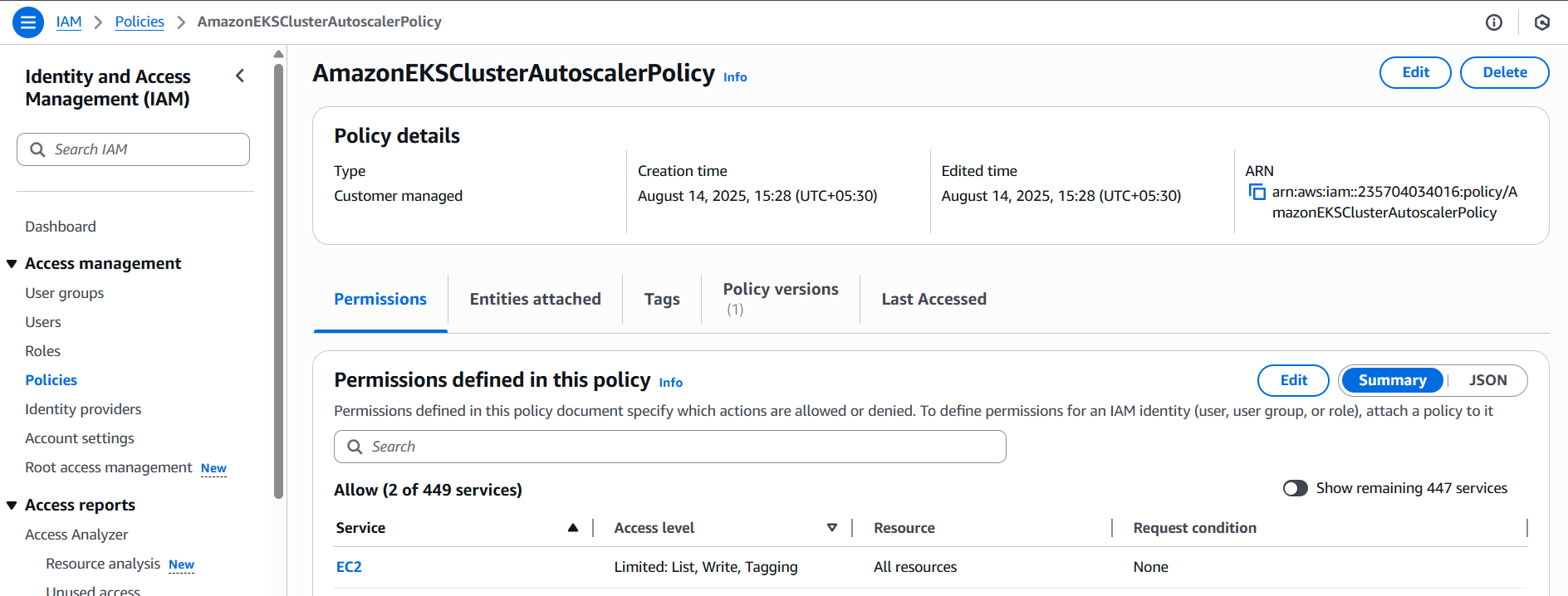
# aws autoscaling describe-auto-scaling-groups --query "AutoScalingGroups[\*].AutoScalingGroupName" --output table

2) update the tags with name and execute below command.

# aws autoscaling create-or-update-tags --tags \

ResourceId= eksctl-free-tier-eks-nodegroup-free-tier-nodes-NodeGroup-Jl2Txy2bqFU7,ResourceType=auto-scaling-group,Key=k8s.io/cluster-autoscaler/enabled,Value=true,PropagateAtLaunch=true \

ResourceId= eksctl-free-tier-eks-nodegroup-free-tier-nodes-NodeGroup-Jl2Txy2bqFU7,ResourceType=auto-scaling-group,Key=k8s.io/cluster-autoscaler/medium-eks-cluster,Value=owned,PropagateAtLaunch=true



eksctl create iamserviceaccount \

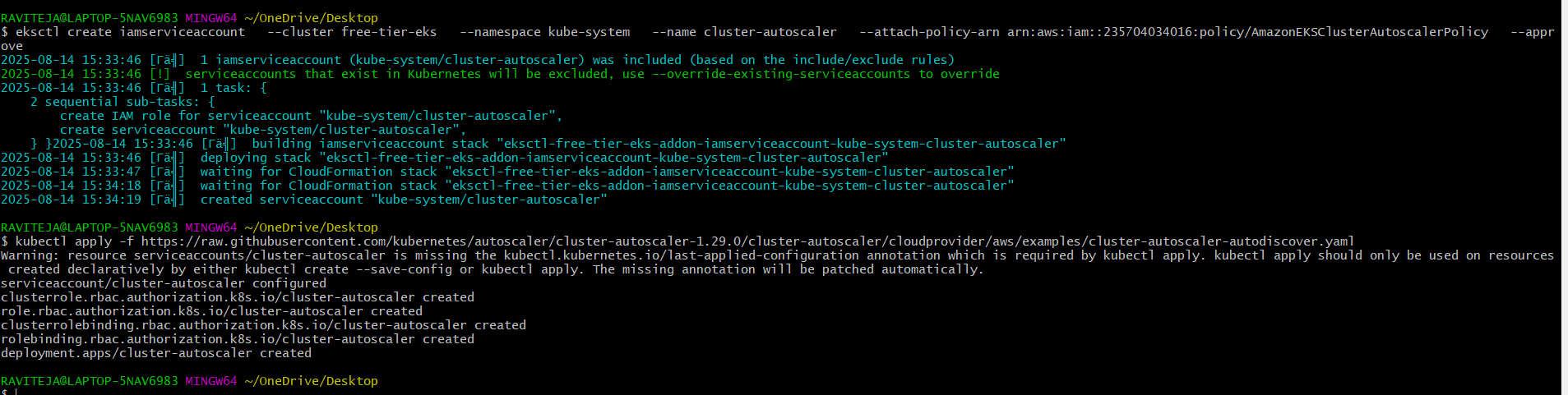
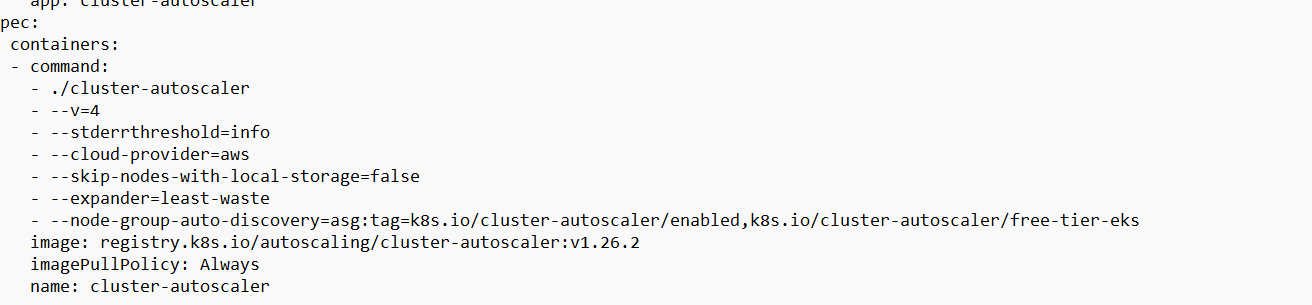
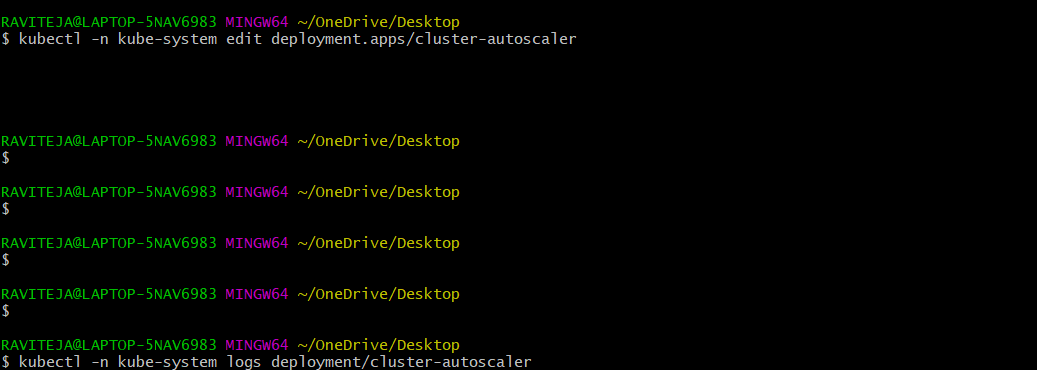
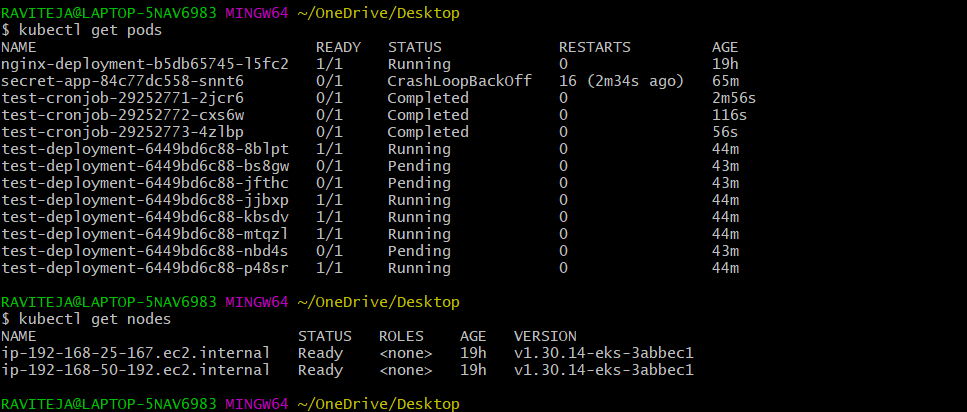
--cluster free-tier-eks \

--namespace kube-system \

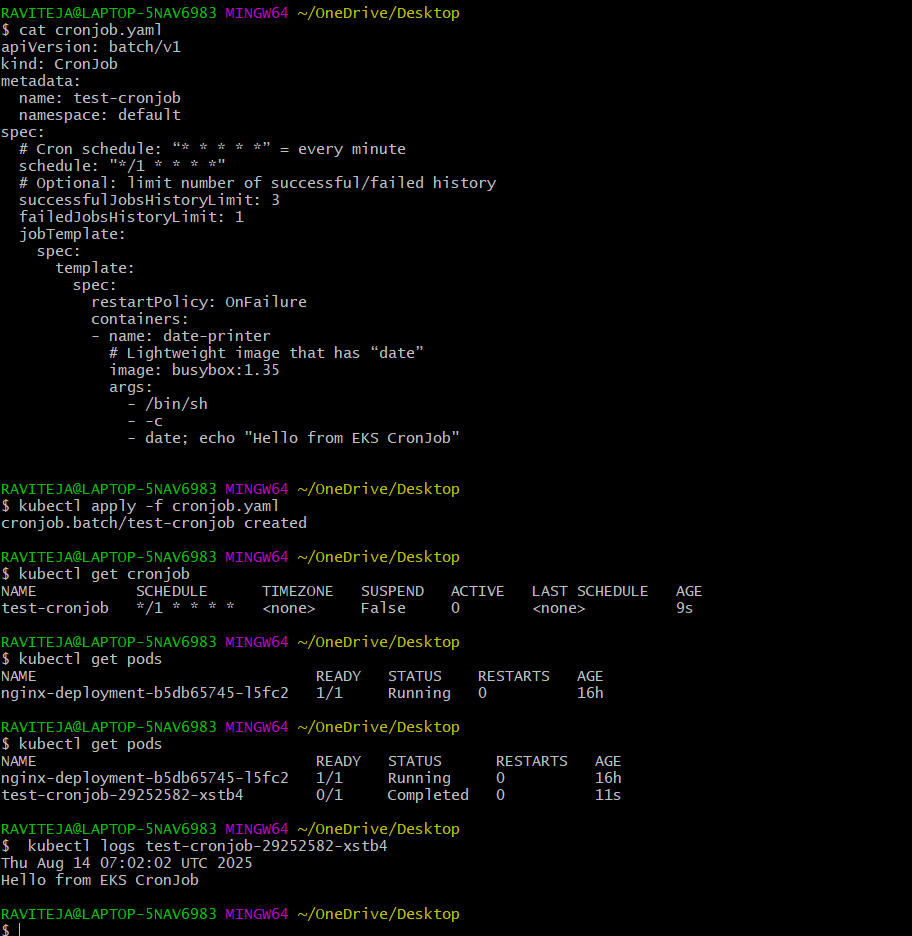
--name cluster-autoscaler \

--attach-policy-arn arn:aws:iam:: 235704034016: policy/AmazonEKSClusterAutoscalerPolicy \

--approve

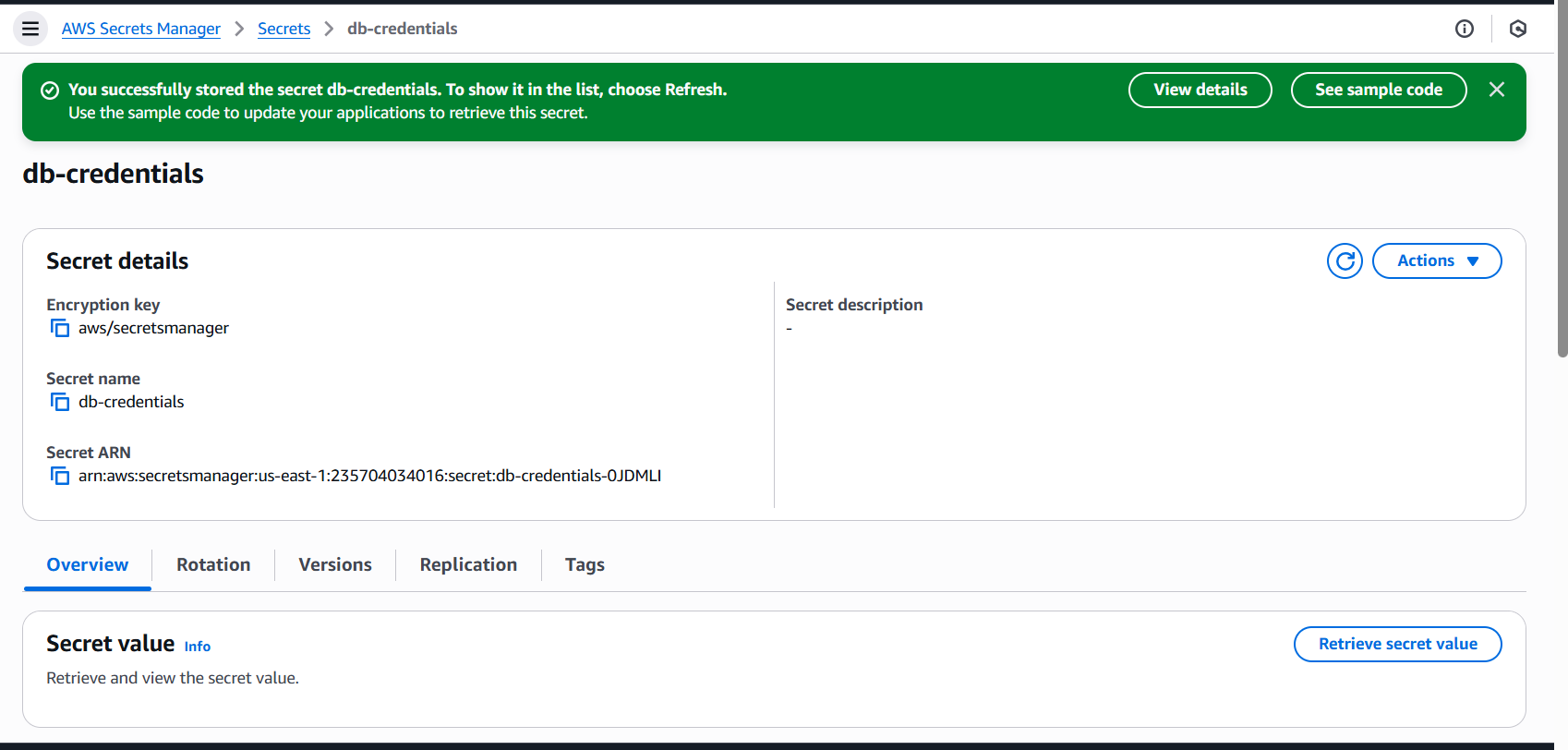
    

5) Setup cronjob and job



6) Create secret and inject inside pod

Create secret in aws secret manager:



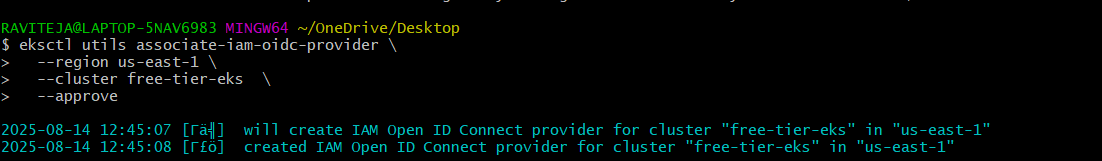
Enable OIDC:

eksctl utils associate-iam-oidc-provider \

--region us-east-1 \

--cluster free-tier-eks \

--approve



Create a policy & Role

4) Create iam policy using below json template.

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

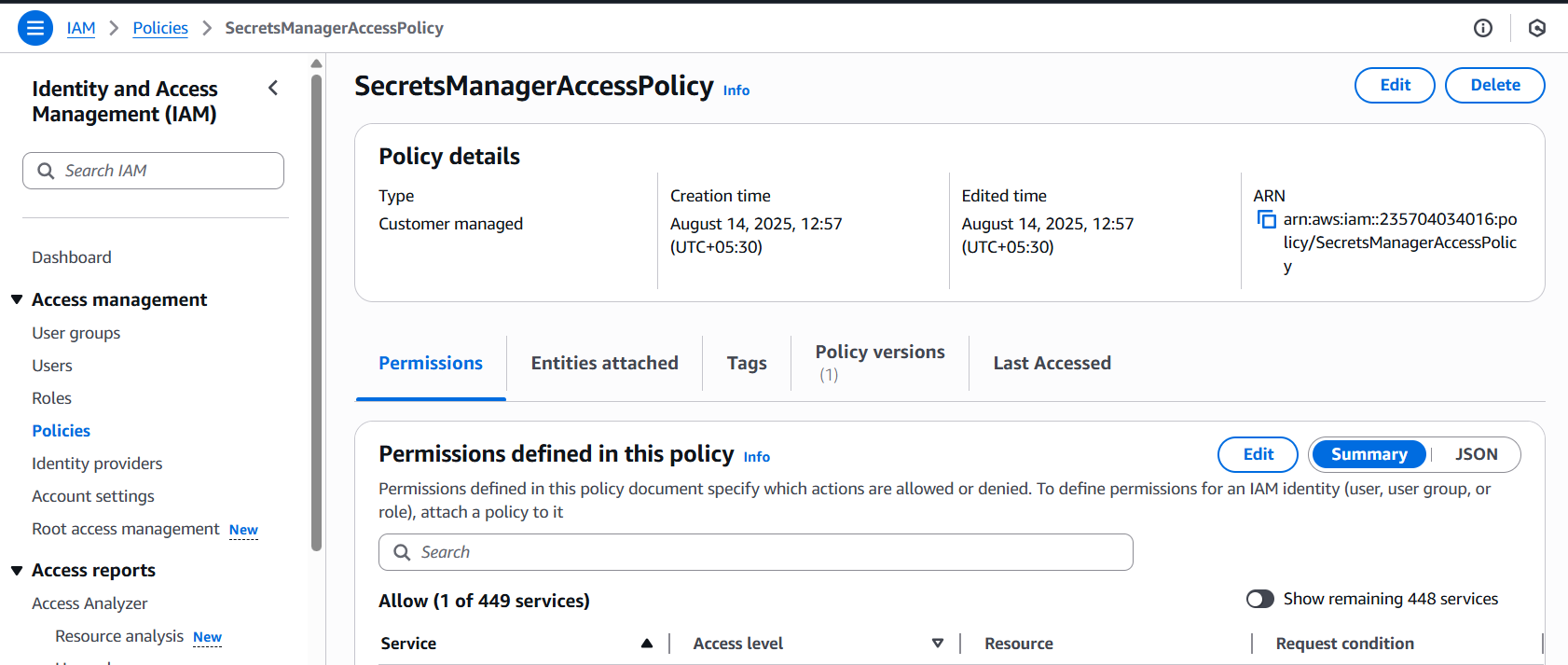
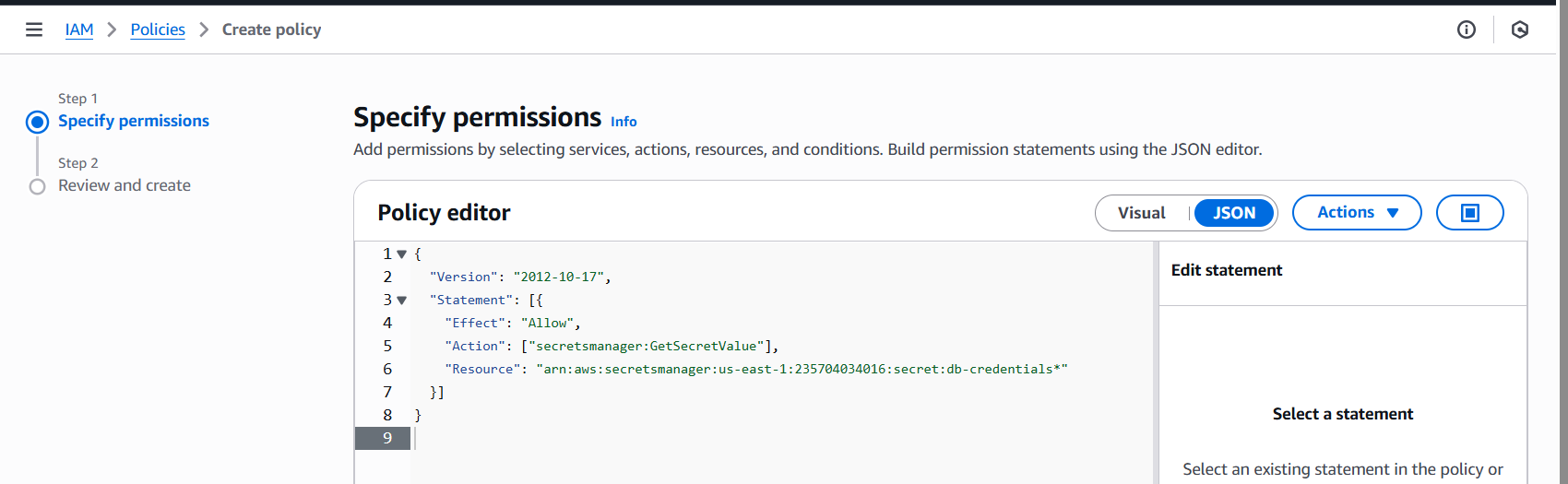
"Action": ["secretsmanager:GetSecretValue"],

"Resource": "arn:aws:secretsmanager:us-east-1: 235704034016:secret:db-credentials\*"

}]

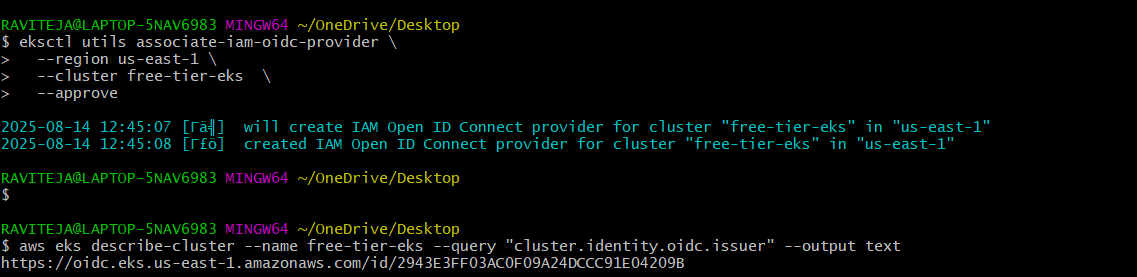
}

Name of policy: SecretsManagerAccessPolicy



To check the oidc:

aws eks describe-cluster --name my-eks-cluster --query "cluster.identity.oidc.issuer" --output text



5) Create one role with custom trust policy using below json format and attach the policy created in step 4.

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Principal": {

"Federated": "arn:aws:iam::915992498174:oidc-provider/oidc.eks.us-east-1.amazonaws.com/id/2943E3FF03AC0F09A24DCCC91E04209B"

},

"Action": "sts:AssumeRoleWithWebIdentity",

"Condition": {

"StringEquals": {

"oidc.eks.us-east-1.amazonaws.com/id/2943E3FF03AC0F09A24DCCC91E04209B:sub": "system:serviceaccount:default:secrets-sa"

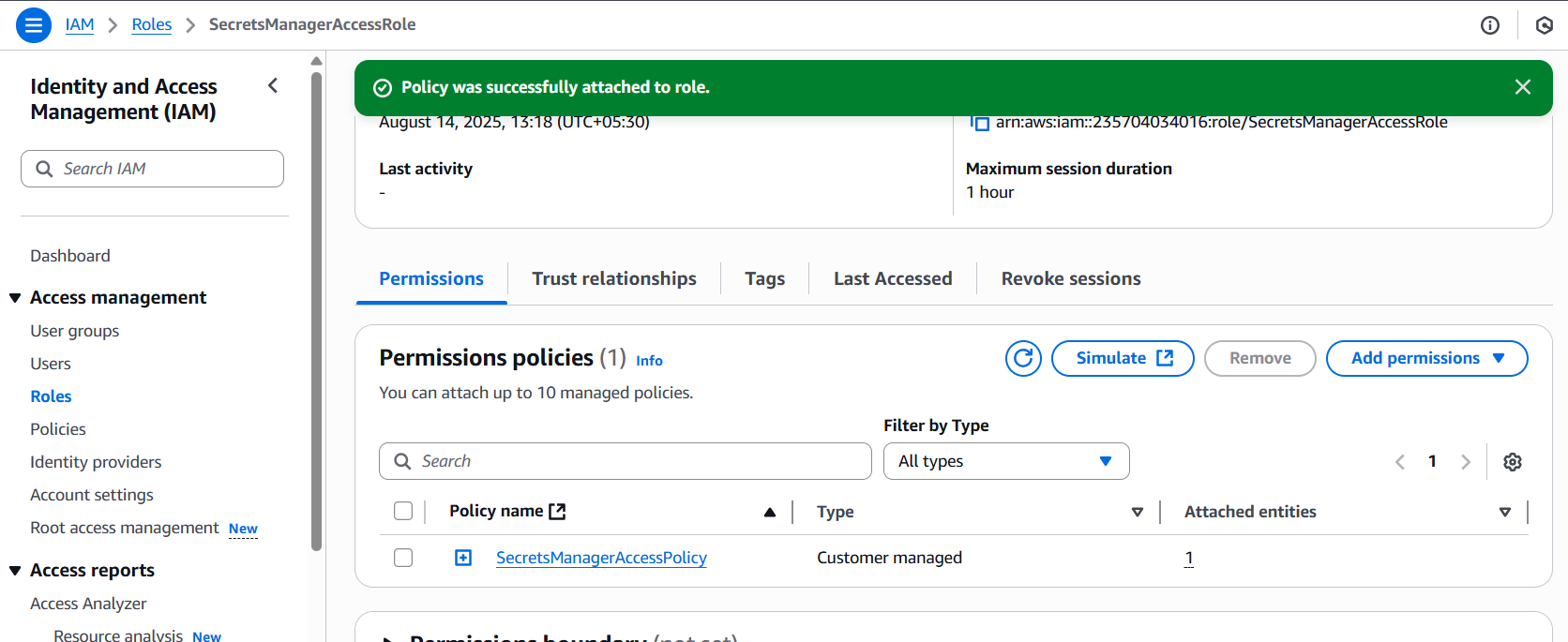
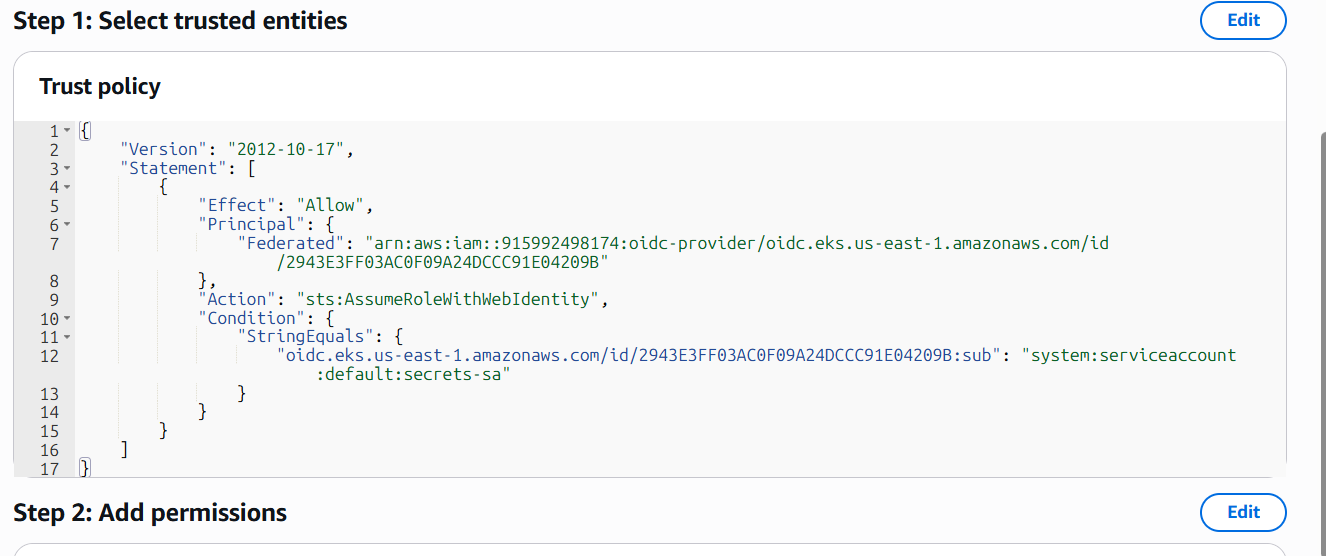
}

}

}]

}

2943E3FF03AC0F09A24DCCC91E04209B



Create Kubernetes Service Account

apiVersion: v1

kind: ServiceAccount

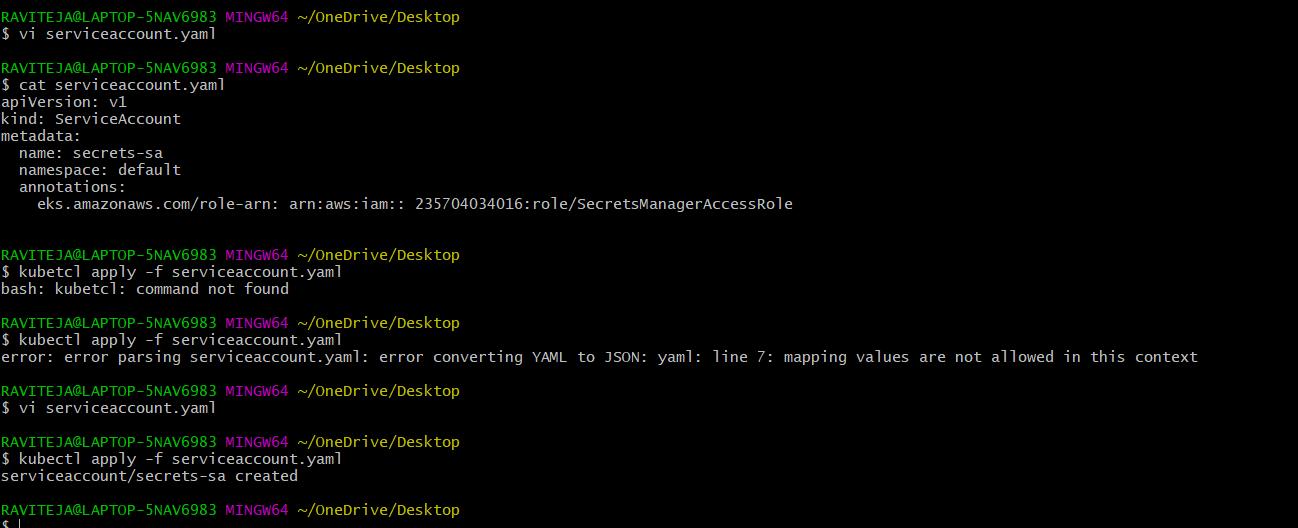
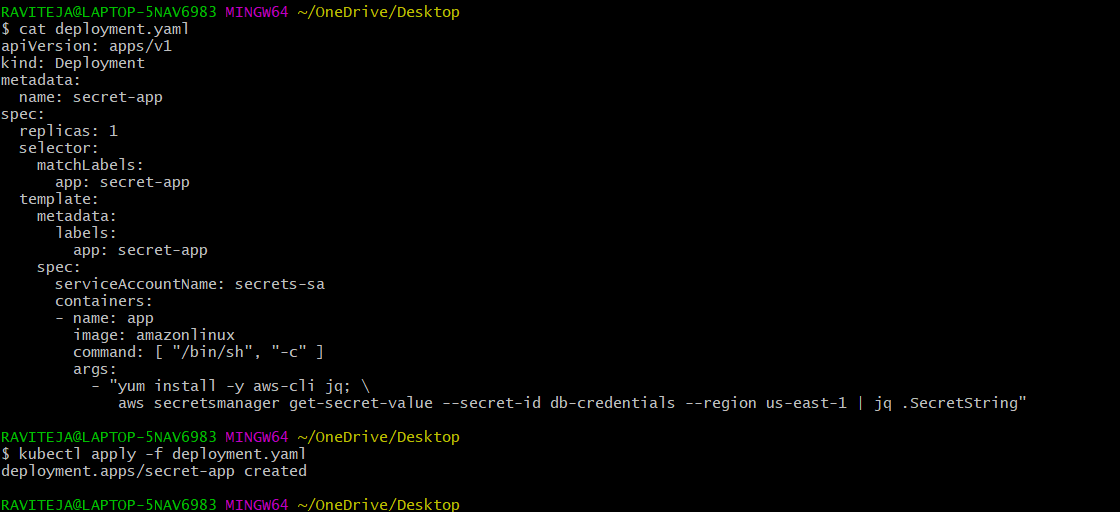
metadata:

name: secrets-sa

namespace: default

annotations:

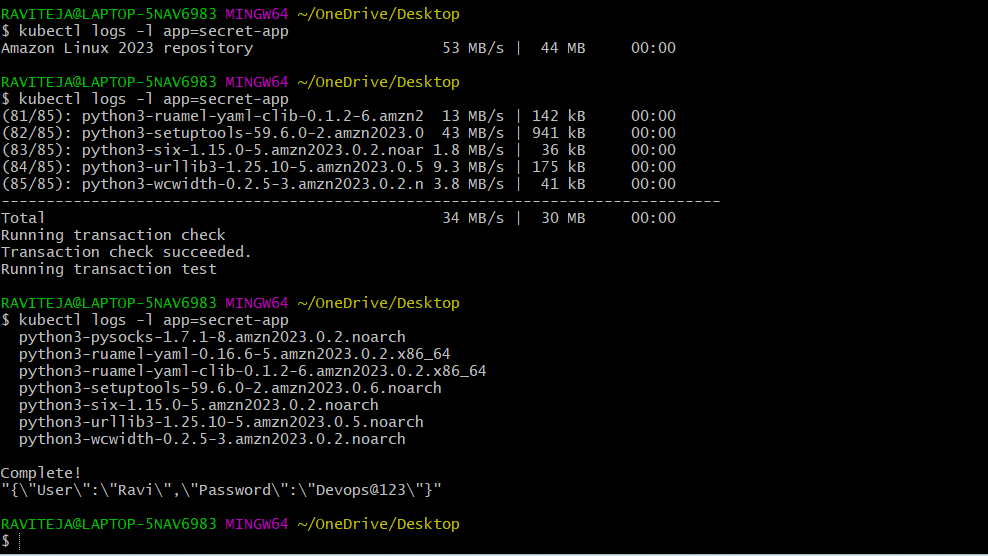
eks.amazonaws.com/role-arn: arn:aws:iam:: 235704034016:role/SecretsManagerAccessRole

Test the credentials by checking logs.

kubectl logs -l app=secret-app

you can see the secret details in logs.



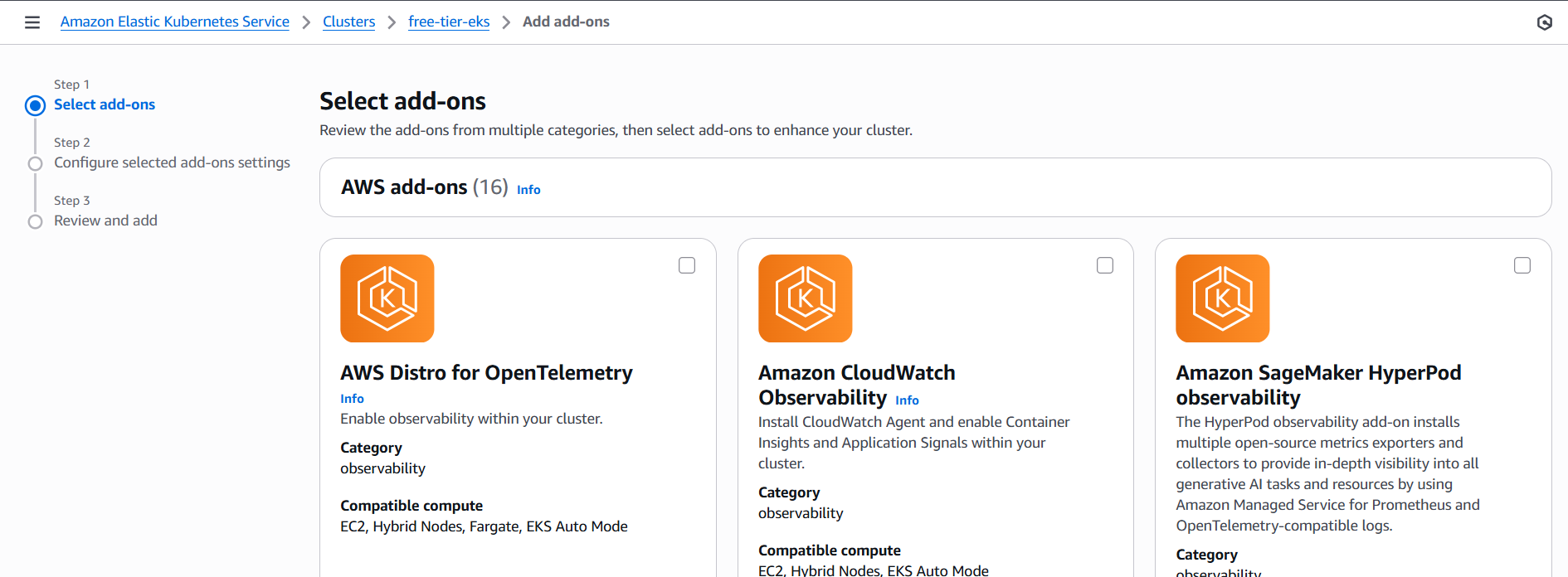
7) Check different add-ons available on eks.

aws eks describe-addon-versions \

--region <your-region> \

--query "addons[].addonName" \

--output table



On **Amazon EKS**, “add-ons” are prepackaged, AWS-managed Kubernetes components that you can enable directly from the EKS console or CLI.

Here’s a breakdown of the main **official EKS add-ons** you can install:

**Core AWS-Managed EKS Add-ons**

These are the most common ones:

| **Add-on Name** | **Purpose** |
| --- | --- |
| **Amazon VPC CNI** (vpc-cni) | Handles Kubernetes pod networking in AWS VPC. Allows pods to get VPC-native IP addresses. |
| **CoreDNS** (coredns) | Provides DNS for your Kubernetes services and pods. |
| **Kube Proxy** (kube-proxy) | Handles network rules for services in Kubernetes. |
| **Amazon EBS CSI Driver** (aws-ebs-csi-driver) | Lets pods use Amazon EBS volumes for persistent storage. |
| **Amazon EFS CSI Driver** (aws-efs-csi-driver) | Lets pods mount Amazon EFS file systems. |
| **Amazon FSx CSI Driver** (aws-fsx-csi-driver) | Enables FSx for Lustre/NetApp integration with Kubernetes. |
| **Amazon CloudWatch Observability** (amazon-cloudwatch-observability) | Simplifies logging, metrics, and traces collection. |
| **Amazon GuardDuty EKS Runtime Monitoring** | Adds security monitoring to detect suspicious activity in containers. |
| **Amazon Kube-state-metrics** (kube-state-metrics) | Exposes Kubernetes object metrics for Prometheus. |

**How to list available EKS add-ons (CLI)**

You can check AWS’s current add-on list for your region:

aws eks describe-addon-versions \

--region us-east-1 \

--query "addons[].addonName"

**Installing an Add-on**

Example: Install **Amazon EBS CSI Driver** as an EKS-managed add-on:

eksctl create addon \

--name aws-ebs-csi-driver \

--cluster my-eks-cluster \

--region us-east-1 \

--service-account-role-arn arn:aws:iam::<ACCOUNT\_ID>:role/AmazonEKS\_EBS\_CSI\_DriverRole

If you want, I can run through **the exact AWS CLI command** to list *all* available add-ons in your region so you see what’s available right now — the list changes over time.  
Do you want me to pull the live list for **us-east-1**?

8) Upgrade eks cluster

Upgrading an EKS cluster with eksctl is a **two-step** process:

1. **Upgrade the control plane** (the EKS cluster version).
2. **Upgrade your node groups** to match the new version.

**1️⃣ Check Current Cluster Version**

eksctl get cluster --region us-east-1

**2️⃣ Upgrade Control Plane**

Example: Upgrade from 1.29 to 1.30

eksctl upgrade cluster \

--name my-eks-cluster \

--region us-east-1 \

--version 1.30 \

--approve

* --approve skips the confirmation prompt.
* This upgrades **only the control plane**, not your worker nodes.

**3️⃣ Upgrade Node Groups**

List node groups:

eksctl get nodegroup --cluster my-eks-cluster --region us-east-1

Upgrade each node group:

eksctl upgrade nodegroup \

--cluster my-eks-cluster \

--name standard-workers \

--region us-east-1 \

--kubernetes-version 1.30 \

--approve

This launches new nodes with the updated AMI and gracefully drains old ones.

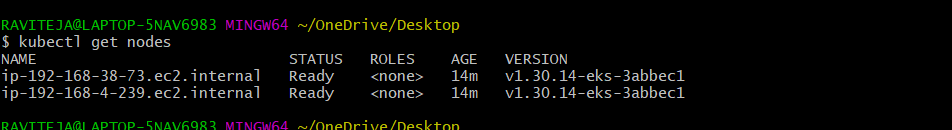
**4️⃣ Verify**

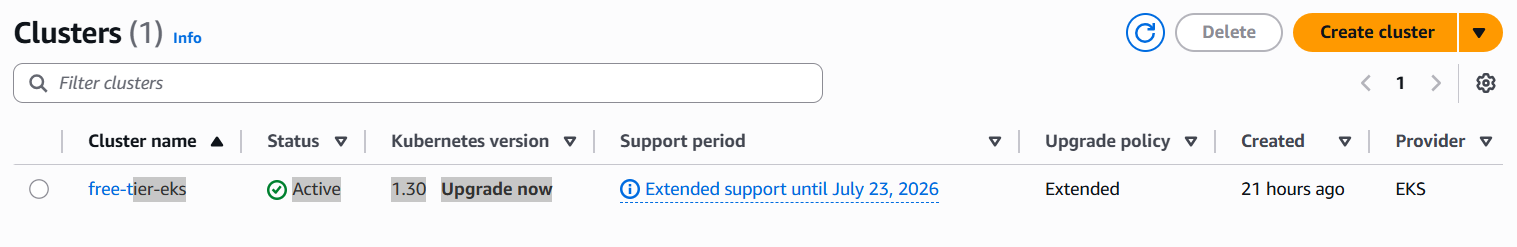
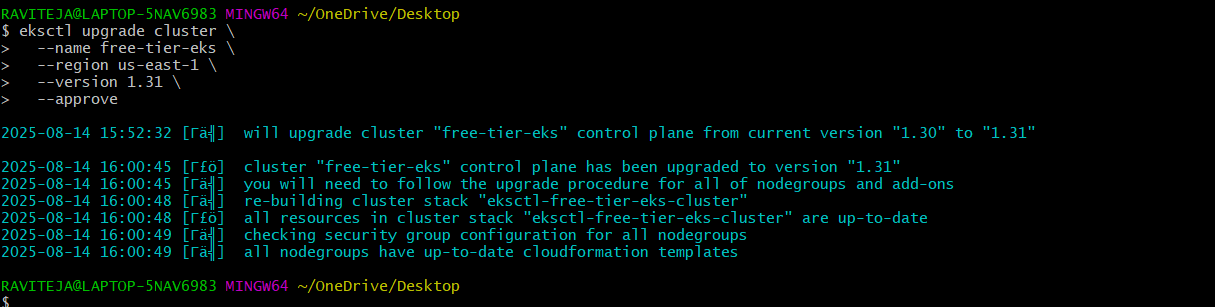
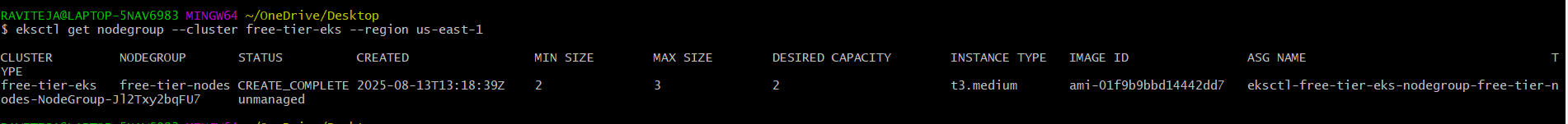
kubectl get nodes

Make sure all nodes show the new Kubernetes version.

💡 **Tip:** EKS only supports upgrades **one minor version at a time** (e.g., 1.28 → 1.29 → 1.30). You can’t skip versions.

Do you want me to also give you the **safe rolling upgrade method** so your workloads have zero downtime during the node group upgrade? That’s the part most people overlook.



eksctl upgrade nodegroup \

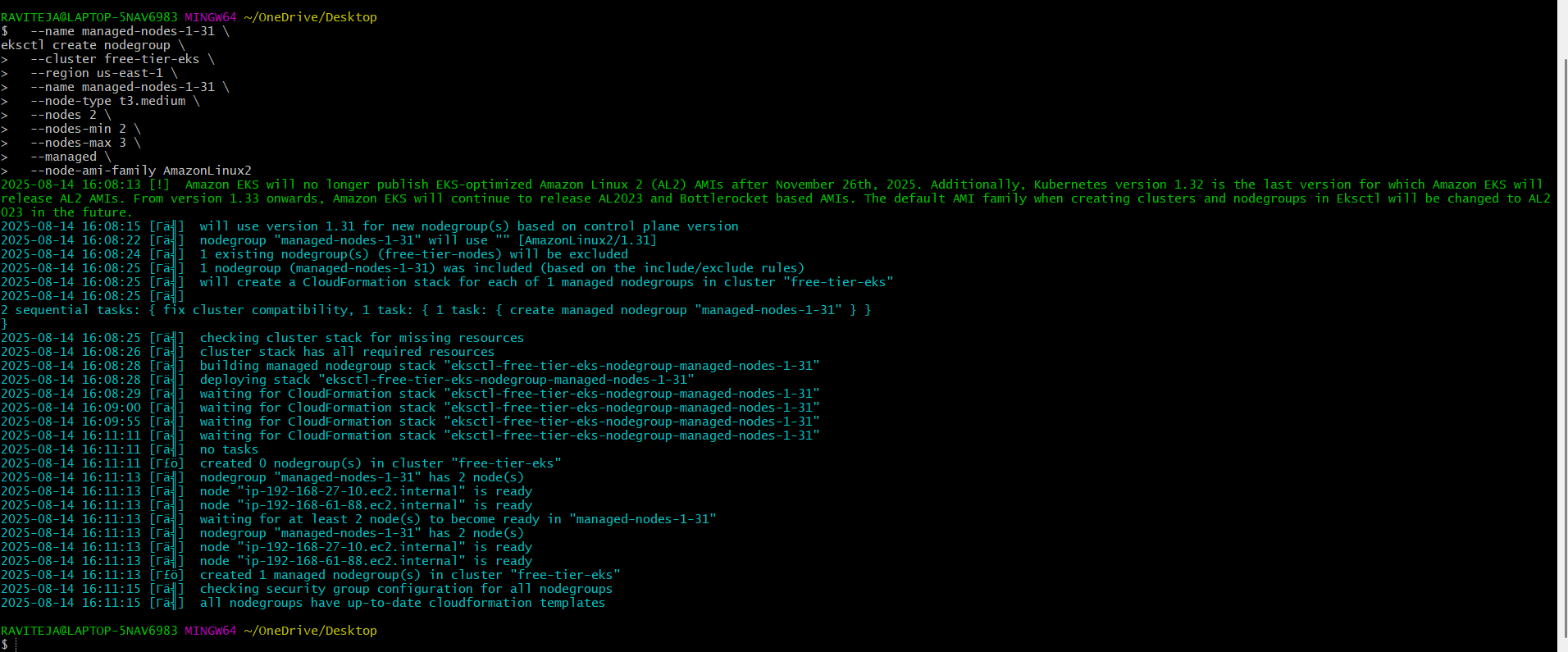
--cluster free-tier-eks \

--name free-tier-nodes \

--region us-east-1 \

--kubernetes-version 1.31 \

--approve



ip-192-168-25-167.ec2.internal

kubectl drain ip-192-168-25-167.ec2.internal --ignore-daemonsets --delete-emptydir-data

