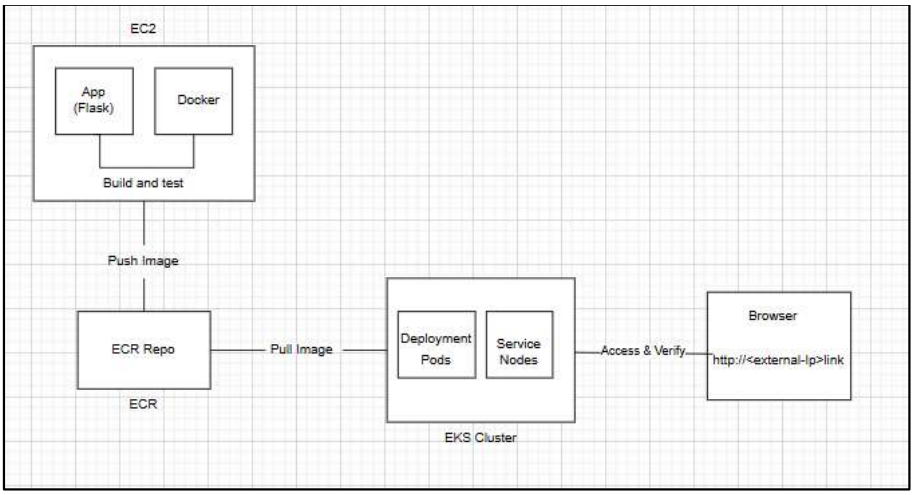
Task : 18

Kubernetes , ECR and Docker concept

Objective

Deployed a Python Flask application on AWS EKS using Kubernetes Deployments and Services, enabling external access via LoadBalancer while managing containerized workloads in a scalable cloud environment.

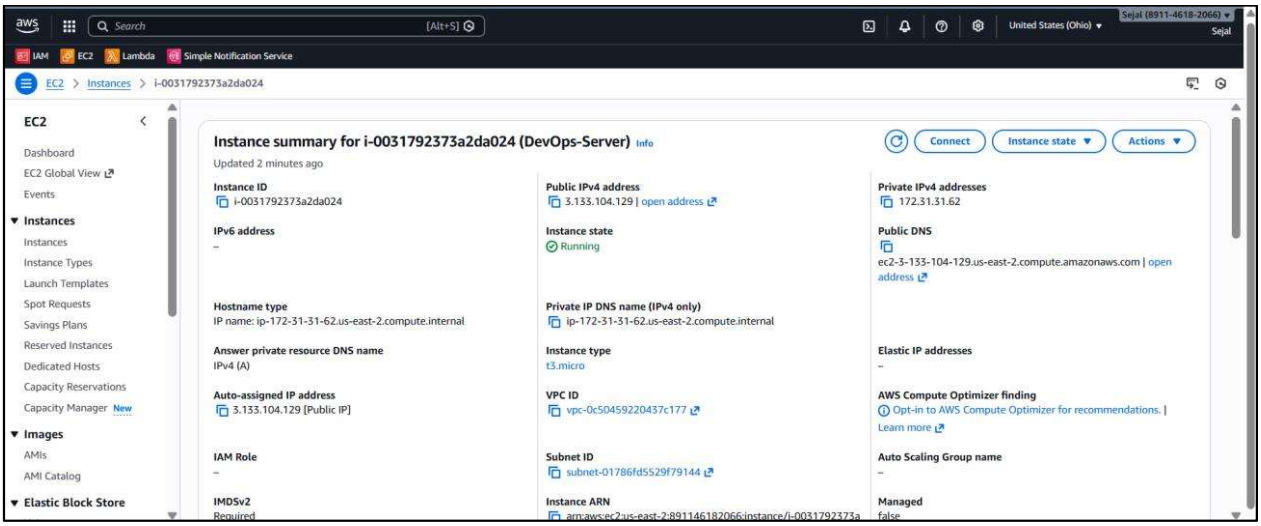
Flow Diagram :

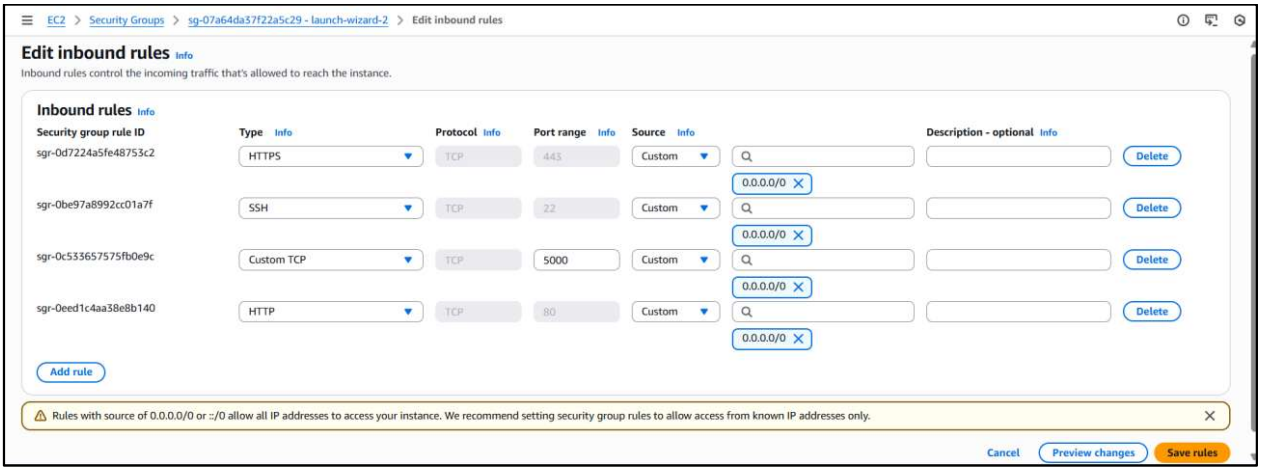


Step 1 : Create Ec2 Instance.

- Name : DevOps-Server
- Security group :
  - Allow ports:

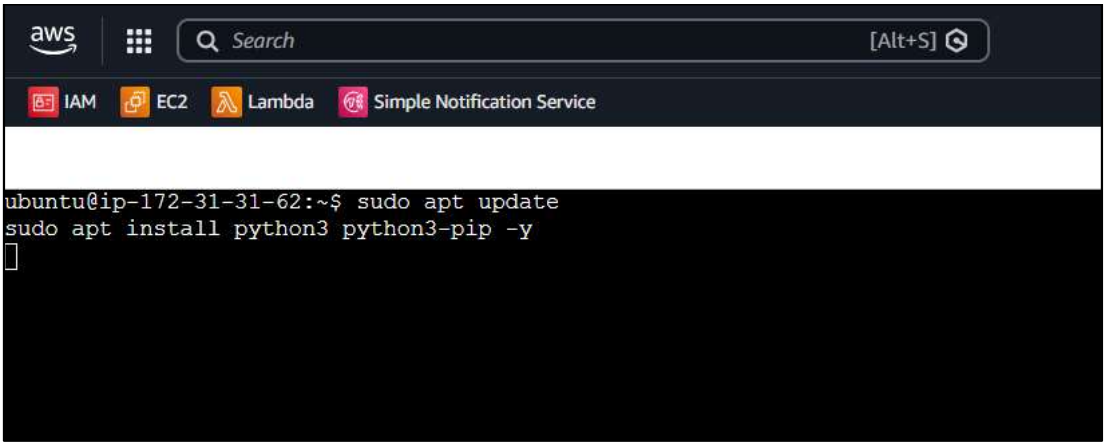
Port	Reason
22	SSH
5000	Flask
80	HTTP
30000-32767	Kubernetes NodePort





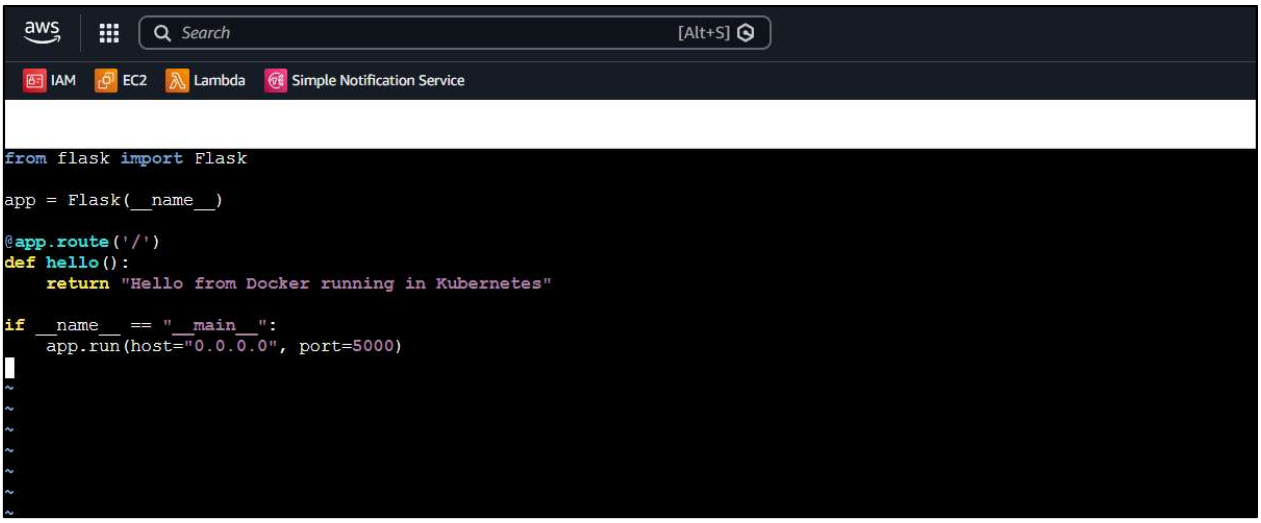
Step 2 : Create and deploy application on Server

- Install Python & tools
  - sudo apt update
  - sudo apt install python3 python3-pip -y



- Create project folder:
  - mkdir flask-app
  - cd flask-app

- Create app file
  - Vim app.py



- Create Virtual environment and active it
  - Python –m venv venv
  - Source venv/bin/activate

```
ubuntu@ip-172-31-31-62:~/flask-app$
ubuntu@ip-172-31-31-62:~/flask-app$ source venv/bin/activate
(venv) ubuntu@ip-172-31-31-62:~/flask-app$ ls
app.py  venv
(venv) ubuntu@ip-172-31-31-62:~/flask-app$
```

- Install Flask:
  - pip3 install flask

```
(venv) ubuntu@ip-172-31-31-62:~/flask-app$ pip install flask
Collecting flask
  Downloading flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
Collecting blinker>=1.9.0 (from flask)
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.3.1-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.2.0 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting markupsafe>=2.1.1 (from flask)
  Downloading markupsafe-3.0.3-cp312-cp312-manylinux2014 x86 64.manylinux 2 17 x86 64.manylinux 2 28 x86 64.whl.metadata (2.7 kB)
```

- Run application
  - Python app.py

```
^C(venv) ubuntu@ip-172-31-31-62:~/flask-app$ sudo vim app.py
(venv) ubuntu@ip-172-31-31-62:~/flask-app$ python app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.31.62:5000
Press CTRL+C to quit
```

- Check application
  - http://<public_ip>:5000


Not secure 3.133.104.129:5000
Hello from Docker running in Kubernetes

Step 3 : Docker Image Creation

- Install Docker
  - sudo apt install docker.io -y
  - sudo systemctl start docker
  - sudo usermod -aG docker ubuntu newgrp docker

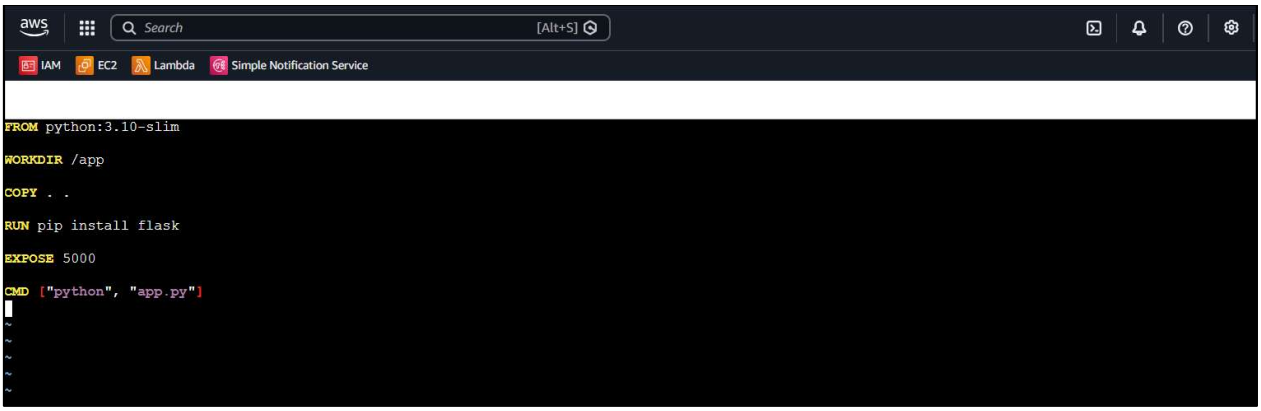
```
^C(venv) ubuntu@ip-172-31-31-62:~/flask-app$ sudo apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 4 not upgraded.
Need to get 76.1 MB of archives.
After this operation, 288 MB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.3.3-0ubuntu1~24.04.3 [8815 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.28-0ubuntu1~24.04.2 [38.4 MB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dns-root-data all 2024071801~ubuntu0.24.04.1 [5918 B]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dnsmasq-base amd64 2.90-2ubuntu0.1 [376 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 28.2.2-0ubuntu1~24.04.1 [28.3 MB]
Get:8 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 ubuntu-fan all 0.12.16+24.04.1 [34.2 kB]
Fetched 76.1 MB in 1s (89.3 MB/s)
```

```
(venv) ubuntu@ip-172-31-31-62:~/flask-app$ sudo systemctl start docker
(venv) ubuntu@ip-172-31-31-62:~/flask-app$
```

```
(venv) ubuntu@ip-172-31-31-62:~/flask-app$ sudo usermod -aG docker ubuntu
newgrp docker
ubuntu@ip-172-31-31-62:~/flask-app$
```

- Create Dockerfile
    - Inside flask-app folder: nano Dockerfile

```
ubuntu@ip-172-31-31-62:~/flask-app$ sudo vim Dockerfile
ubuntu@ip-172-31-31-62:~/flask-app$
```




```
ubuntu@ip-172-31-31-62:~/flask-app$ ls
Dockerfile  app.py  venv
ubuntu@ip-172-31-31-62:~/flask-app$
```

- Build Docker Image
    - docker build -t flask-k8s-app .

```
ubuntu@ip-172-31-31-62:~/flask-app$ docker build -t flask-k8s-app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  19.02MB
Step 1/6 : FROM python:3.10-slim
3.10-slim: Pulling from library/python
0c8d55a45c0d: Pulling fs layer
b9561eba91ba: Pulling fs layer
0cca1a604ed2: Pulling fs layer
d00020a00819: Pulling fs layer
d00020a00819: Waiting
b9561eba91ba: Verifying Checksum
b9561eba91ba: Download complete
d00020a00819: Verifying Checksum
```

- Check image:
    - docker images

```
ubuntu@ip-172-31-31-62:~/flask-app$ docker images
REPOSITORY      TAG         IMAGE ID       CREATED          SIZE
flask-k8s-app   latest      e2c6f5ef37d6   37 seconds ago   152MB
python          3.10-slim   6973bbbb27d3   31 hours ago     122MB
ubuntu@ip-172-31-31-62:~/flask-app$
```

- Run container locally
  - docker run -d -p 5000:5000 flask-k8s-app

```
ubuntu@ip-172-31-31-62:~/flask-app$ docker run -d -p 5000:5000 flask-k8s-app
aee3a431e998fd176b02a48e9db8c4c901f5759d0575167136e4105254d101a0
ubuntu@ip-172-31-31-62:~/flask-app$
```

- Check application on browser.
  - http://<public_ip>:5000

- stop container
  - docker ps
  - docker stop CONTAINER_ID

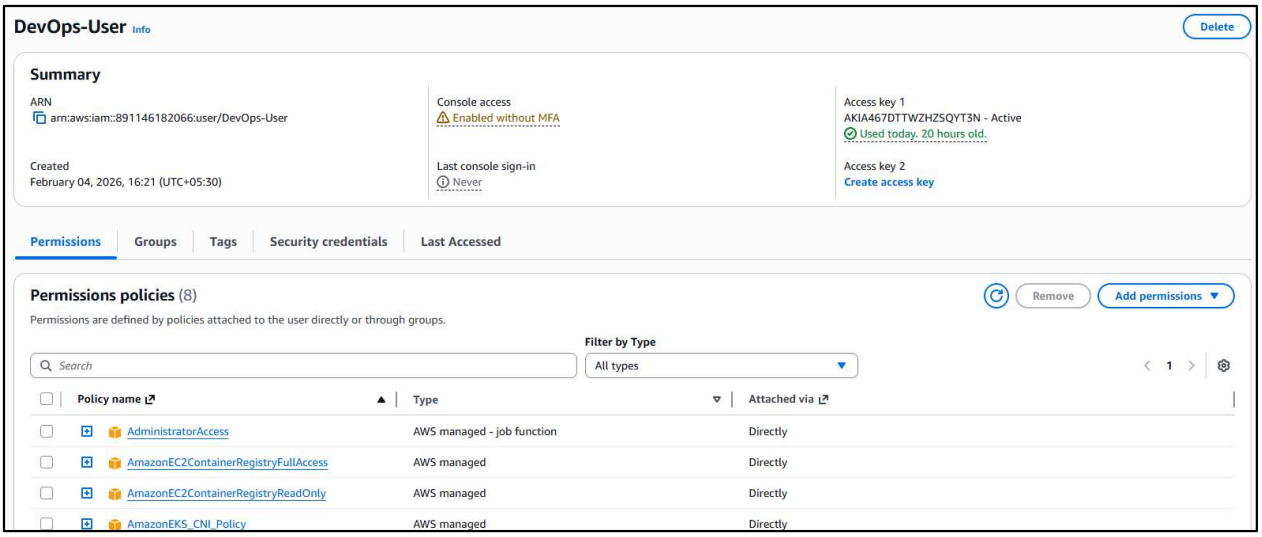
```
ubuntu@ip-172-31-31-62:~/flask-app$ docker ps
CONTAINER ID   IMAGE          COMMAND           CREATED              STATUS              PORTS                                       NAMES
aee3a431e998   flask-k8s-app  "python app.py"   About a minute ago   Up About a minute   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   festive_galois
ubuntu@ip-172-31-31-62:~/flask-app$
```

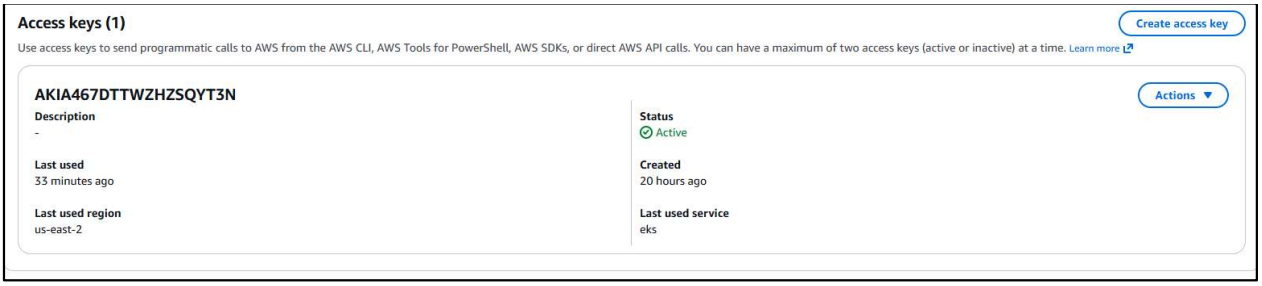
```
ubuntu@ip-172-31-31-62:~/flask-app$ docker stop aee3a431e998
aee3a431e998
```

```
ubuntu@ip-172-31-31-62:~/flask-app$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-31-62:~/flask-app$
```

Step 4 : Create IAM User

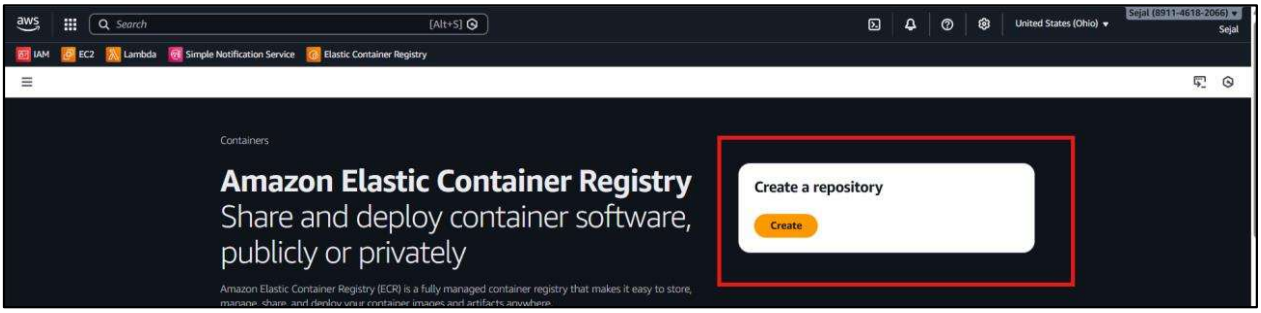
- Left menu → click **Users**
- Click **Create user**
    - User name: `DevOps-User`
- Attach Permissions



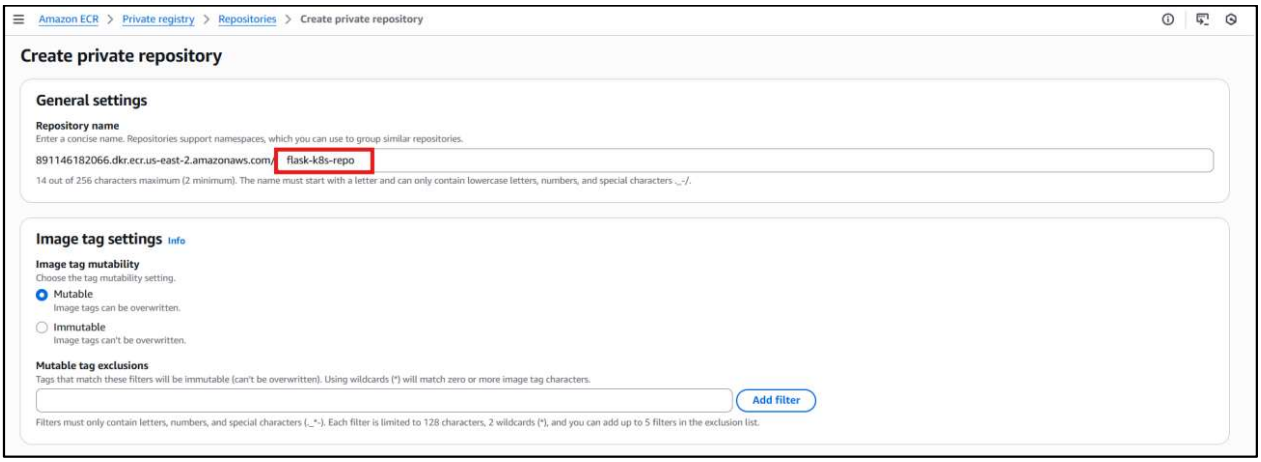


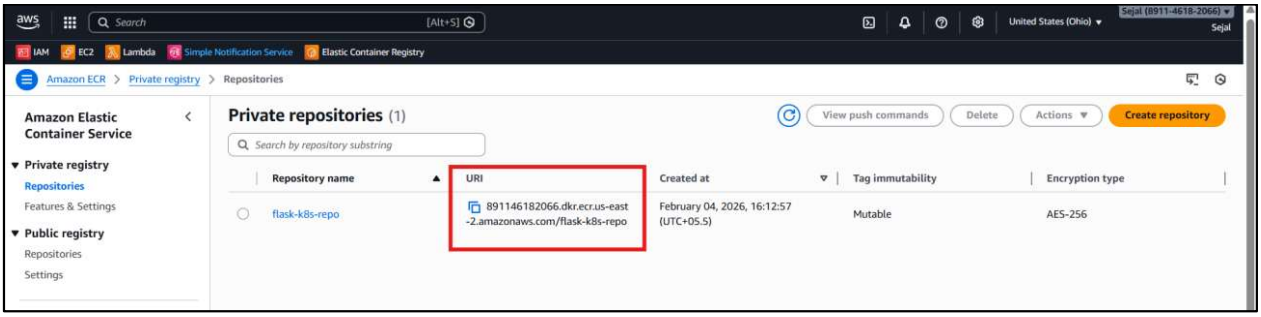
Step 5 : Amazon ECR Repository

- Go AWS Console → ECR



- Repository
- Name: flask-k8s-repo





- Install AWS CLI on EC2
  - sudo apt install awscli -y

```
ubuntu@ip-172-31-31-62:~$ aws --version
aws-cli/2.33.14 Python/3.13.11 Linux/6.14.0-1018-aws exe/x86_64.ubuntu.24
```

- Configure:
    - aws configure
- Enter:
    - Access key
    - Secret key
    - Region (example: ap-south-1)
    - output: json

```
ubuntu@ip-172-31-31-62:~$ aws configure
AWS Access Key ID [None]: AKIA467DTTWZHZSQYT3N
AWS Secret Access Key [None]: R9lGTIGyuMZmMY9YJ5Xdw1xrflKd7eEJjcNcTpfA
Default region name [None]:
Default output format [None]:
ubuntu@ip-172-31-31-62:~$
```

- Login Docker to ECR
    - aws ecr get-login-password --region <region_name> | docker login --username AWS --password-stdin <ecr_url>

```
ubuntu@ip-172-31-31-62:~$ aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 891146182066.dkr.ecr.us-east-2.amazonaws.com/flask-k8s-repo

WARNING! Your credentials are stored unencrypted in '/home/ubuntu/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
ubuntu@ip-172-31-31-62:~$
```

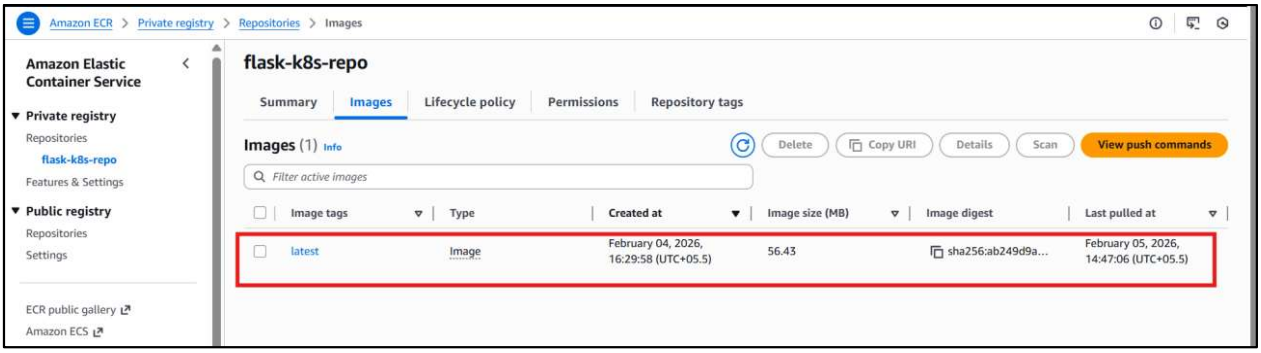
- Tag Image
    - docker tag flask-k8s-app:latest <ecr_url>:latest

```
ubuntu@ip-172-31-31-62:~$ docker tag flask-k8s-app:latest 891146182066.dkr.ecr.us-east-2.amazonaws.com/flask-k8s-repo:latest
ubuntu@ip-172-31-31-62:~$
```

- Push to ECR
    - docker push <ecr_url>:latest

```
ubuntu@ip-172-31-31-62:~$ docker push 891146182066.dkr.ecr.us-east-2.amazonaws.com/flask-k8s-repo:latest
The push refers to repository [891146182066.dkr.ecr.us-east-2.amazonaws.com/flask-k8s-repo]
5fd6f784a1a5: Pushing [==================================================>]  12.64MB
67cb3830c39b: Pushing [==================================================>]  19.02MB
c3a2e1ac4bc0: Pushed
73fac8bb9d80: Pushed
d394714285eb: Pushing [=====================>                             ]  17.43MB/39.44MB
47593b3f7136: Pushed
a8ff6f8cbdfd: Pushing [=============>                                     ]  20.69MB/78.62MB
```

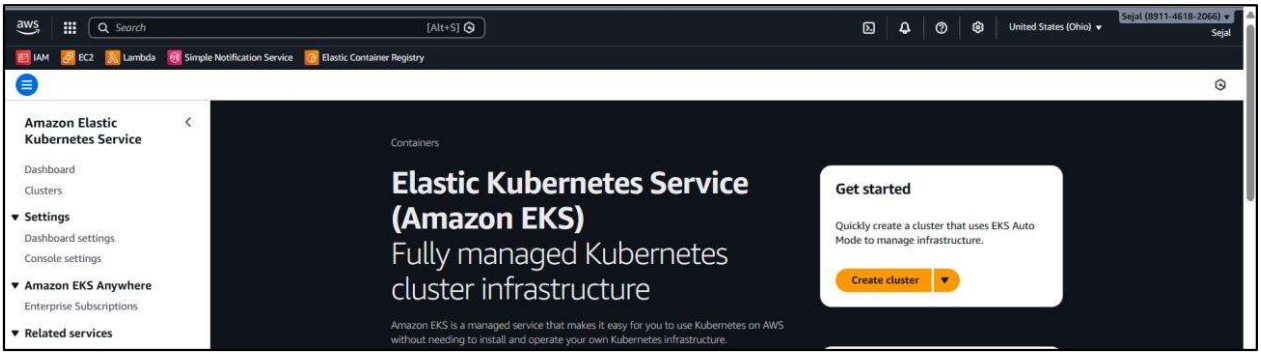
- Image is successfully upload in RCR

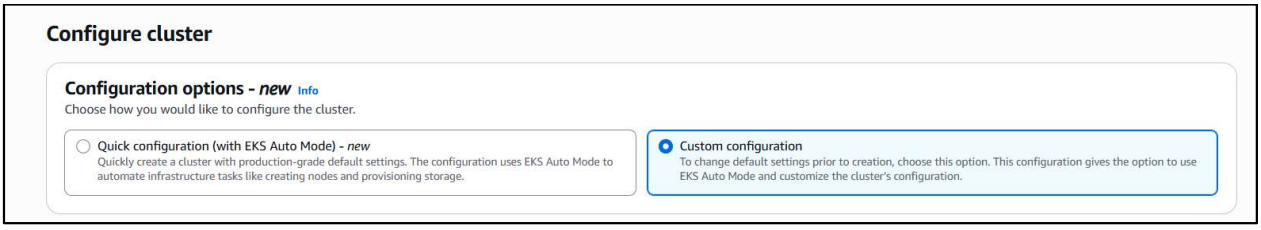


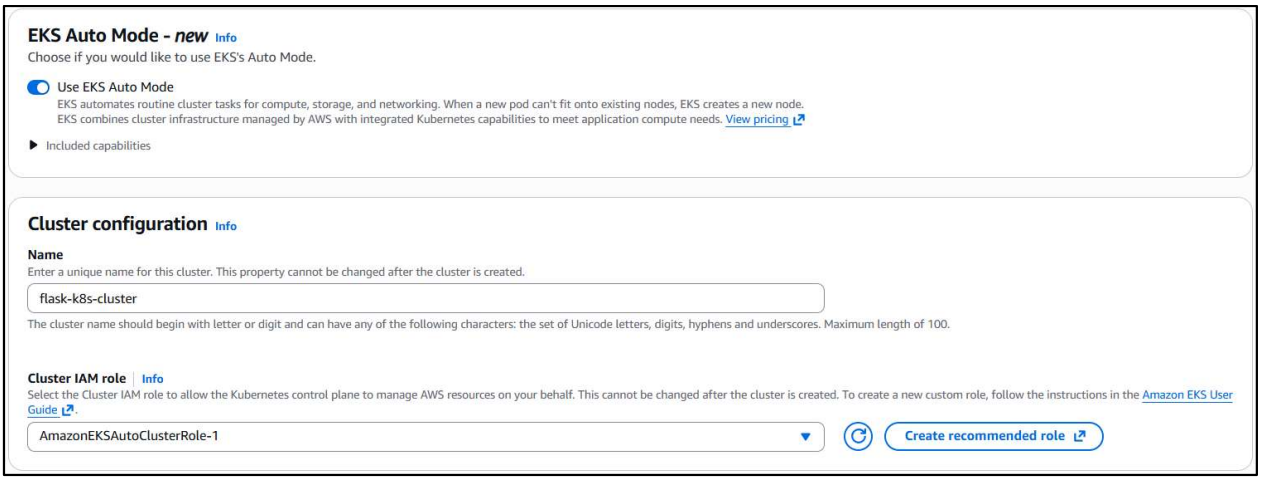
Step 6 : Create EKS Cluster

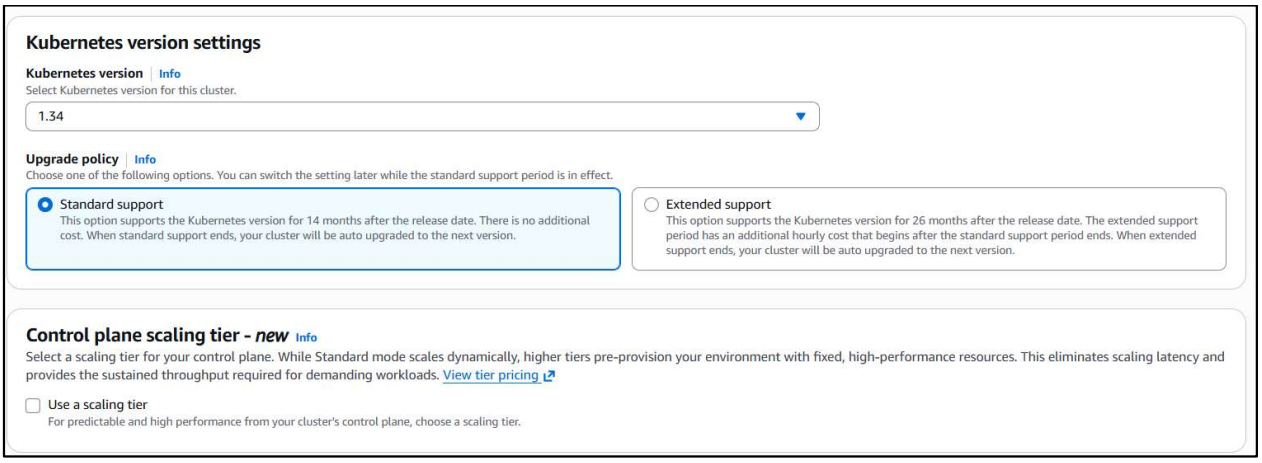
Create a New Cluster.

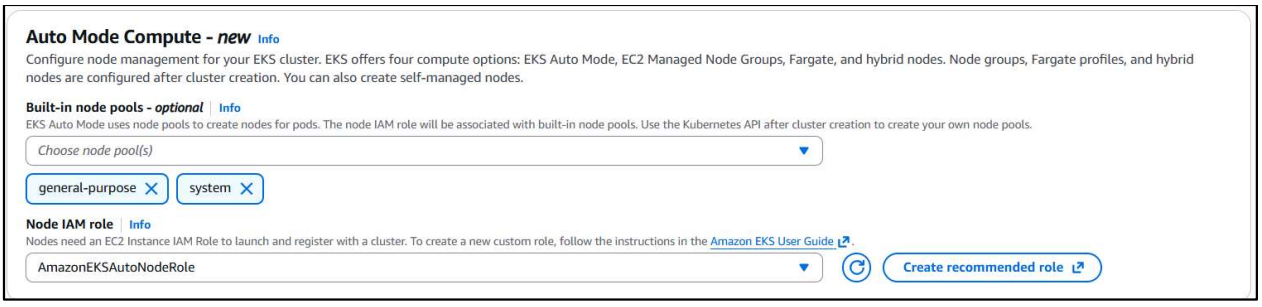
- Name : flask-k8s-cluster
- Kubernetes Version : 1.34
- Upgrade Policy : Standard Policy
- Attach role : Recommended
- Allow cluster administrator access
- Cluster authentication mode : EKS API
- ARC Zonal shift : Disable
- Select VPC and Subnets
- Attach security group
- cluster endpoint access : Public
- enable CloudWatch, Network monitoring,
- Control plane logs : Check all
- select add ons : Metrics Server, Cube proxy, CoreDNS

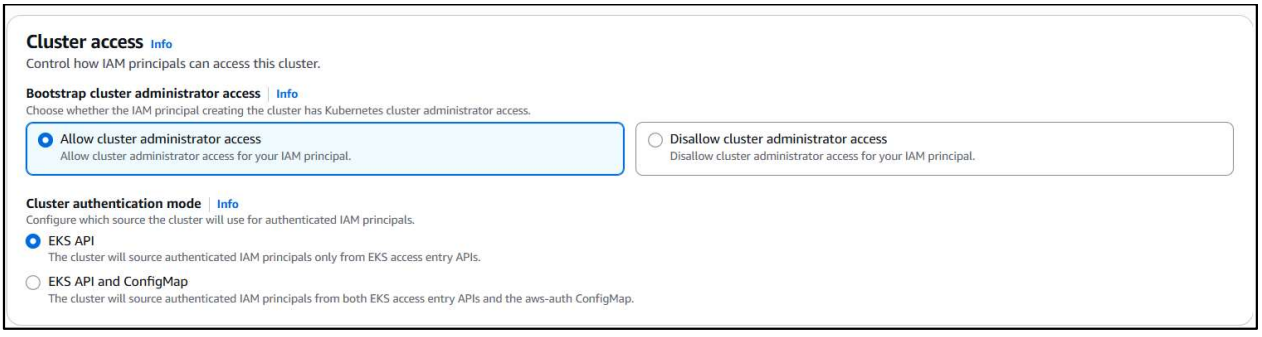


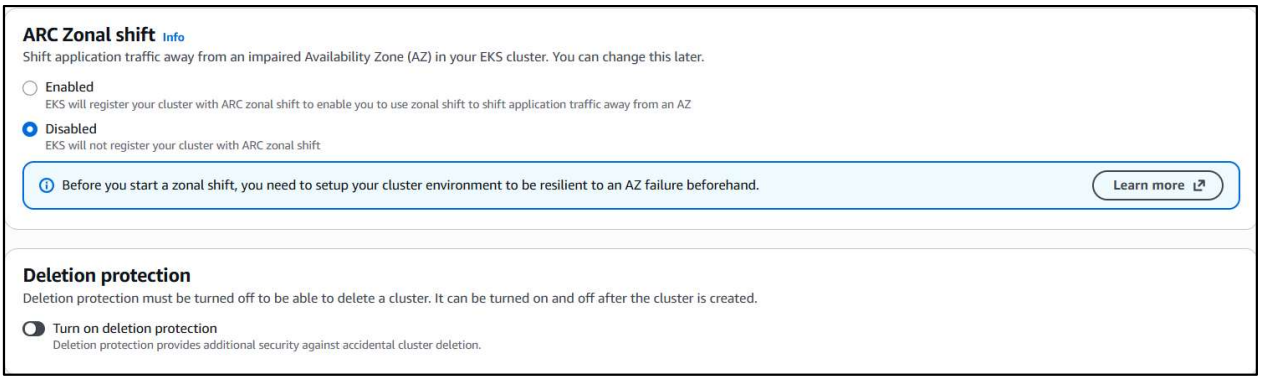


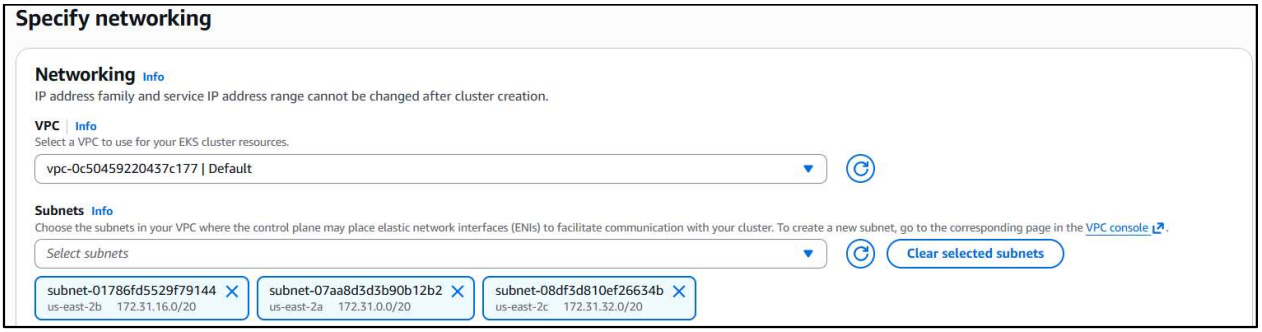

EKS Auto Mode - new Info
Choose if you would like to use EKS's Auto Mode.
Use EKS Auto Mode
EKS automates routine cluster tasks for compute, storage, and networking. When a new pod can't fit onto existing nodes, EKS creates a new node.
EKS combines cluster infrastructure managed by AWS with integrated Kubernetes capabilities to meet application compute needs. View pricing
Included capabilities
Cluster configuration Info
Name
Enter a unique name for this cluster. This property cannot be changed after the cluster is created.
flask-k8s-cluster
The cluster name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 100.
Cluster IAM role Info
Select the Cluster IAM role to allow the Kubernetes control plane to manage AWS resources on your behalf. This cannot be changed after the cluster is created. To create a new custom role, follow the instructions in the Amazon EKS User Guide.
AmazonEKSAutoClusterRole-1
Create recommended role

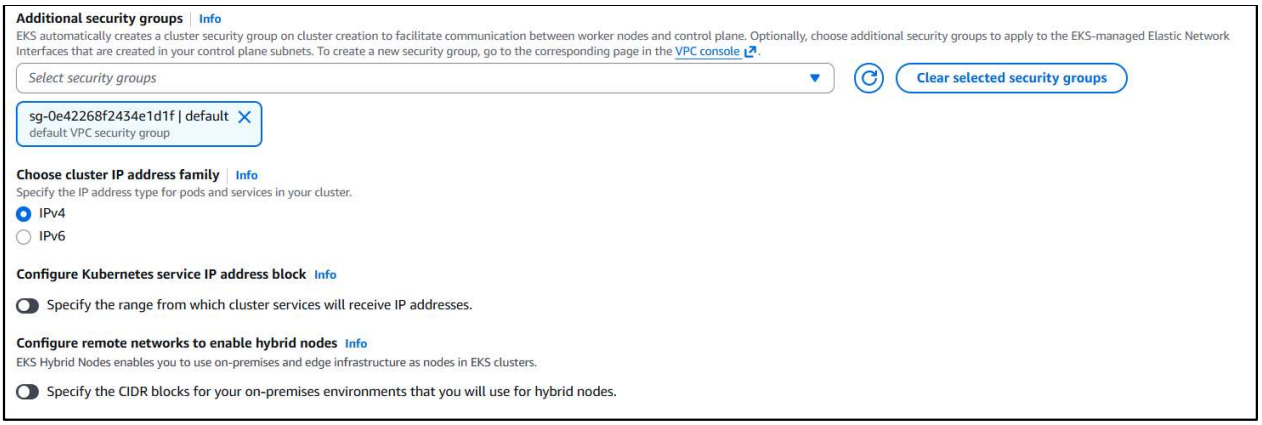

Kubernetes version settings
Kubernetes version Info
Select Kubernetes version for this cluster.
1.34
Upgrade policy Info
Choose one of the following options. You can switch the setting later while the standard support period is in effect.
Standard support
This option supports the Kubernetes version for 14 months after the release date. There is no additional cost. When standard support ends, your cluster will be auto upgraded to the next version.
Extended support
This option supports the Kubernetes version for 26 months after the release date. The extended support period has an additional hourly cost that begins after the standard support period ends. When extended support ends, your cluster will be auto upgraded to the next version.
Control plane scaling tier - new Info
Select a scaling tier for your control plane. While Standard mode scales dynamically, higher tiers pre-provision your environment with fixed, high-performance resources. This eliminates scaling latency and provides the sustained throughput required for demanding workloads. View tier pricing
Use a scaling tier
For predictable and high performance from your cluster's control plane, choose a scaling tier.

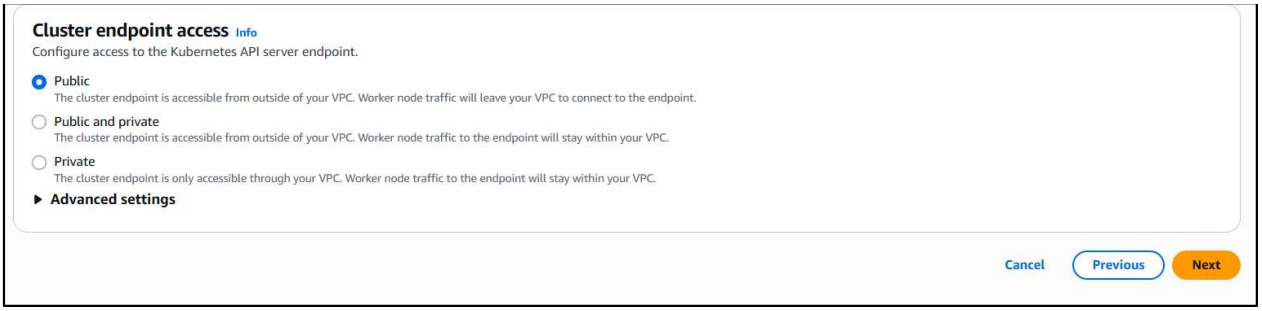

Auto Mode Compute - new Info
Configure node management for your EKS cluster. EKS offers four compute options: EKS Auto Mode, EC2 Managed Node Groups, Fargate, and hybrid nodes. Node groups, Fargate profiles, and hybrid nodes are configured after cluster creation. You can also create self-managed nodes.
Built-in node pools - optional Info
EKS Auto Mode uses node pools to create nodes for pods. The node IAM role will be associated with built-in node pools. Use the Kubernetes API after cluster creation to create your own node pools.
Choose node pool(s)
general-purpose
system
Node IAM role Info
Nodes need an EC2 Instance IAM Role to launch and register with a cluster. To create a new custom role, follow the instructions in the Amazon EKS User Guide.
AmazonEKSAutoNodeRole
Create recommended role

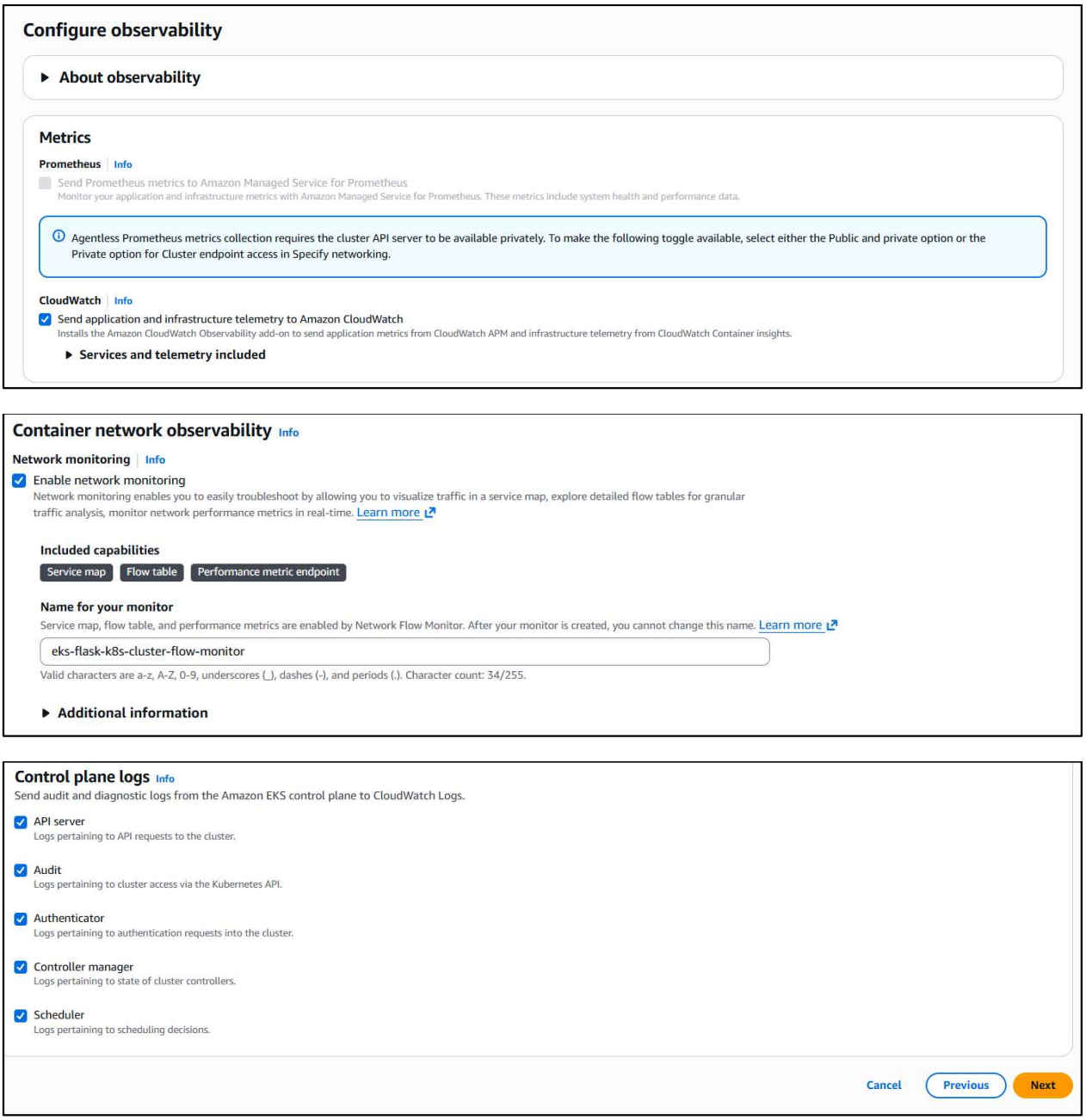

Cluster access Info
Control how IAM principals can access this cluster.
Bootstrap cluster administrator access Info
Choose whether the IAM principal creating the cluster has Kubernetes cluster administrator access.
Allow cluster administrator access
Allow cluster administrator access for your IAM principal.
Disallow cluster administrator access
Disallow cluster administrator access for your IAM principal.
Cluster authentication mode Info
Configure which source the cluster will use for authenticated IAM principals.
EKS API
The cluster will source authenticated IAM principals only from EKS access entry APIs.
EKS API and ConfigMap
The cluster will source authenticated IAM principals from both EKS access entry APIs and the aws-auth ConfigMap.


ARC Zonal shift Info
Shift application traffic away from an impaired Availability Zone (AZ) in your EKS cluster. You can change this later.
Enabled
EKS will register your cluster with ARC zonal shift to enable you to use zonal shift to shift application traffic away from an AZ
Disabled
EKS will not register your cluster with ARC zonal shift
Before you start a zonal shift, you need to setup your cluster environment to be resilient to an AZ failure beforehand.
Learn more
Deletion protection
Deletion protection must be turned off to be able to delete a cluster. It can be turned on and off after the cluster is created.
Turn on deletion protection
Deletion protection provides additional security against accidental cluster deletion.


Specify networking
Networking Info
IP address family and service IP address range cannot be changed after cluster creation.
VPC Info
Select a VPC to use for your EKS cluster resources.
vpc-0c50459220437c177 | Default
Subnets Info
Choose the subnets in your VPC where the control plane may place elastic network interfaces (ENIs) to facilitate communication with your cluster. To create a new subnet, go to the corresponding page in the VPC console.
Select subnets
Clear selected subnets
subnet-01786fd5529f79144
us-east-2b 172.31.16.0/20
subnet-07aa8d3d3b90b12b2
us-east-2a 172.31.0.0/20
subnet-08df3d810ef26634b
us-east-2c 172.31.32.0/20

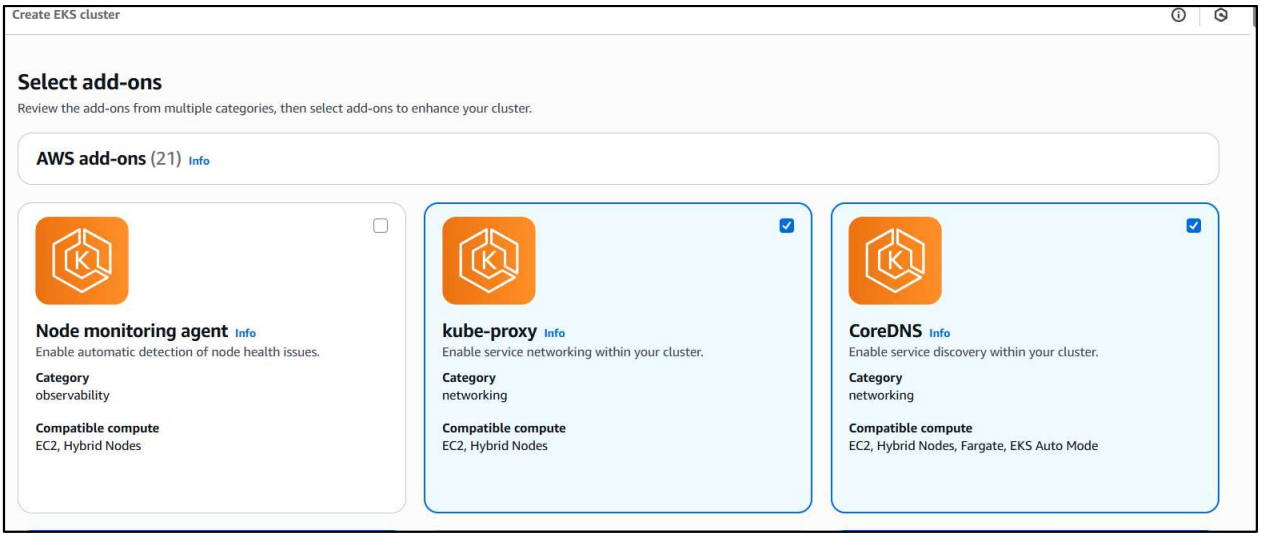

Additional security groups Info
EKS automatically creates a cluster security group on cluster creation to facilitate communication between worker nodes and control plane. Optionally, choose additional security groups to apply to the EKS-managed Elastic Network Interfaces that are created in your control plane subnets. To create a new security group, go to the corresponding page in the VPC console.
Select security groups
Clear selected security groups
sg-0e42268f2434e1d1f | default
default VPC security group
Choose cluster IP address family Info
Specify the IP address type for pods and services in your cluster.
IPv4
IPv6
Configure Kubernetes service IP address block Info
Specify the range from which cluster services will receive IP addresses.
Configure remote networks to enable hybrid nodes Info
EKS Hybrid Nodes enables you to use on-premises and edge infrastructure as nodes in EKS clusters.
Specify the CIDR blocks for your on-premises environments that you will use for hybrid nodes.


Cluster endpoint access Info
Configure access to the Kubernetes API server endpoint.
Public
The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.
Public and private
The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.
Private
The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.
Advanced settings
Cancel
Previous
Next

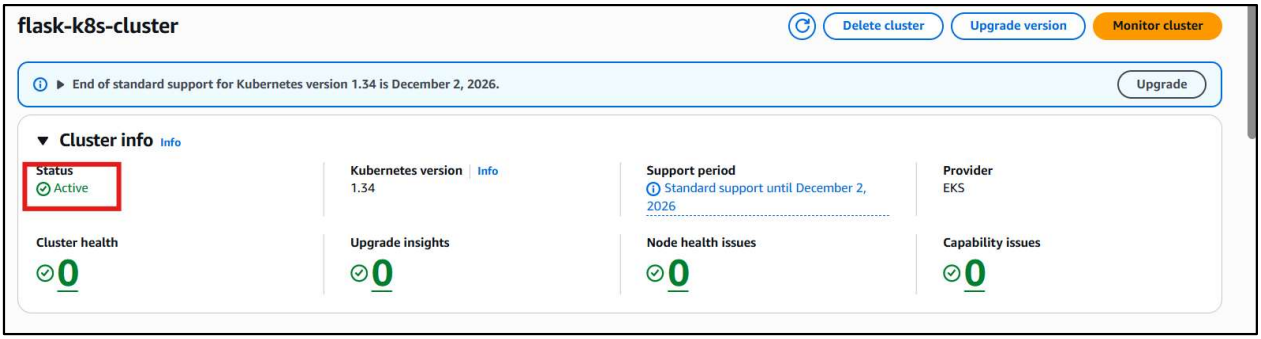

Configure observability
About observability
Metrics
Prometheus | Info
Send Prometheus metrics to Amazon Managed Service for Prometheus
Monitor your application and infrastructure metrics with Amazon Managed Service for Prometheus. These metrics include system health and performance data.
Agentless Prometheus metrics collection requires the cluster API server to be available privately. To make the following toggle available, select either the Public and private option or the Private option for Cluster endpoint access in Specify networking.
CloudWatch | Info
Send application and infrastructure telemetry to Amazon CloudWatch
Installs the Amazon CloudWatch Observability add-on to send application metrics from CloudWatch APM and infrastructure telemetry from CloudWatch Container insights.
Services and telemetry included

Container network observability Info
Network monitoring | Info
Enable network monitoring
Network monitoring enables you to easily troubleshoot by allowing you to visualize traffic in a service map, explore detailed flow tables for granular traffic analysis, monitor network performance metrics in real-time. Learn more
Included capabilities
Service map
Flow table
Performance metric endpoint
Name for your monitor
Service map, flow table, and performance metrics are enabled by Network Flow Monitor. After your monitor is created, you cannot change this name. Learn more
eks-flask-k8s-cluster-flow-monitor
Valid characters are a-z, A-Z, 0-9, underscores (_), dashes (-), and periods (.). Character count: 34/255.
Additional information

Control plane logs Info
Send audit and diagnostic logs from the Amazon EKS control plane to CloudWatch Logs.
API server
Logs pertaining to API requests to the cluster.
Audit
Logs pertaining to cluster access via the Kubernetes API.
Authenticator
Logs pertaining to authentication requests into the cluster.
Controller manager
Logs pertaining to state of cluster controllers.
Scheduler
Logs pertaining to scheduling decisions.
Cancel
Previous
Next

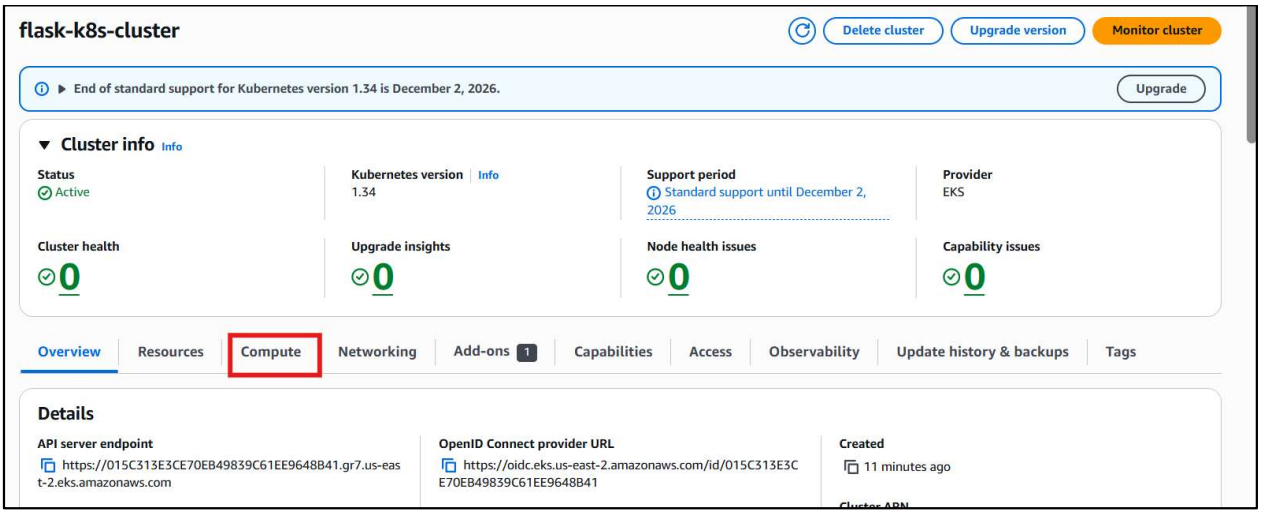


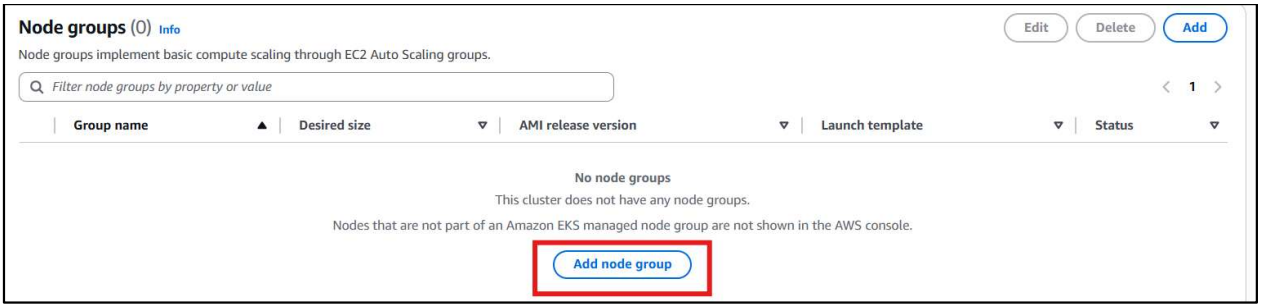
- Cluster is created and active .

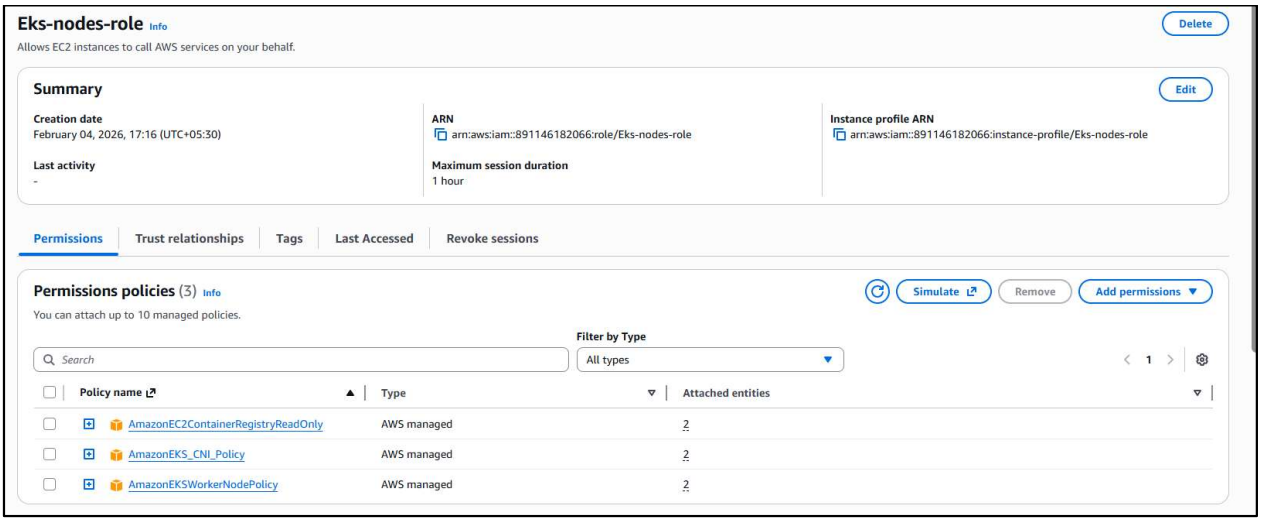


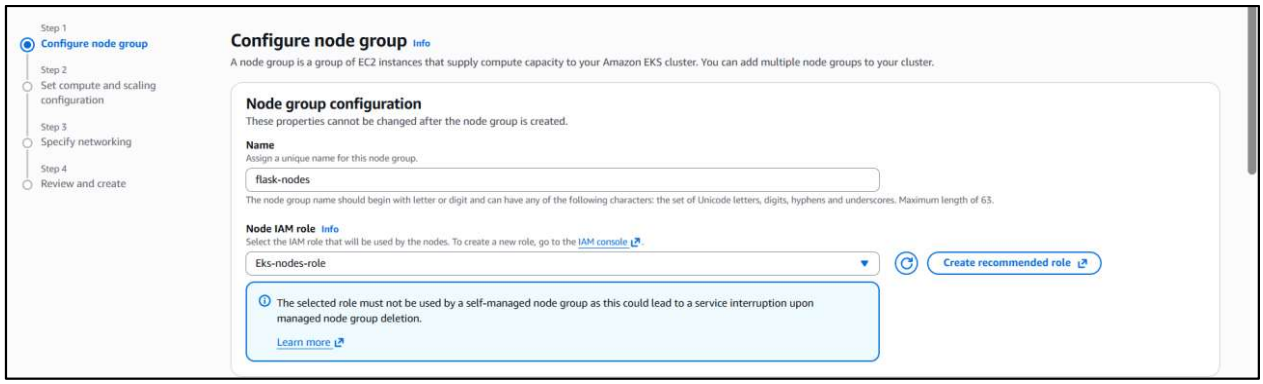
Step 7 : Create Node Group.

- Click on Compute
- Add nodes group.
- Create recommended role.

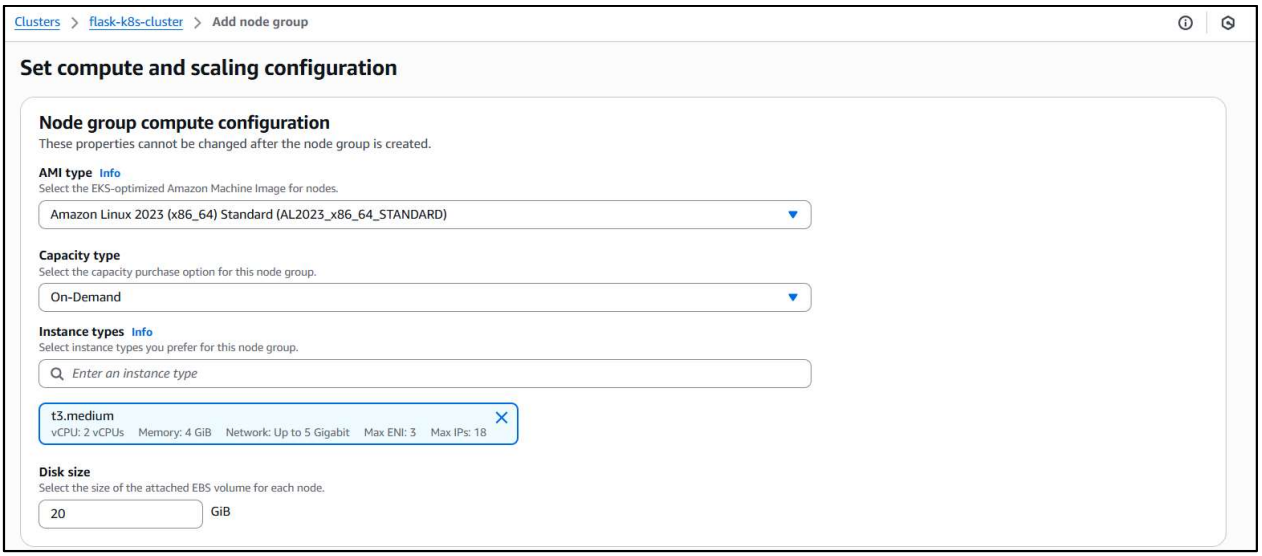




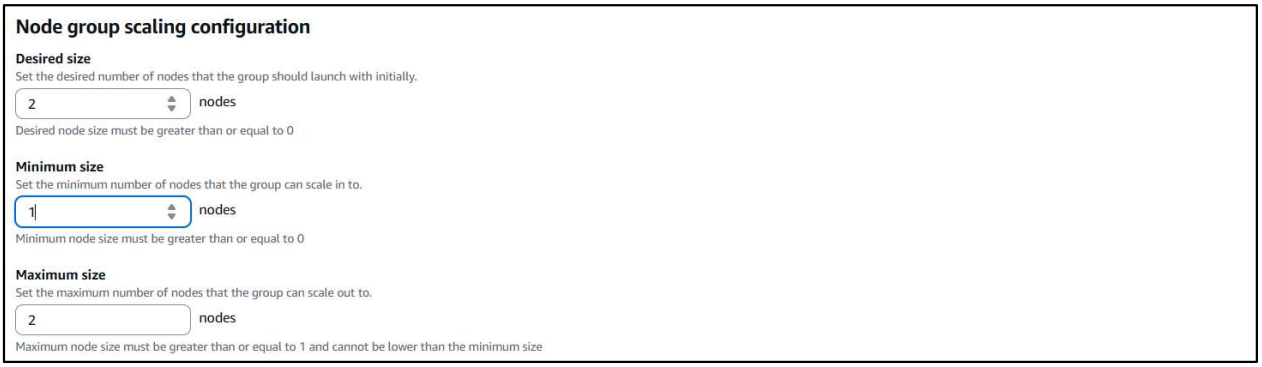



Step 1
Configure node group
Step 2
Set compute and scaling configuration
Step 3
Specify networking
Step 4
Review and create
Configure node group Info
A node group is a group of EC2 instances that supply compute capacity to your Amazon EKS cluster. You can add multiple node groups to your cluster.
Node group configuration
These properties cannot be changed after the node group is created.
Name
Assign a unique name for this node group.
flask-nodes
The node group name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 63.
Node IAM role Info
Select the IAM role that will be used by the nodes. To create a new role, go to the IAM console.
Eks-nodes-role
Create recommended role
The selected role must not be used by a self-managed node group as this could lead to a service interruption upon managed node group deletion.
Learn more

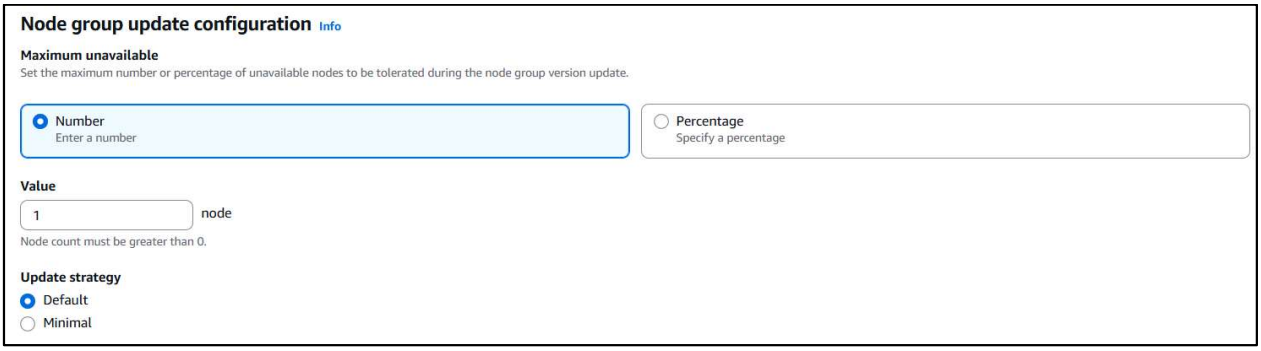
- AMI Type : Default
- Capacity type : On-Demand
- Instance types : t3.medium



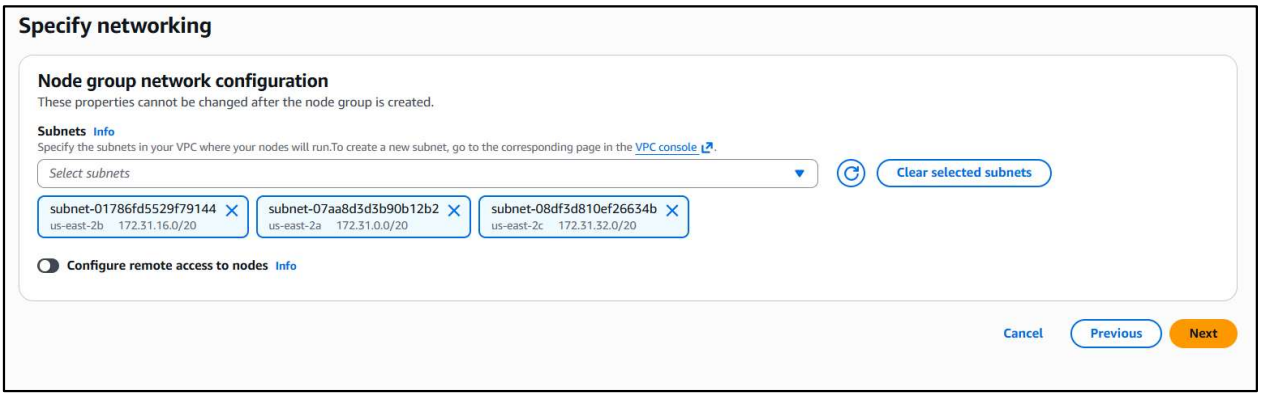
- Set
  - Desired Size : 2
  - Minimum : 2
  - Maximum :2



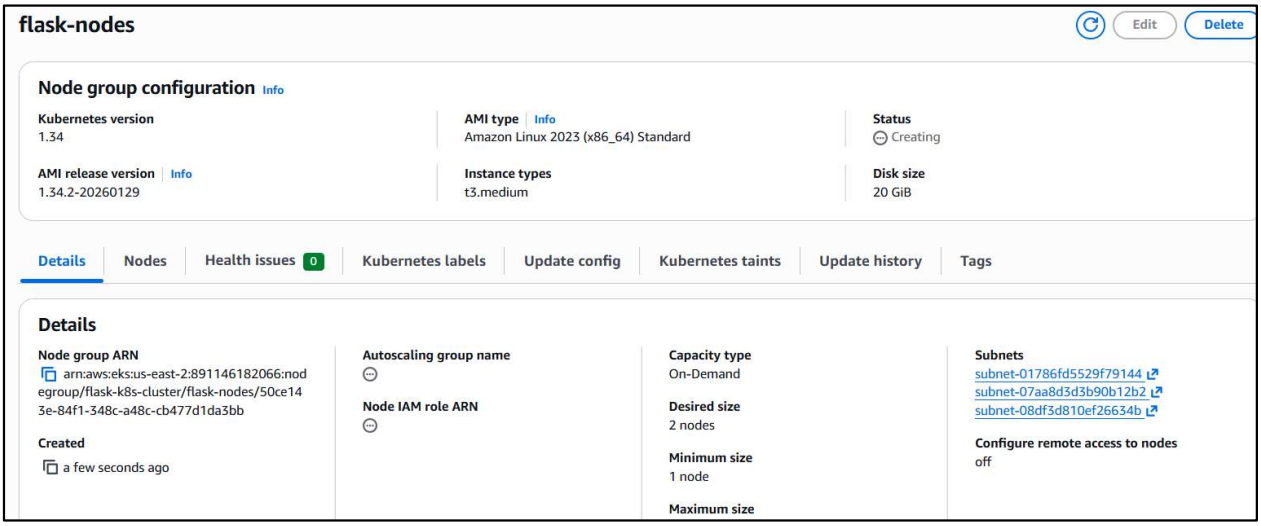
- Set maximum unavilability :1



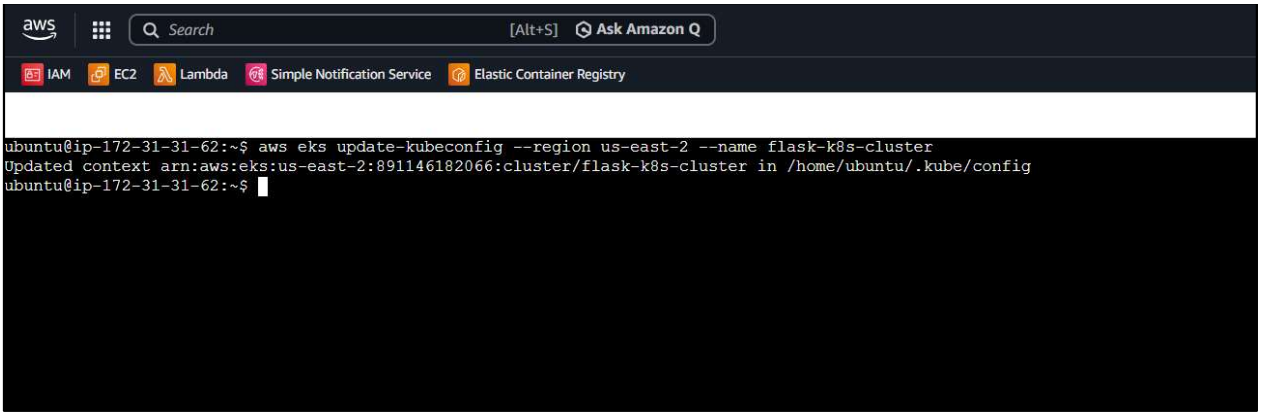
- Select private subnets.



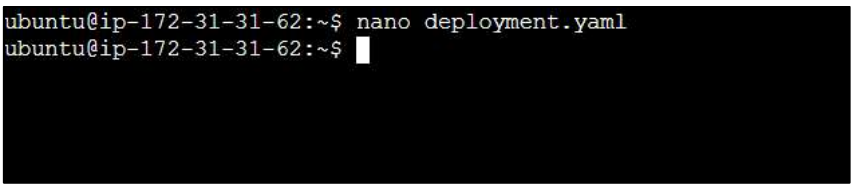
- Node group is created.

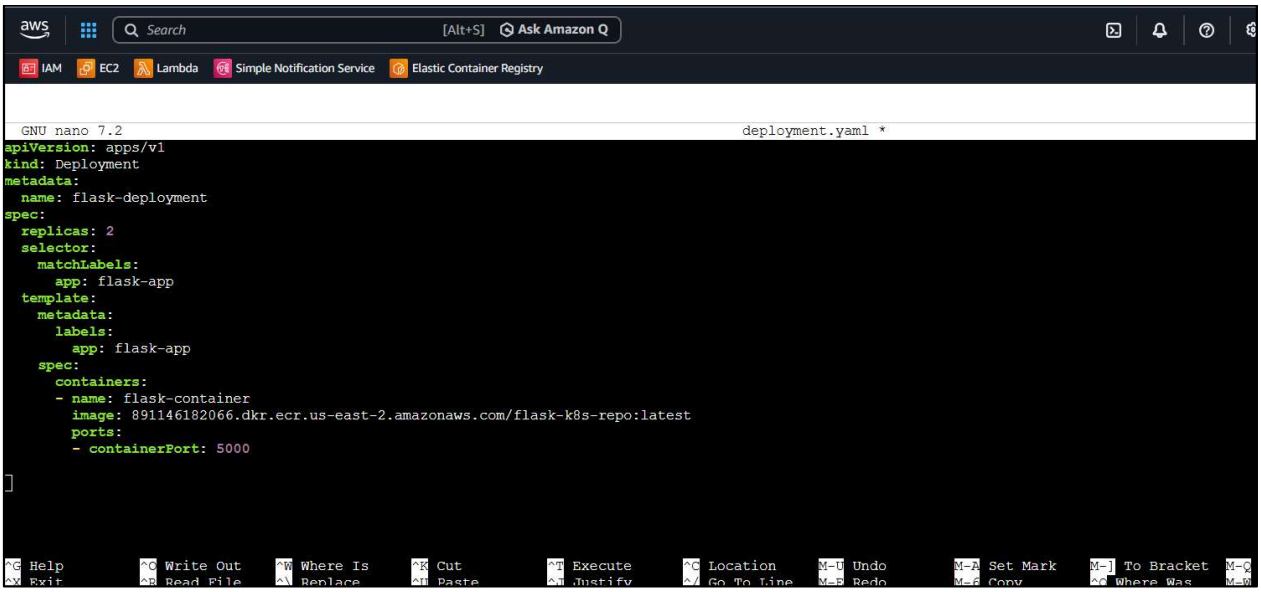


- Connect to EKS cluster
    - aws eks update-kubeconfig --region <region_name> --name <cluster_name>



- Create Deployment YAML
    - nano deployment.yaml

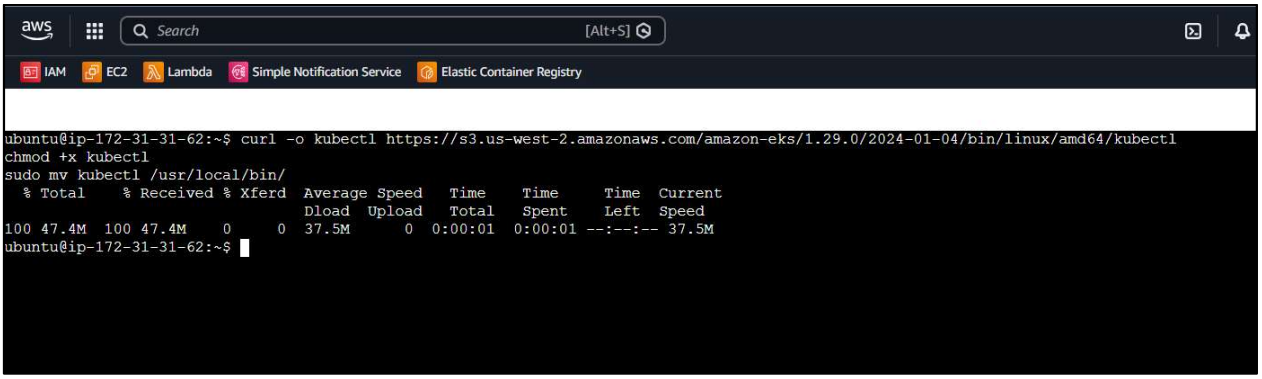




- Create Service YAML
    - nano service.yaml


GNU nano 7.2 service.yaml *
apiVersion: v1
kind: Service
metadata:
name: flask-service
spec:
type: NodePort
selector:
app: flask-app
ports:
- port: 80
targetPort: 5000
nodePort: 30007

- Install kubectl
    - curl -o kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.0/2024-01-04/bin/linux/amd64/kubectl
    - chmod +x kubectl
    - sudo mv kubectl /usr/local/bin/



- Connect to EKS cluster
    - aws eks update-kubeconfig --region <region_name > --name <cluster_name>

```
ubuntu@ip-172-31-31-62:~$ aws eks update-kubeconfig --region us-east-2 --name flask-app-k8s-cluster
Updated context arn:aws:eks:us-east-2:891146182066:cluster/flask-app-k8s-cluster in /home/ubuntu/.kube/config
ubuntu@ip-172-31-31-62:~$
```

- Test:
    - kubectl get nodes

```
ubuntu@ip-172-31-31-62:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
flask-deployment-86b4974695-67n99   1/1     Running   0          6m7s
flask-deployment-86b4974695-nvvzd   1/1     Running   0          6m7s
test-nginx                          1/1     Running   0          9m
```

- Apply configs
    - kubectl apply -f deployment.yaml
    - kubectl apply -f service.yaml

```
ubuntu@ip-172-31-31-62:~$ kubectl apply -f deployment.yaml
deployment.apps/flask-deployment unchanged
ubuntu@ip-172-31-31-62:~$ kubectl apply -f service.yaml
service/flask-service unchanged
ubuntu@ip-172-31-31-62:~$
```

- Check pods:
  - kubectl get pods

```
ubuntu@ip-172-31-31-62:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
flask-deployment-86b4974695-67n99   1/1     Running   0          53m
flask-deployment-86b4974695-nvvzd   1/1     Running   0          53m
test-nginx                          1/1     Running   0          56m
ubuntu@ip-172-31-31-62:~$
```

- Check service:
  - kubectl get svc

```
ubuntu@ip-172-31-31-62:~$ kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
flask-service   NodePort    172.20.30.141   <none>        80:30007/TCP   43s
kubernetes      ClusterIP   172.20.0.1      <none>        443/TCP        147m
```

- Get node public IP:
  - kubectl get nodes -o wide

```
ubuntu@ip-172-31-31-62:~$ kubectl get nodes -o wide
NAME                                          STATUS   ROLES    AGE    VERSION               INTERNAL-IP    EXTERNAL-IP   OS-IMAGE
                KERNEL-VERSION                CONTAINER-RUNTIME
i-038a3db49ded215b9                           Ready    <none>   136m   v1.34.3-eks-3c60543   10.1.11.149    <none>        Bottlerocket (EKS Auto, Standard) 2026.1.28 (aws-k8s-1.
34-standard)   6.12.64                        containerd://2.1.6+bottlerocket
ip-10-1-12-58.us-east-2.compute.internal      Ready    <none>   132m   v1.34.2-eks-ecaa3a6   10.1.12.58     <none>        Amazon Linux 2023.10.20260120
               6.12.64-87.122.amzn2023.x86_64   containerd://2.1.5
```

- For running application on ELB
- need to correctly map **5000 → 80 → 30007** and then access it via **Node IP**.
- Edit service.yaml properly:

```
apiVersion: v1
kind: Service
metadata:
  name: flask-service
spec:
  type: LoadBalancer
  selector:
    app: flask-app
  ports:
    - port: 80
      targetPort: 5000
~
~
~
~
```

- Apply:
  - kubectl apply -f service.yaml

```
ubuntu@ip-172-31-31-62:~$ kubectl apply -f service.yaml
service/flask-service unchanged
```

- kubectl get svc flask-service

```
ubuntu@ip-172-31-31-62:~$ kubectl get svc flask-service
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
flask-service   LoadBalancer   172.20.30.141   afa92a4dcee094763aa626d90f4ac029-1408905745.us-east-2.elb.amazonaws.com   80:30007/TCP   60m
ubuntu@ip-172-31-31-62:~$
```

- Check Application on Browser
  - Http://<external-Ip>link