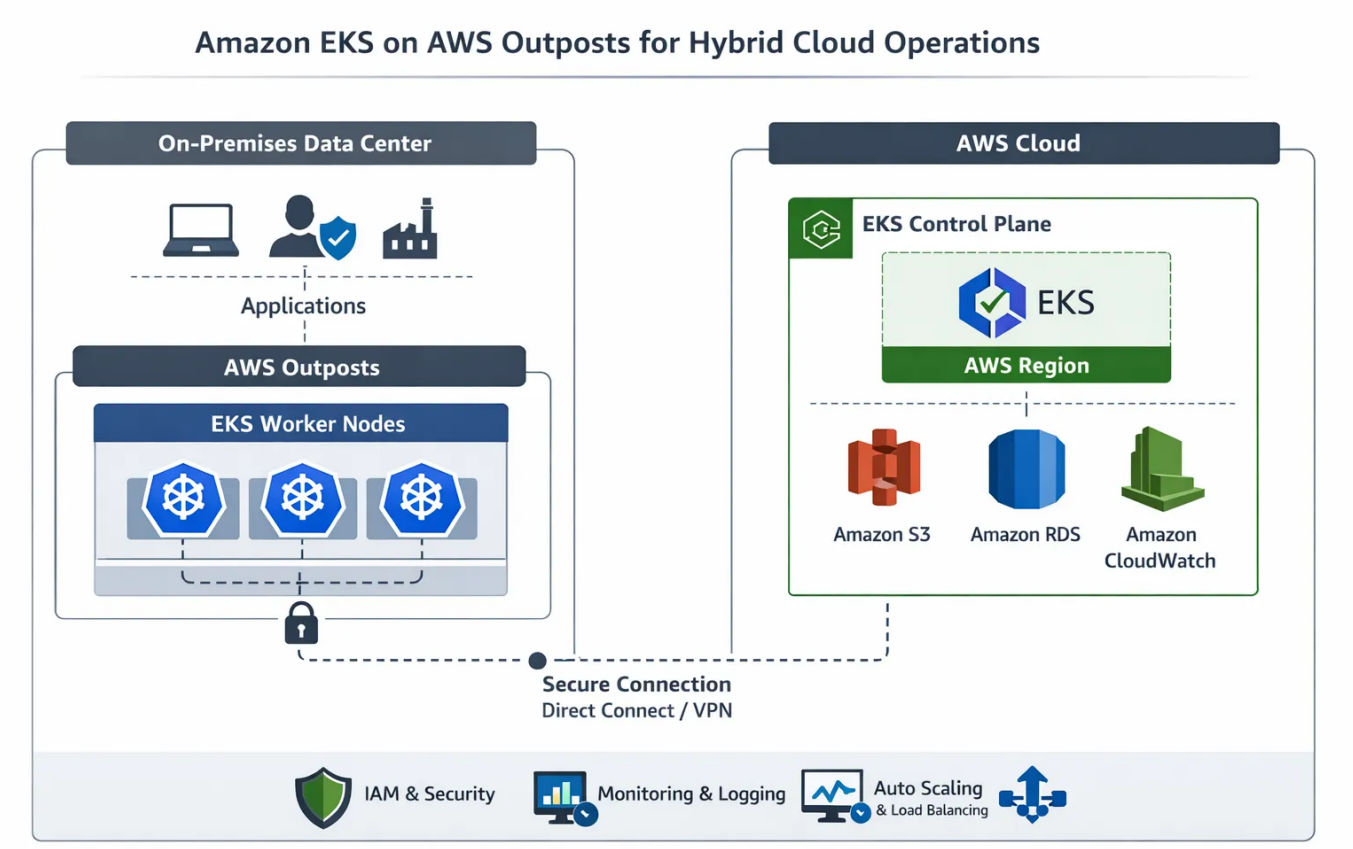
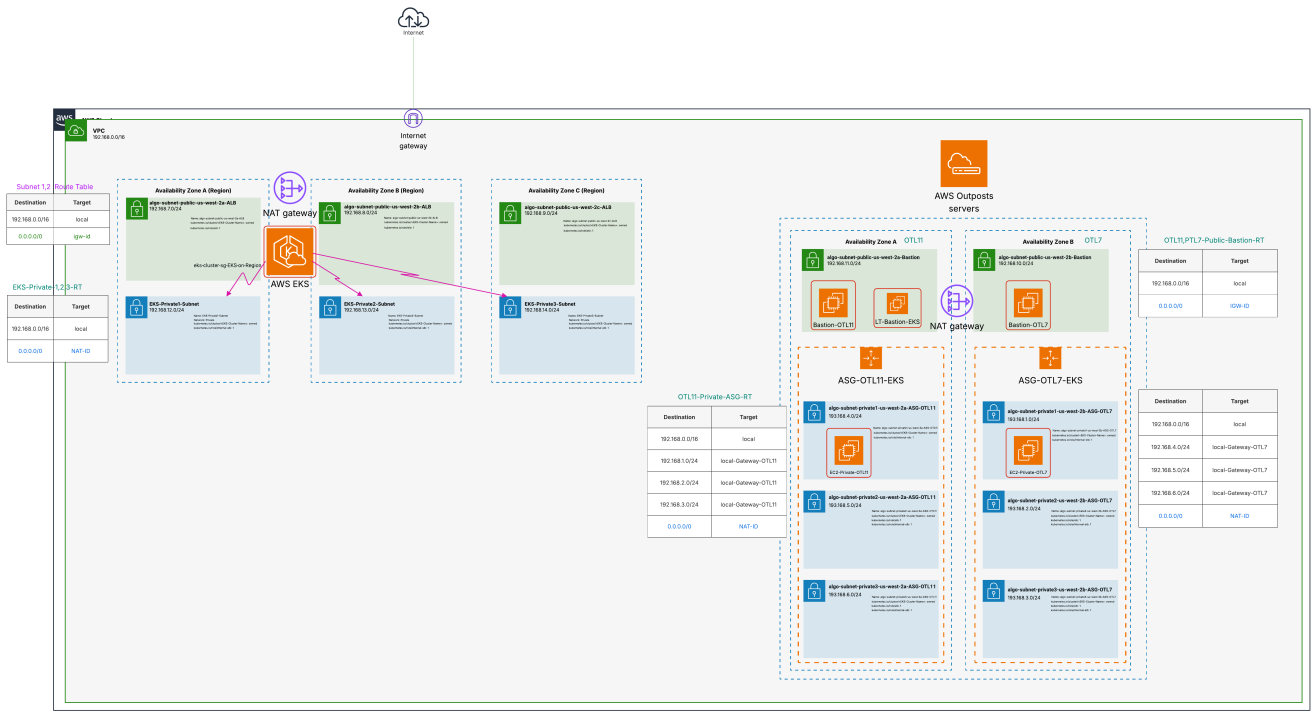
Amazon EKS (Self-managed)

Architecture





Amazon EKS (Control Plane) — Self-managed

1 Create IAM Roles

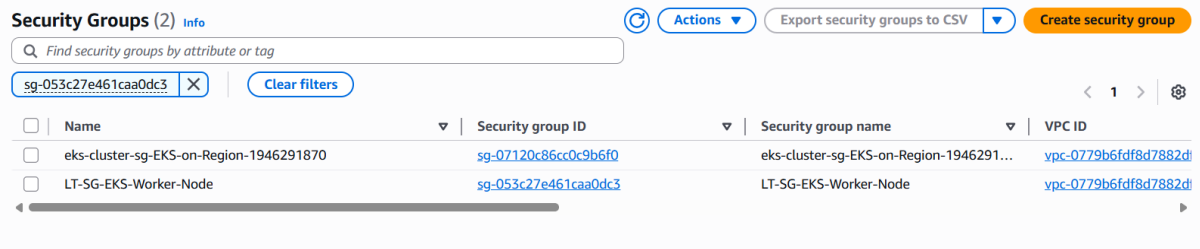
- **Cluster IAM Role** → Create → AWS Service → **Service: EKS** (**EKS - Cluster**) → **Add permissions (AmazonEKSClusterPolicy**)
- **Next**
  - **Role Name**: **EKSClusterRole**
  - **Description**: Allow the **K8s Cluster Control Plane** to manage AWS Resources on your behalf
- **Node Group** needs an **EC2 Instance IAM Role** to launch and register with a cluster.
  - **Node IAM Role** → AWS Service → **Service: EC2** (**EC2**) → **Add permissions (AmazonSSMManagedInstanceCore**, **AmazonEKSComputePolicy AmazonEC2ContainerRegistryReadOnly**, **AmazonEKS_CNI_Policy**, **AmazonEKSWorkerNodePolicy**)
  - **Role Name**: **EKSNodeRole**
  - **Description**: Allow **EC2 Instance** to call AWS services on your behalf.

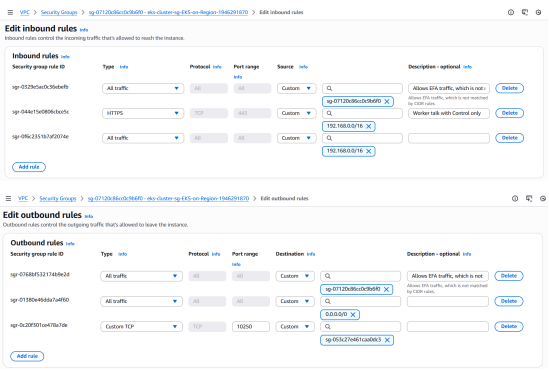
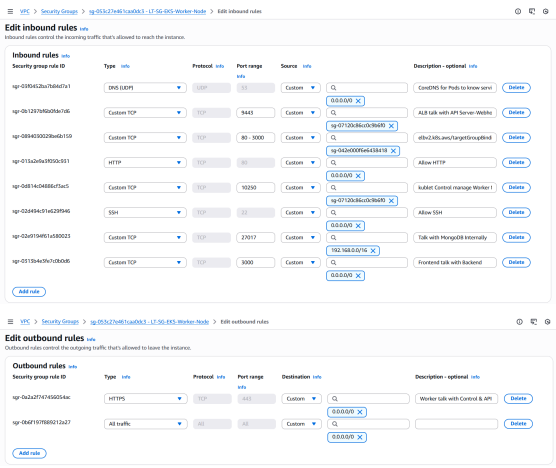
2 Create the EKS Cluster

- Select **Custom Configuration**
- ⃠ **Disable EKS Auto Mode**
  - ✔ You have now disabled AWS automatic **Node** management (**Self-Managed Nodes**)
- **Kubernetes control plane location: AWS Cloud**
- **Cluster name**: **EKS-on-Region**
- **Cluster IAM Role**: **EKSClusterRole**
- Keep the rest of the configurations default, and click on **next**
- **Networking**
  - Select <Your **VPC>**
  - Select **3 Private subnets (Have only one Route Table) & Created on Region & Tag them**
    - **Route them** to **NAT** (Allows the private subnet to access the internet)
    - **Sub-Name:** (Region)
      - **Tags for all Private Subnets of outpost**

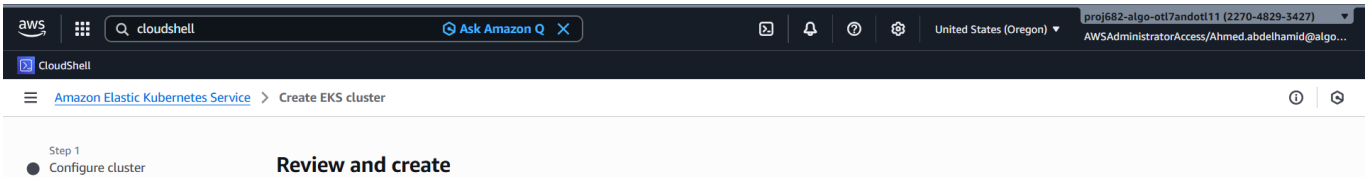
```
Network: Private
Name: <Your-Subnet-Name>
kubernetes.io/cluster/<EKS-Cluster-Name>: shared
# ALB works inside the VPC only (Private Subnet)
kubernetes.io/role/internal-elb: 1
```

- **Security Group (Skip it)**
    - **Security Group will be created by default — Control Plane**



Firewall	SG for Control Plane	SG for Worker Node
SG for Inbound & Outbound		

- **Cluster endpoint access → Private (Access Cluster from inside VPC only)**
- Keep the rest of the configurations default, and click on **next**
- Skip Observability changes and click on **next**
- **Add the following add-ons**:
    - **Amazon VPC CNI**
    - **kube-proxy**
    - **CoreDNS**
    - *Amazon EBS CSI Driver*
    - *Amazon EKS Pod Identity Agent*
    - *Metrics Server*
- **Next → Next**
- Review and create the cluster.



Step 2
Specify networking
Step 3
Configure observability
Step 4
Select add-ons
Step 5
Configure selected add-ons settings
Step 6
Review and create

Step 1: Cluster

Edit

Cluster configuration

Name
EKS-on-Region

Kubernetes version
1.34

EKS Auto Mode
Disabled

Upgrade policy
Standard support

Cluster IAM role
arn:aws:iam::227048293427:role/EKSClusterRole

Kubernetes cluster administrator access
Allow cluster administrator access

Authentication mode
EKS API

Control plane scaling tier
Standard

ARC Zonal shift

ARC Zonal shift
Disabled

Deletion protection

Deletion protection
Disabled

Tags (0)

Tags that you've added. Each tag consists of a key and an optional value.

1

Key	Value

No tags
This cluster does not have any tags.

Step 2: Networking

Edit

Networking

These properties cannot be changed after the cluster is created.

VPC
vpc-0779b6fdf8d7882df

Subnets
subnet-007120692f1211954
subnet-034f8bb5247adf32c
subnet-02d4bcca4d489638c

Cluster IP address family
IPv4

Cluster endpoint access

API server endpoint access
Private

Step 3: Observability

Edit

Container network observability Info

Network monitoring status
Disabled

Included capabilities
Service map | Flow table | Performance metric endpoint

Network flow monitor configuration

Network flow monitor name
eks-EKS-on-Region-flow-monitor

Local resources
EKS cluster

Remote resources
Everywhere in us-west-2

Control plane logs

API server
on

Authenticator
on

Scheduler
on

Audit
on

Controller manager
on

Step 4: Add-ons

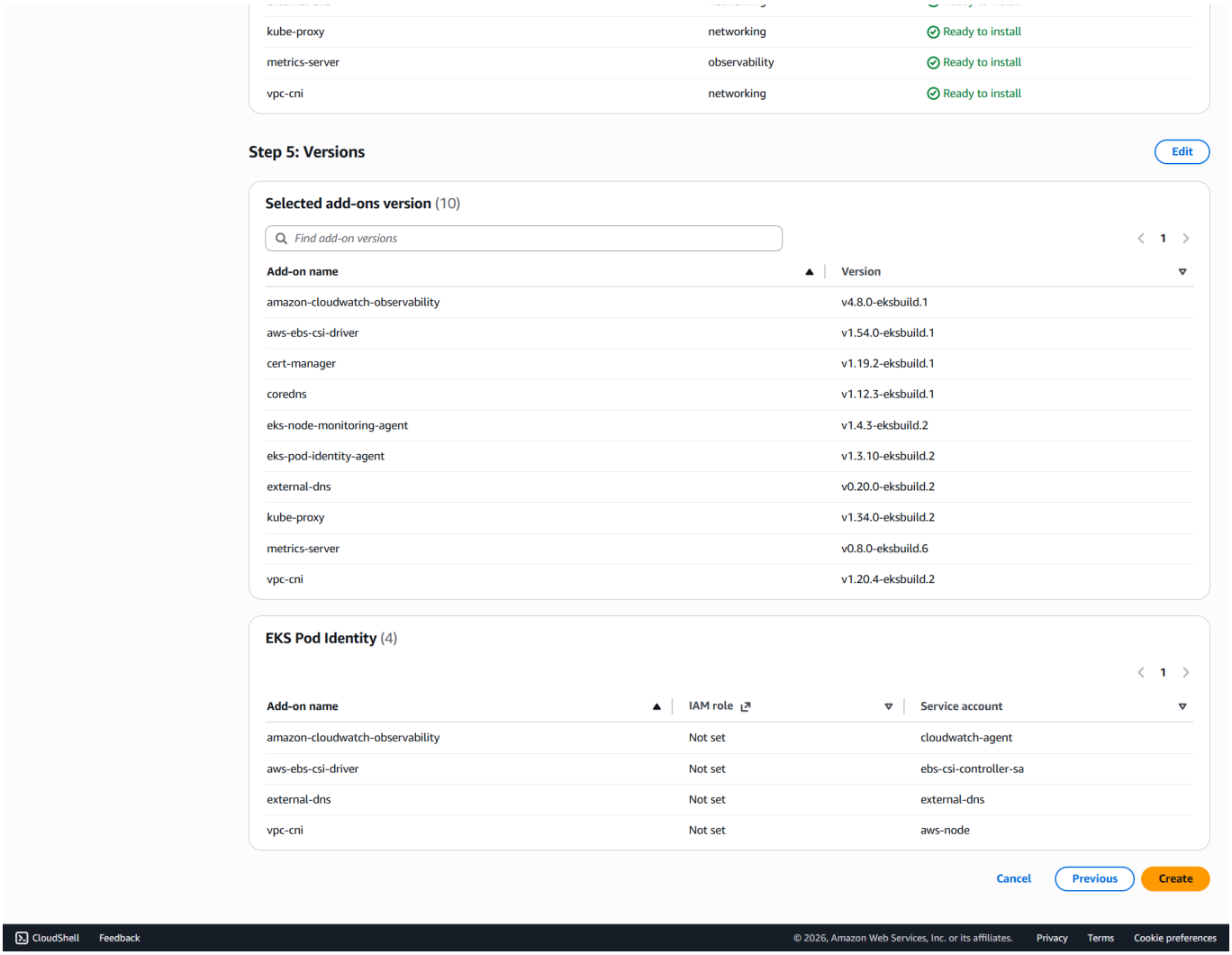
Edit

Selected add-ons (10)

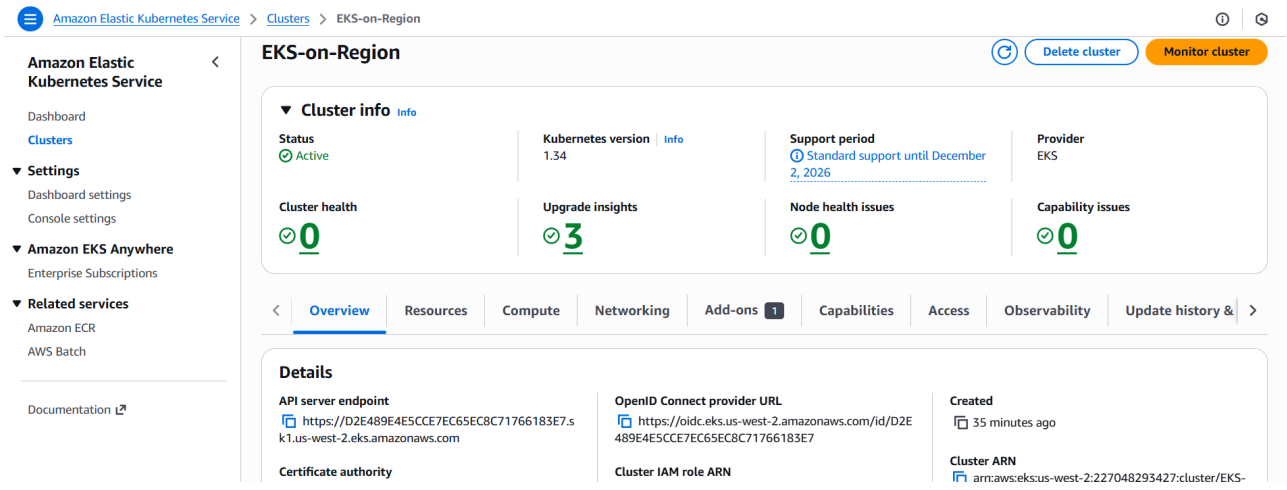
Find add-on

1

Add-on name	Type	Status
amazon-cloudwatch-observability	observability	Ready to install
aws-ebs-csi-driver	storage	Ready to install
cert-manager	security	Ready to install
coredns	networking	Ready to install
eks-node-monitoring-agent	observability	Ready to install
eks-pod-identity-agent	security	Ready to install
external-dns	networking	Ready to install



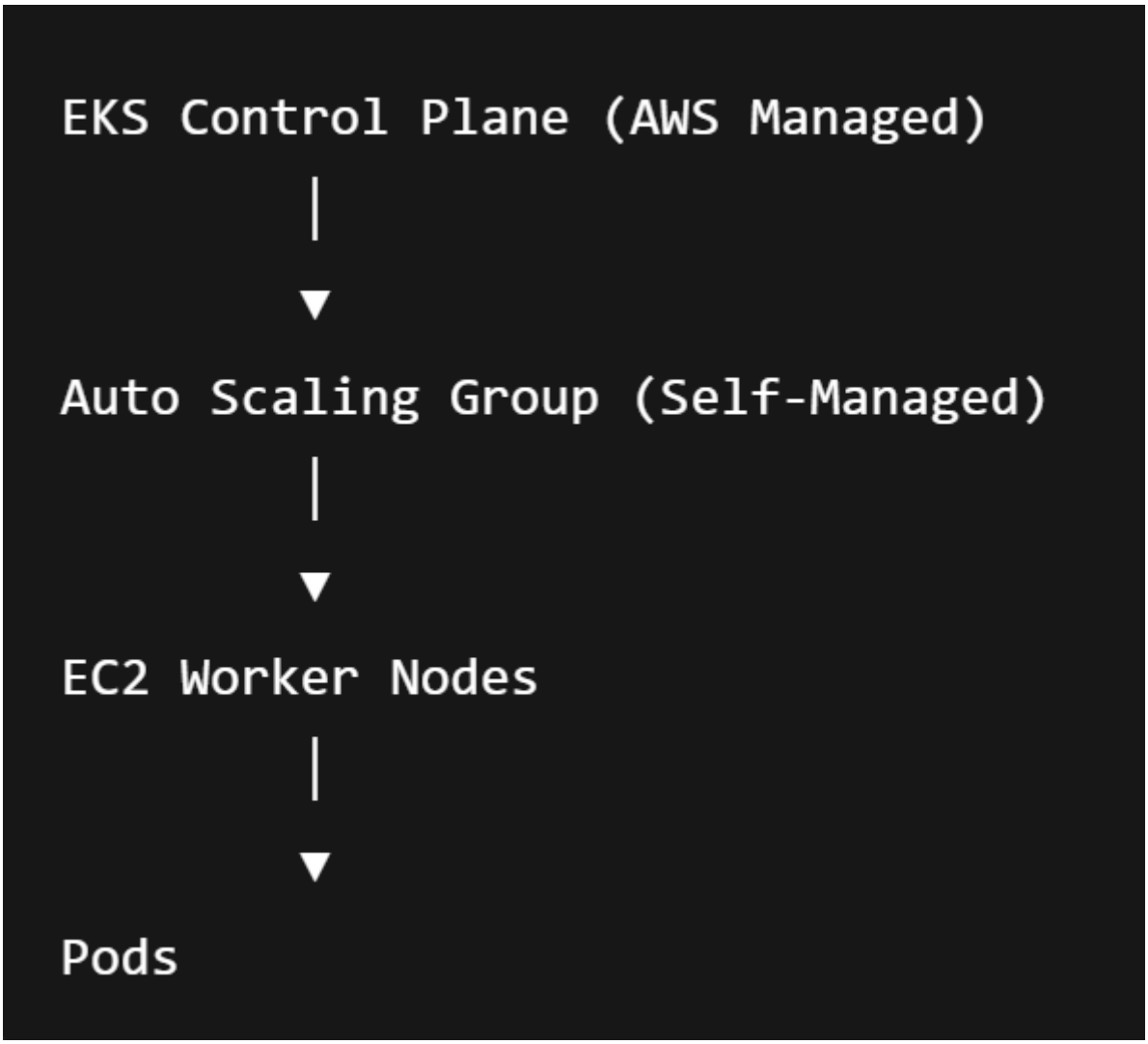
- **Wait 8–10 minutes for provisioning**



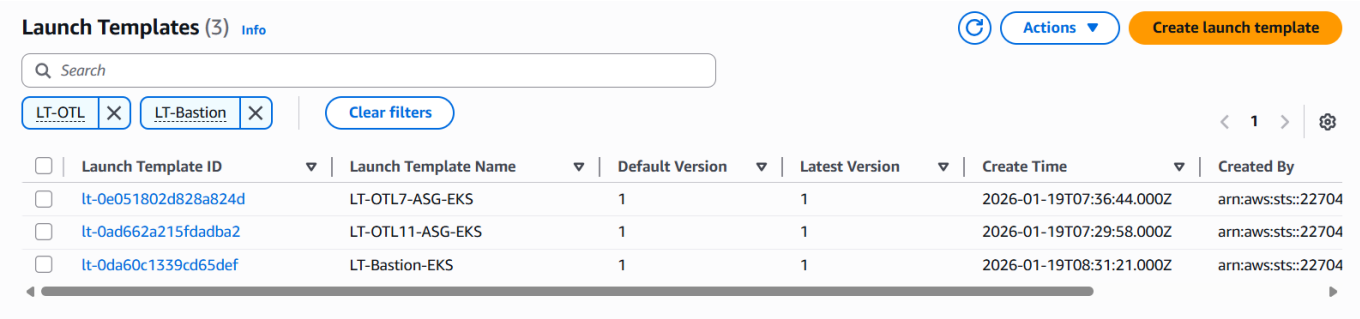
3 Create a Node Group

- **EKS Cluster** is working (**without Node Group**)
- Managed **Nodegroups** are not supported on **Outposts**

🧠 Self-Managed Worker Nodes



📌 Create 3 Launch Templates



- **Amazon Machine Image (AMI):**
  - Before choosing **AMI-xx**, ensure that the **Amazon-eks-node**
  - From **Community AMIs (500): Search for EKS Optimized**
    - Choose the right one
  - After choosing the right **ami**, check that `kubelet` installed through `sudo systemctl status kubelet` via `SSM`
- **Instance type — OTL11 m5.xlarge**
- **Instance type — OTL7 c5.xlarge**

- **Instance type — m5.xlarge (Bastion)**
- **Key pair name:**
- **Networking:**
    - **Subnet: Don't include in launch template**
    - **Availability Zone: Don't include in launch template**
    - **Security Group:**
        - **Name: LT-SG-EKS-Worker-Node**
        - **Security Group:** Add → **SGs — Worker Node**
- **EBS Volume: 20GB, gp2**
- **Advanced details**
    - **IAM instance profile**
        - **Add Role for Launch Template**
        - **Role Name: EKSNodeRole**
        - **Create a Role for Bastion**
            - **Use Case: EC2** (EC2)
            - **Add permissions**
            - **Role Name: abdelhamid-SSM-Role**
            - Attach it to **EC2**
        - **Connect with EC2**
            - **AWS Systems Manager** → Session Manager → Start session
    - **Metadata response hop limit: 2**
    - **User Data for (Launch Template) — NodeConfig**

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==BOUNDARY=="

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
# If any command fails → the script stops immediately ❌

--==BOUNDARY==
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
```

```
spec:
  cluster:
    name: <EKS-Cluster-Name>
    apiServerEndpoint: <EKS-endpoint>
    certificateAuthority: <EKS-certificate-Authority>
    # ➜ The **IP range** from which the **Pods** will take an IP address
    cidr: "10.100.0.0/16"
  kubelet:
    config:
      maxPods: 110
  node:
    labels:
      app: my-app
      # Kubernetes will not be able to differentiate between OTL1 and OTL7
      # outpost: otl7
      # outpost: otl11
      eks.amazonaws.com/capacityType: ON_DEMAND

--==BOUNDARY==--
```

- **User Data for (BastionTemplate)**

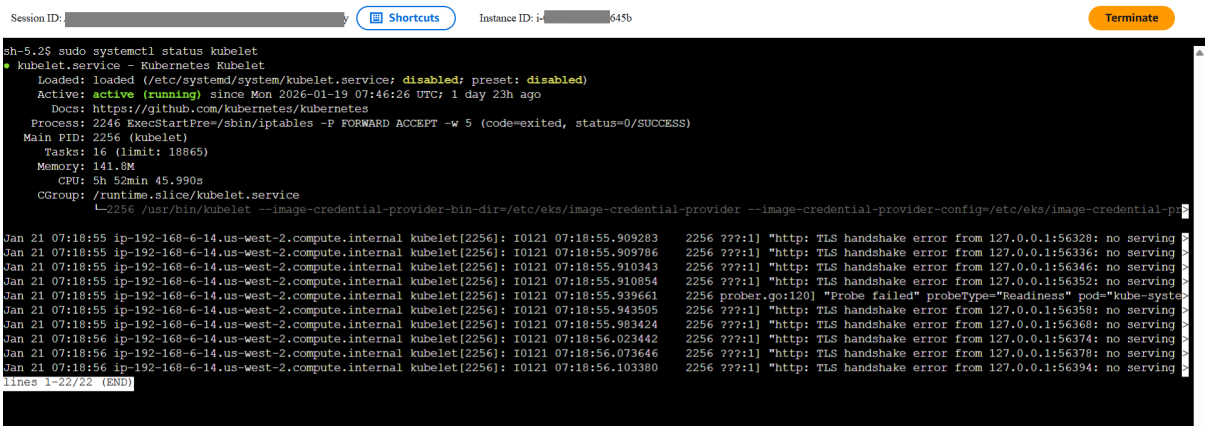
```
#!/bin/bash
yum update -y
sudo yum install -y awscli
# Install kubectl
curl -LO https://dl.k8s.io/release/$(curl -Ls
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
kubectl version --client
```

📌 Create 2 Auto Scaling Groups

We will create 2 Auto Scaling Groups; AWS does not allow a single instance to appear in more than one Outpost.

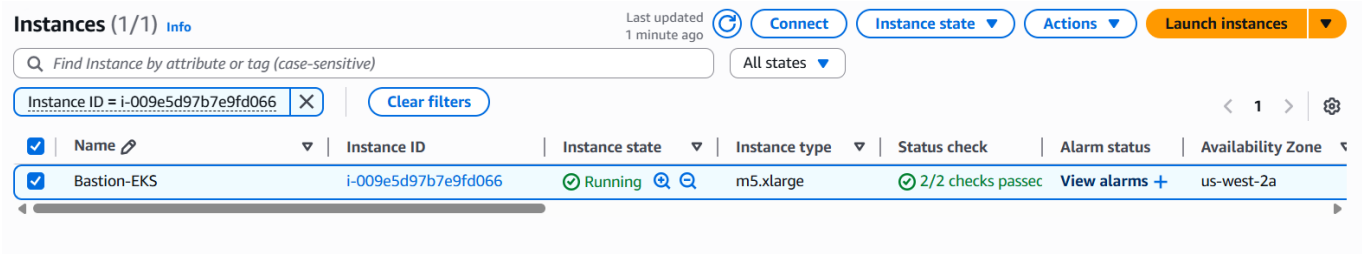
- **Name: ASG-OTL11-EKS**
- **Launch template: LT-OTL11-ASG-EKS**
- **Name: ASG-OTL7-EKS**
- **Launch template: LT-OTL7-ASG-EKS**
- **VPC: 'Your-VPC'**
- **Availability Zones and subnets:**
    - **3 Private Subnets (OTL11)**

    - **3 Private Subnets (OTL7)**
- **Availability Zone distribution: Balanced best effort**
- **Load balancing: No load balancer**
- **VPC Lattice integration options: No VPC Lattice service**
- **Group size**
    - **Desired capacity: 3**
    - **Min desired capacity: 1**
    - **Max desired capacity: 7**
- **Automatic scaling - *optional: No scaling policies***
- **Instance maintenance policy: No Policy**
- **Next**
- After creating the **Auto Scaling Group**, → Check that `kubelet` is installed on **Worker Node**
    - Go to **AWS System Manager** (**SSM**) → Choose the `right ami` → Start Session → `sudo systemctl status kubelet`



🔍 Testing (Bastion)

Create an EC2 Instance from Launch Template (LT-Bastion-EKS)



Pre-Test Setup & Validation

Before Testing – EKS Access	Bastion Access & Cluster Validation
Go to **EKS (Access) → IAM access entries → Create**	**SSH** to Bastion host
IAM principal ARN: `EKSNodeRole (Worker Node Role)`	`aws eks update-kubeconfig --region <Region> --name <EKS-Name>`
Policy name: `AmazonEKSClusterAdminPolicy`	`kubectl get ns`
Click **Add policy**	`kubectl get nodes`