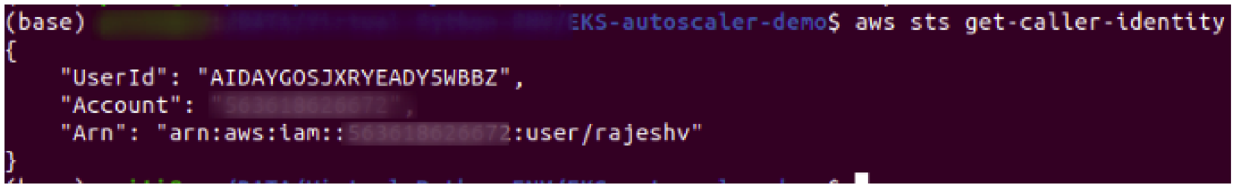
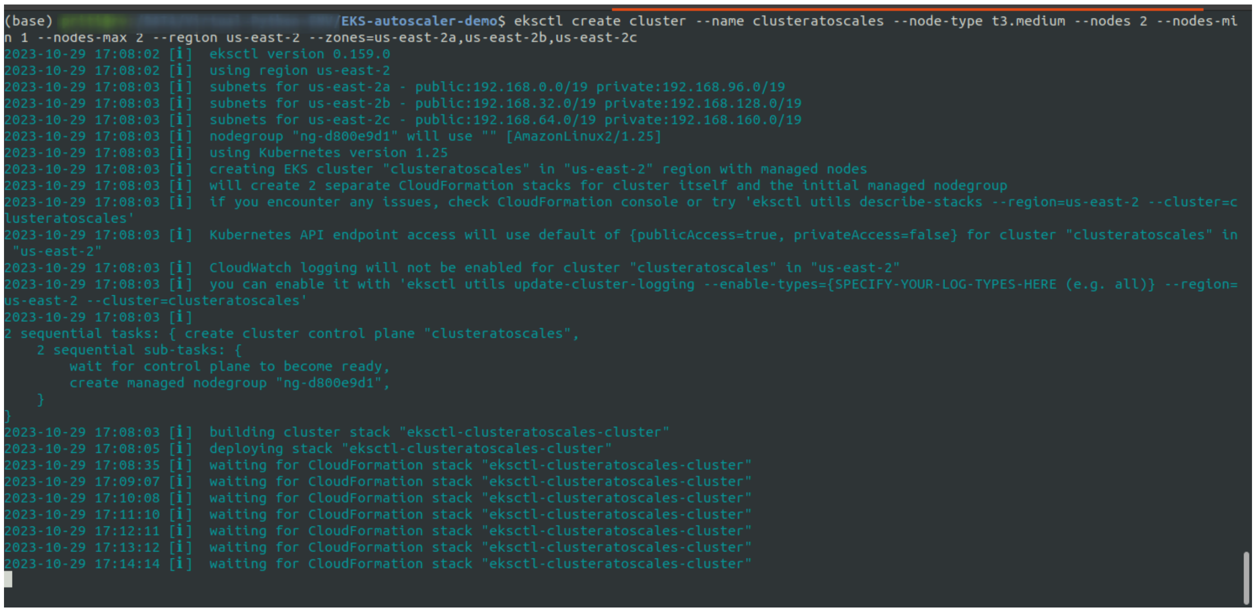
Check your AWS IAM user access:

aws sts get-caller-identity



```
(base) [redacted]EKS-autoscaler-demo$ aws sts get-caller-identity
{
    "UserId": "AIDAYGOSJXRYEADY5WBBZ",
    "Account": [redacted]
    "Arn": "arn:aws:iam::[redacted]:user/rajeshv"
}
```

Create EKS Cluster:



```
(base) [redacted]EKS-autoscaler-demo$ eksctl create cluster --name clusteratoscales --node-type t3.medium --nodes 2 --nodes-mi
n 1 --nodes-max 2 --region us-east-2 --zones=us-east-2a,us-east-2b,us-east-2c
2023-10-29 17:08:02 [i]  eksctl version 0.159.0
2023-10-29 17:08:02 [i]  using region us-east-2
2023-10-29 17:08:03 [i]  subnets for us-east-2a - public:192.168.0.0/19 private:192.168.96.0/19
2023-10-29 17:08:03 [i]  subnets for us-east-2b - public:192.168.32.0/19 private:192.168.128.0/19
2023-10-29 17:08:03 [i]  subnets for us-east-2c - public:192.168.64.0/19 private:192.168.160.0/19
2023-10-29 17:08:03 [i]  nodegroup "ng-d800e9d1" will use "" [AmazonLinux2/1.25]
2023-10-29 17:08:03 [i]  using Kubernetes version 1.25
2023-10-29 17:08:03 [i]  creating EKS cluster "clusteratoscales" in "us-east-2" region with managed nodes
2023-10-29 17:08:03 [i]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2023-10-29 17:08:03 [i]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-2 --cluster=c
lusteratoscales'
2023-10-29 17:08:03 [i]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "clusteratoscales" in
 "us-east-2"
2023-10-29 17:08:03 [i]  CloudWatch logging will not be enabled for cluster "clusteratoscales" in "us-east-2"
2023-10-29 17:08:03 [i]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=
us-east-2 --cluster=clusteratoscales'
2023-10-29 17:08:03 [i]
2 sequential tasks: { create cluster control plane "clusteratoscales",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create managed nodegroup "ng-d800e9d1",
    }
}
2023-10-29 17:08:03 [i]  building cluster stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:08:05 [i]  deploying stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:08:35 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:09:07 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:10:08 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:11:10 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:12:11 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:13:12 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:14:14 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
```

Logs:

```
(base) priti@rv:/DATA/Virtual-Python-ENV/EKS-autoscaler-demo$ eksctl create cluster --name
clusteratoscales --node-type t3.medium --nodes 2 --nodes-min 1 --nodes-max 2 --region
us-east-2 --zones=us-east-2a,us-east-2b,us-east-2c
2023-10-29 17:08:02 [i]  eksctl version 0.159.0
2023-10-29 17:08:02 [i]  using region us-east-2
2023-10-29 17:08:03 [i]  subnets for us-east-2a - public:192.168.0.0/19 private:192.168.96.0/19
2023-10-29 17:08:03 [i]  subnets for us-east-2b - public:192.168.32.0/19
private:192.168.128.0/19
2023-10-29 17:08:03 [i]  subnets for us-east-2c - public:192.168.64.0/19
private:192.168.160.0/19
2023-10-29 17:08:03 [i]  nodegroup "ng-d800e9d1" will use "" [AmazonLinux2/1.25]
2023-10-29 17:08:03 [i]  using Kubernetes version 1.25
2023-10-29 17:08:03 [i]  creating EKS cluster "clusteratoscales" in "us-east-2" region with
managed nodes
2023-10-29 17:08:03 [i]  will create 2 separate CloudFormation stacks for cluster itself and the
initial managed nodegroup
```

```
2023-10-29 17:08:03 [i]  if you encounter any issues, check CloudFormation console or try
'eksctl utils describe-stacks --region=us-east-2 --cluster=clusteratoscales'
2023-10-29 17:08:03 [i]  Kubernetes API endpoint access will use default of {publicAccess=true,
privateAccess=false} for cluster "clusteratoscales" in "us-east-2"
2023-10-29 17:08:03 [i]  CloudWatch logging will not be enabled for cluster "clusteratoscales" in
"us-east-2"
2023-10-29 17:08:03 [i]  you can enable it with 'eksctl utils update-cluster-logging
--enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-east-2
--cluster=clusteratoscales'
2023-10-29 17:08:03 [i]
2 sequential tasks: { create cluster control plane "clusteratoscales",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create managed nodegroup "ng-d800e9d1",
    }
}
2023-10-29 17:08:03 [i]  building cluster stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:08:05 [i]  deploying stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:08:35 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:09:07 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:10:08 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:11:10 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:12:11 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:13:12 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:14:14 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:15:15 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:16:16 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
2023-10-29 17:17:18 [i]  waiting for CloudFormation stack "eksctl-clusteratoscales-cluster"
```

```
2023-10-29 17:19:28 [i]  building managed nodegroup stack
"eksctl-clusteratoscales-nodegroup-ng-d800e9d1"
2023-10-29 17:19:30 [i]  deploying stack "eksctl-clusteratoscales-nodegroup-ng-d800e9d1"
2023-10-29 17:19:30 [i]  waiting for CloudFormation stack
"eksctl-clusteratoscales-nodegroup-ng-d800e9d1"
2023-10-29 17:20:01 [i]  waiting for CloudFormation stack
"eksctl-clusteratoscales-nodegroup-ng-d800e9d1"
```

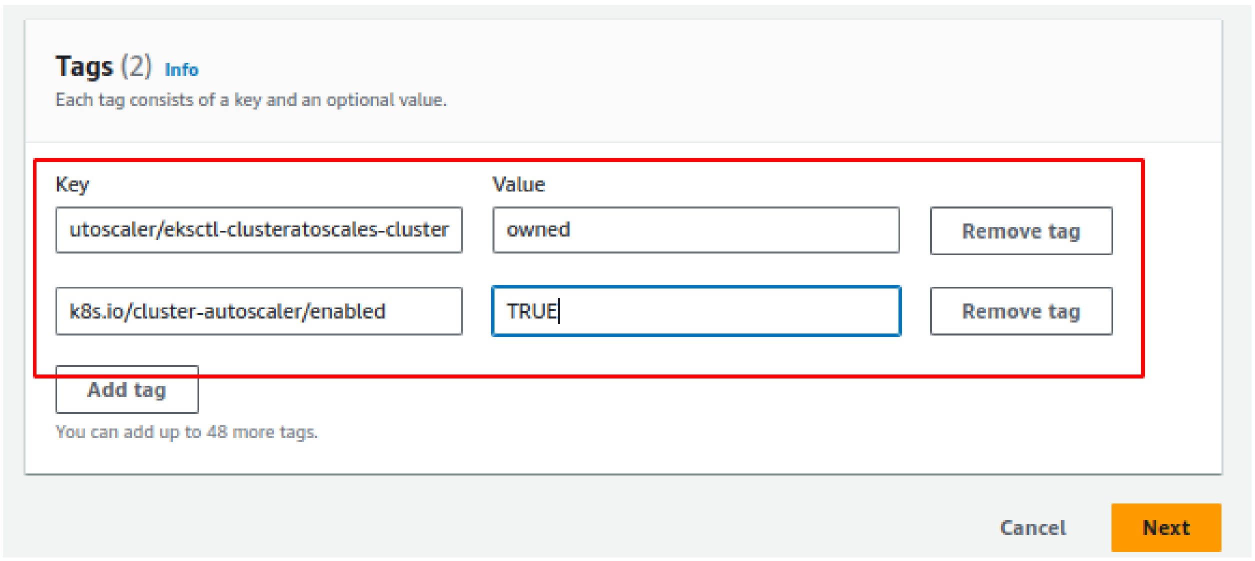
```
2023-10-29 17:20:42 [i]  waiting for CloudFormation stack
"eksctl-clusteratoscales-nodegroup-ng-d800e9d1"
```

```
2023-10-29 17:21:59 [i]  waiting for CloudFormation stack
"eksctl-clusteratoscales-nodegroup-ng-d800e9d1"
2023-10-29 17:21:59 [i]  waiting for the control plane to become ready
2023-10-29 17:22:01 [✔]  saved kubeconfig as "/home/priti/.kube/config"
2023-10-29 17:22:01 [i]  no tasks
2023-10-29 17:22:01 [✔]  all EKS cluster resources for "clusteratoscales" have been created
2023-10-29 17:22:02 [i]  nodegroup "ng-d800e9d1" has 2 node(s)
2023-10-29 17:22:02 [i]  node "ip-192-168-59-27.us-east-2.compute.internal" is ready
2023-10-29 17:22:02 [i]  node "ip-192-168-70-219.us-east-2.compute.internal" is ready
2023-10-29 17:22:02 [i]  waiting for at least 1 node(s) to become ready in "ng-d800e9d1"
2023-10-29 17:22:02 [i]  nodegroup "ng-d800e9d1" has 2 node(s)
2023-10-29 17:22:02 [i]  node "ip-192-168-59-27.us-east-2.compute.internal" is ready
2023-10-29 17:22:02 [i]  node "ip-192-168-70-219.us-east-2.compute.internal" is ready
2023-10-29 17:22:04 [i]  kubectl command should work with "/home/priti/.kube/config", try
'kubectl get nodes'
2023-10-29 17:22:04 [✔]  EKS cluster "clusteratoscales" in "us-east-2" region is ready
```

To work autoscaling

Even in prod ,you have to create separate nodegroup for autoscaling:

Created Separate Node group for autoscaling:

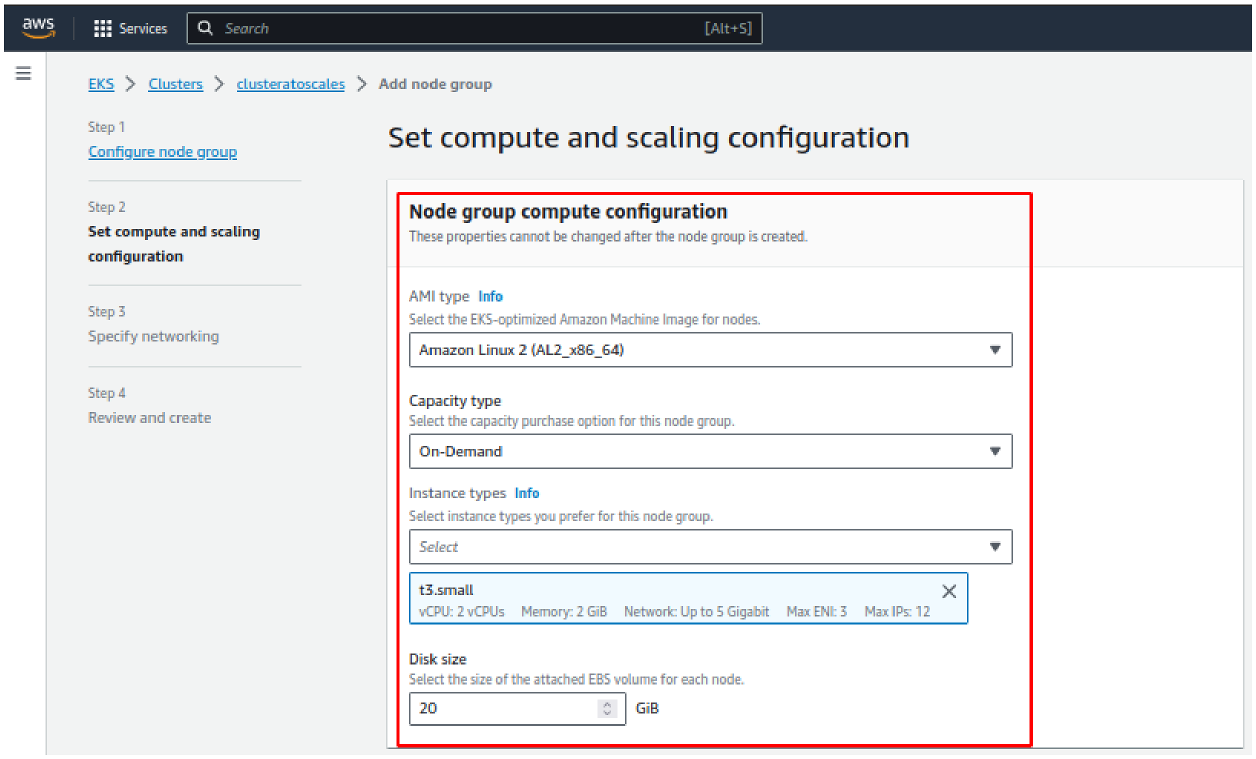


Tags (2) Info

Each tag consists of a key and an optional value.

Key	Value
utoscaler/eksctl-clusteratoscales-cluster	owned
k8s.io/cluster-autoscaler/enabled	TRUE

You can add up to 48 more tags.



Set compute and scaling configuration

Node group compute configuration

These properties cannot be changed after the node group is created.

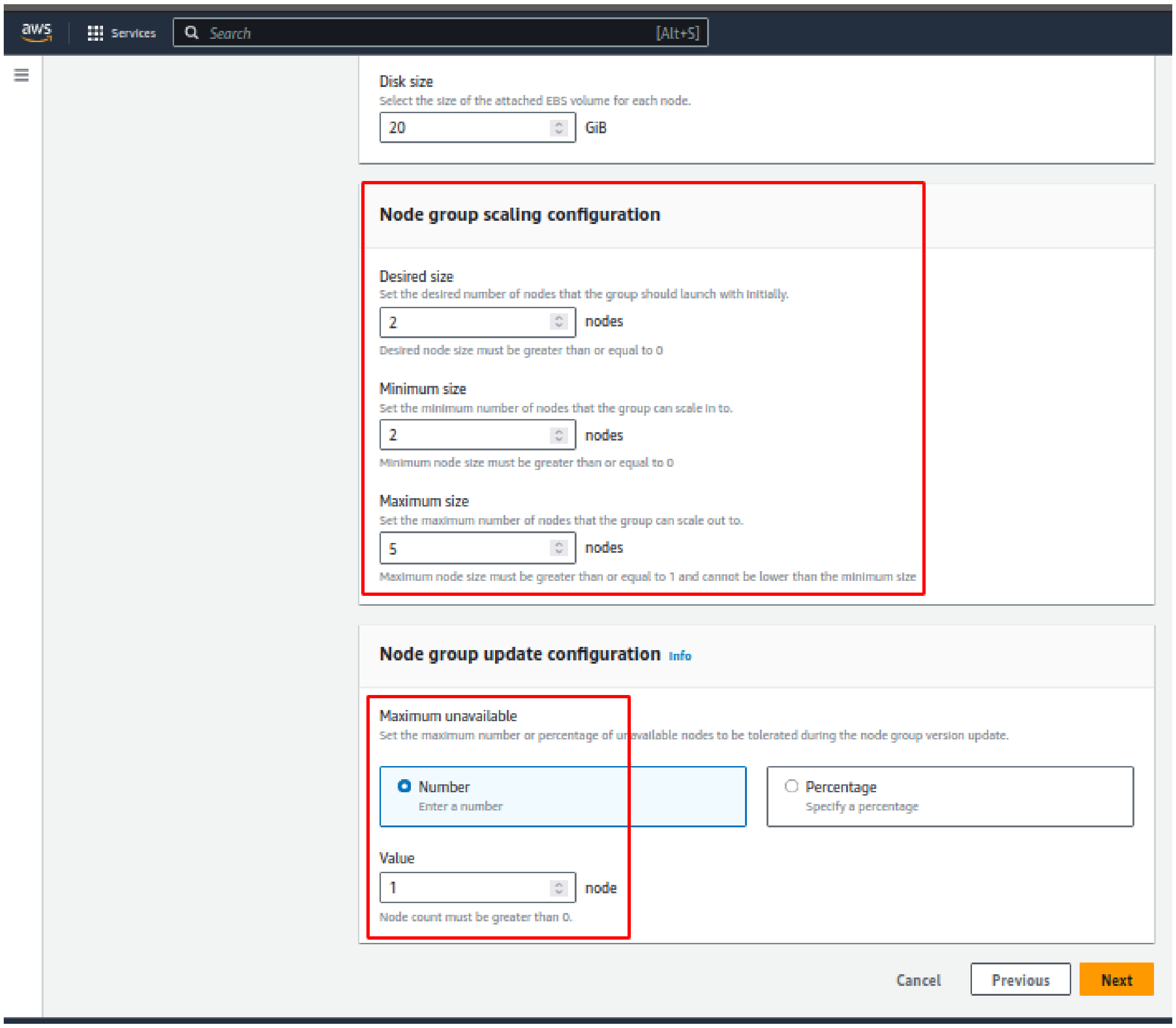
AMI type Info
Select the EKS-optimized Amazon Machine Image for nodes.
Amazon Linux 2 (AL2_x86_64)

Capacity type
Select the capacity purchase option for this node group.
On-Demand

Instance types Info
Select instance types you prefer for this node group.
Select

t3.small
vCPU: 2 vCPUs Memory: 2 GiB Network: Up to 5 Gigabit Max ENI: 3 Max IPs: 12

Disk size
Select the size of the attached EBS volume for each node.
20 GiB



Disk size
Select the size of the attached EBS volume for each node.
20 GiB

Node group scaling configuration

Desired size
Set the desired number of nodes that the group should launch with initially.
2 nodes
Desired node size must be greater than or equal to 0

Minimum size
Set the minimum number of nodes that the group can scale in to.
2 nodes
Minimum node size must be greater than or equal to 0

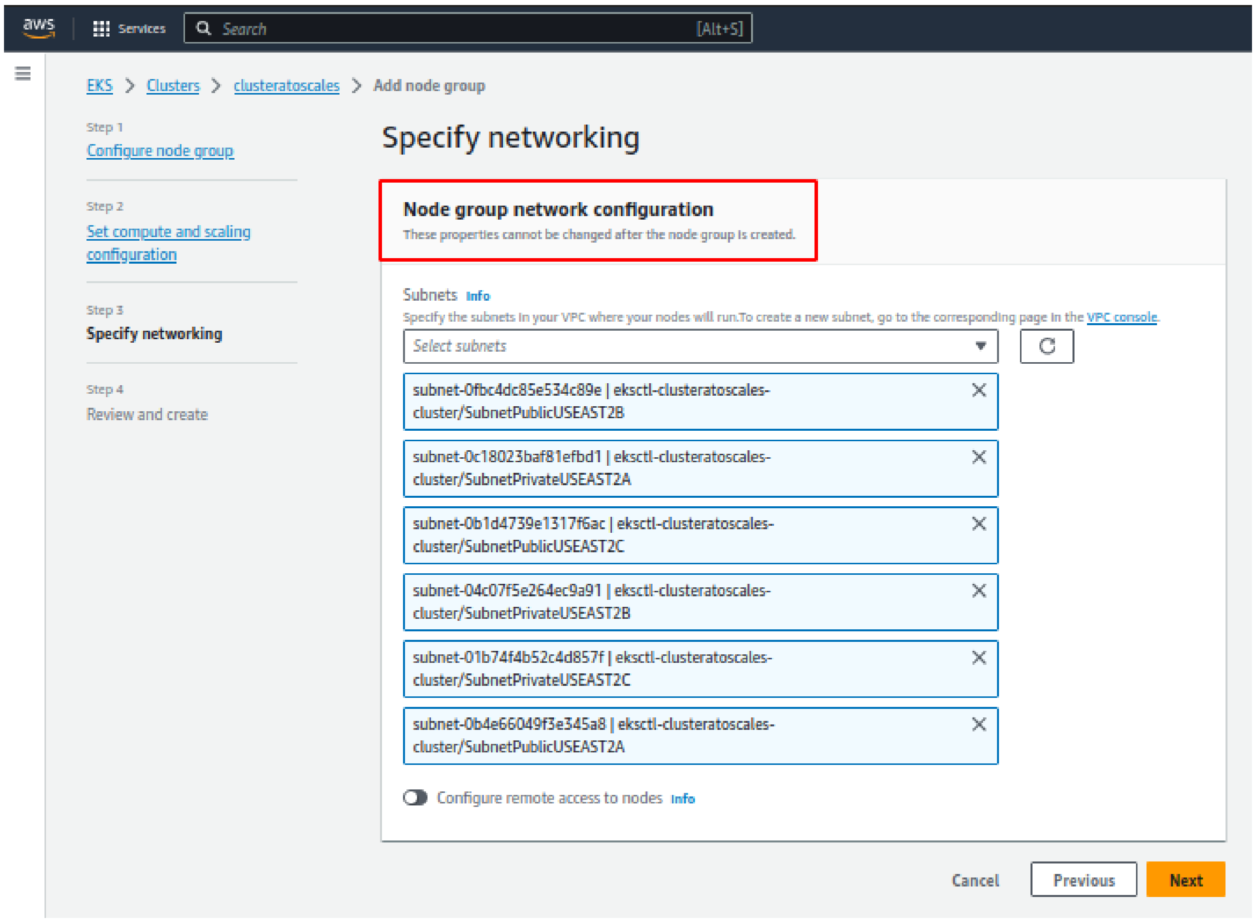
Maximum size
Set the maximum number of nodes that the group can scale out to.
5 nodes
Maximum node size must be greater than or equal to 1 and cannot be lower than the minimum size

Node group update configuration Info

Maximum unavailable
Set the maximum number or percentage of unavailable nodes to be tolerated during the node group version update.

[✓ Number] Enter a number
Percentage Specify a percentage

Value
1 node
Node count must be greater than 0.



EKS > Clusters > clusteratoscales > Add node group

Step 1
Configure node group

Step 2
Set compute and scaling configuration

Step 3
Specify networking

Step 4
Review and create

Specify networking

Node group network configuration
These properties cannot be changed after the node group is created.

Subnets Info
Specify the subnets in your VPC where your nodes will run.To create a new subnet, go to the corresponding page in the VPC console.

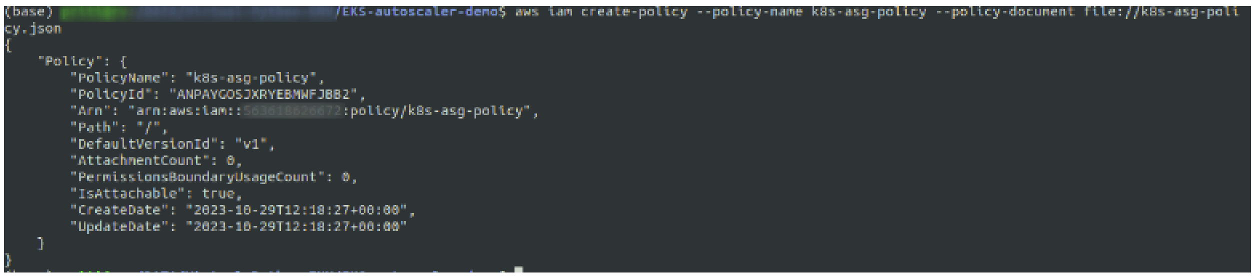
Select subnets

- subnet-0fbc4dc85e534c89e | eksctl-clusteratoscales-cluster/SubnetPublicUSEAST2B
- subnet-0c18023baf81efbd1 | eksctl-clusteratoscales-cluster/SubnetPrivateUSEAST2A
- subnet-0b1d4739e1317f6ac | eksctl-clusteratoscales-cluster/SubnetPublicUSEAST2C
- subnet-04c07f5e264ec9a91 | eksctl-clusteratoscales-cluster/SubnetPrivateUSEAST2B
- subnet-01b74f4b52c4d857f | eksctl-clusteratoscales-cluster/SubnetPrivateUSEAST2C
- subnet-0b4e66049f3e345a8 | eksctl-clusteratoscales-cluster/SubnetPublicUSEAST2A

Configure remote access to nodes Info

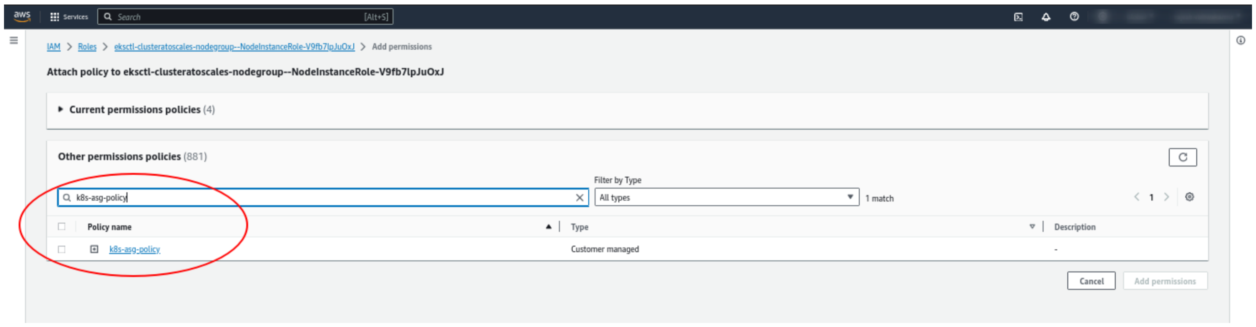
Cancel Previous Next

Create IAM policy:
aws iam create-policy --policy-name k8s-asg-policy --policy-document file://k8s-asg-policy.json



```
(base) [redacted]/EKS-autoscaler-demo$ aws iam create-policy --policy-name k8s-asg-policy --policy-document file://k8s-asg-poli
cy.json
{
    "Policy": {
        "PolicyName": "k8s-asg-policy",
        "PolicyId": "ANPAYGOSJXRYEBMWFJBB2",
        "Arn": "arn:aws:iam::[redacted]:policy/k8s-asg-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-10-29T12:18:27+00:00",
        "UpdateDate": "2023-10-29T12:18:27+00:00"
    }
}
```

Now Go to Cluster >> Configuration >> Node group >> From Detail tab Click on Role ARN



IAM > Roles > eksctl-clusteratoscales-nodegroup--NodeInstanceRole-V9fb7lpJuOxJ > Add permissions

Attach policy to eksctl-clusteratoscales-nodegroup--NodeInstanceRole-V9fb7lpJuOxJ

Current permissions policies (4)

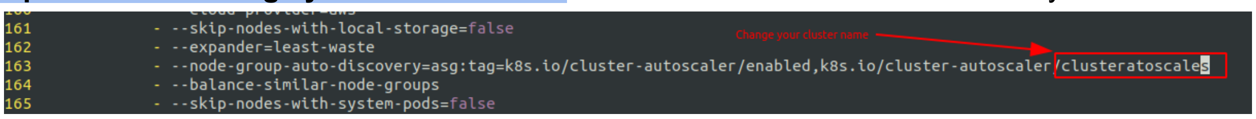
Other permissions policies (881)

Search: k8s-asg-policy — Filter by Type: All types — 1 match

Policy name	Type	Description
k8s-asg-policy	Customer managed	-

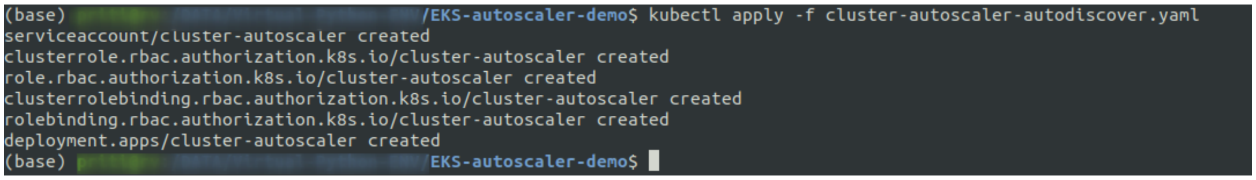
Deploy the Cluster Autoscaler on Management control plan

Open file and change your cluster name: vim cluster-autoscaler-autodiscover.yaml



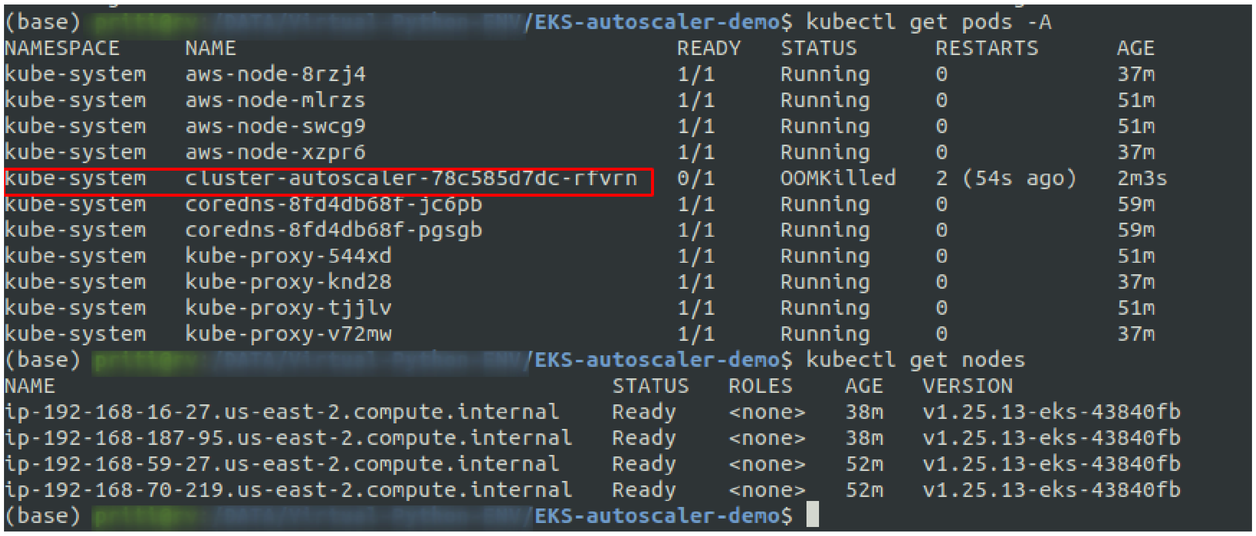
```
            - --skip-nodes-with-local-storage=false
            - --expander=least-waste
            - --node-group-auto-discovery=asg:tag=k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/clusteratoscales
            - --balance-similar-node-groups
            - --skip-nodes-with-system-pods=false
```

Apply the changes:



```
(base) [redacted]/EKS-autoscaler-demo$ kubectl apply -f cluster-autoscaler-autodiscover.yaml
serviceaccount/cluster-autoscaler created
clusterrole.rbac.authorization.k8s.io/cluster-autoscaler created
role.rbac.authorization.k8s.io/cluster-autoscaler created
clusterrolebinding.rbac.authorization.k8s.io/cluster-autoscaler created
rolebinding.rbac.authorization.k8s.io/cluster-autoscaler created
deployment.apps/cluster-autoscaler created
(base) [redacted]/EKS-autoscaler-demo$
```

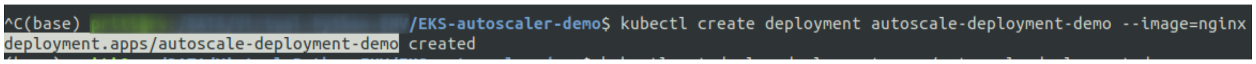
Now we have 4 Nodes with cluster autoscaler pod deployed:



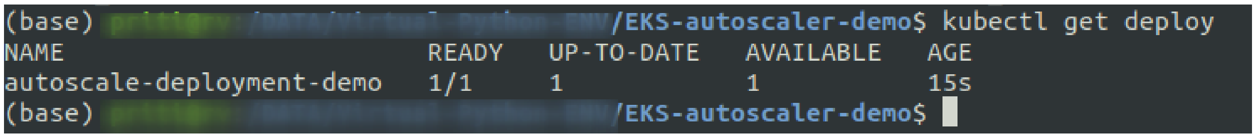
```
(base) [redacted]/EKS-autoscaler-demo$ kubectl get pods -A
NAMESPACE     NAME                                  READY   STATUS      RESTARTS       AGE
kube-system   aws-node-8rzj4                        1/1     Running     0              37m
kube-system   aws-node-mlrzs                        1/1     Running     0              51m
kube-system   aws-node-swcg9                        1/1     Running     0              51m
kube-system   aws-node-xzpr6                        1/1     Running     0              37m
kube-system   cluster-autoscaler-78c585d7dc-rfvrn   0/1     OOMKilled   2 (54s ago)    2m3s
kube-system   coredns-8fd4db68f-jc6pb               1/1     Running     0              59m
kube-system   coredns-8fd4db68f-pgsgb               1/1     Running     0              59m
kube-system   kube-proxy-544xd                      1/1     Running     0              51m
kube-system   kube-proxy-knd28                      1/1     Running     0              37m
kube-system   kube-proxy-tjjlv                      1/1     Running     0              51m
kube-system   kube-proxy-v72mw                      1/1     Running     0              37m
(base) [redacted]/EKS-autoscaler-demo$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
ip-192-168-16-27.us-east-2.compute.internal   Ready    <none>   38m   v1.25.13-eks-43840fb
ip-192-168-187-95.us-east-2.compute.internal  Ready    <none>   38m   v1.25.13-eks-43840fb
ip-192-168-59-27.us-east-2.compute.internal   Ready    <none>   52m   v1.25.13-eks-43840fb
ip-192-168-70-219.us-east-2.compute.internal  Ready    <none>   52m   v1.25.13-eks-43840fb
(base) [redacted]EKS-autoscaler-demo$
```

Note: To perform an auto scaling demo, mimicked as the actual traffic on EKS cluster in production environment I created nginx deployment with single pod and scaled it by 100 pods, so then cluster auto scaled out and when traffic decreased EKS clustered scaled in with default nodes.

Created nginx deployment with one pod

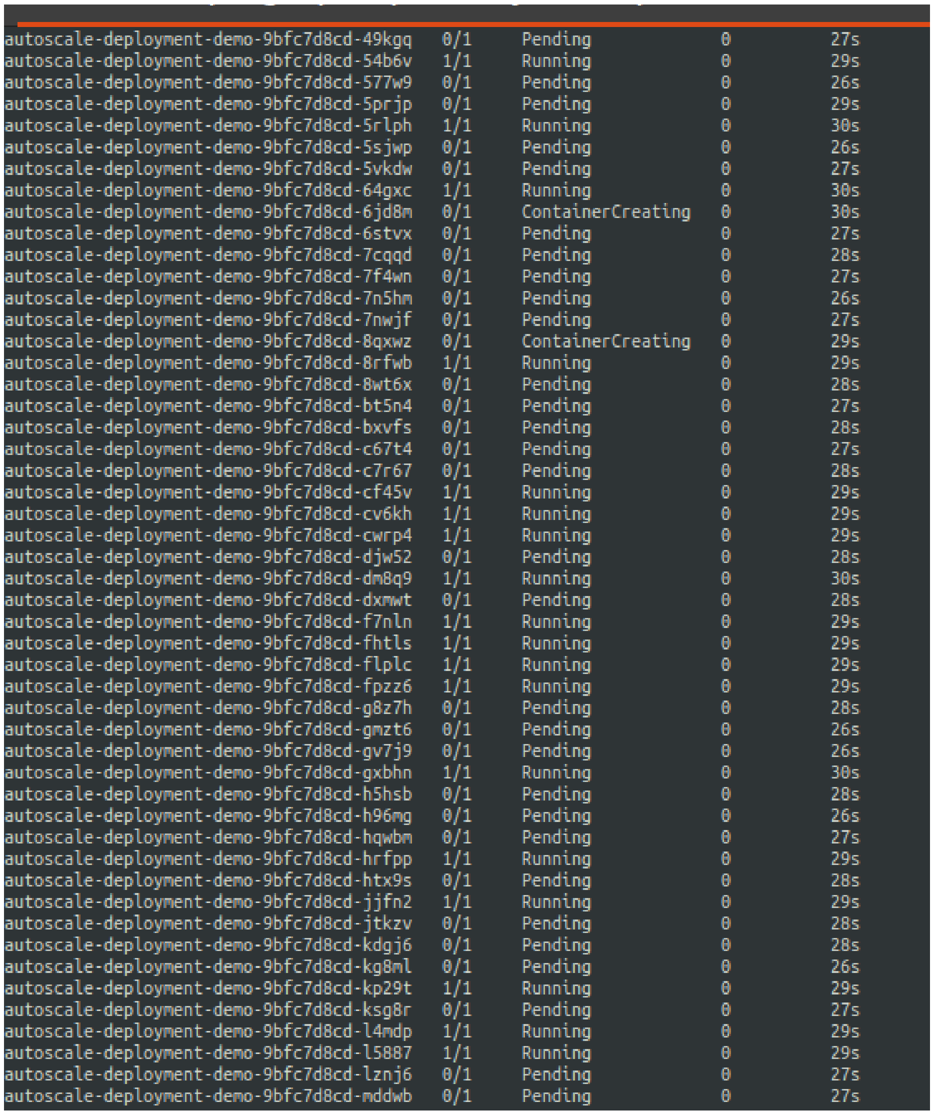


```
^C(base) [redacted]/EKS-autoscaler-demo$ kubectl create deployment autoscale-deployment-demo --image=nginx
deployment.apps/autoscale-deployment-demo created
```



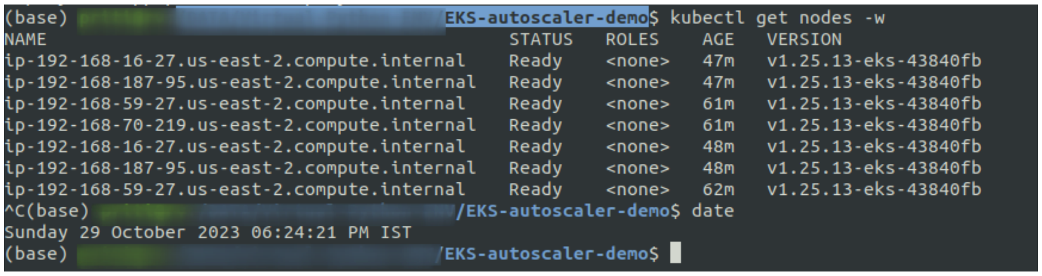
```
(base) [redacted]/EKS-autoscaler-demo$ kubectl get deploy
NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
autoscale-deployment-demo   1/1     1            1           15s
(base) [redacted]/EKS-autoscaler-demo$
```

Scaled out the deployment with 100 Pods, and watched EKS cluster auto scaling works.



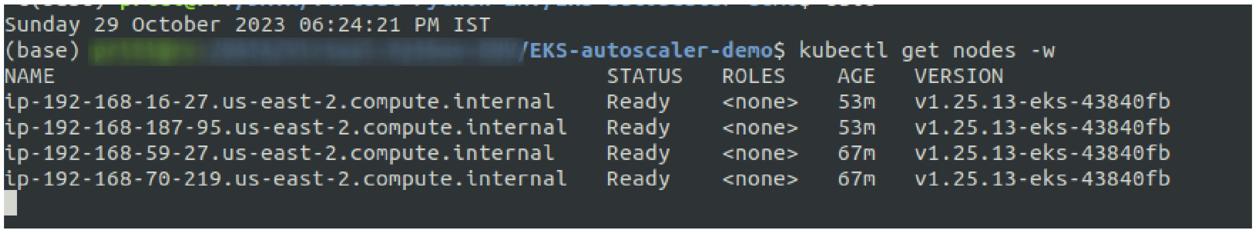
```
autoscale-deployment-demo-9bfc7d8cd-49kgq   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-54b6v   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-577w9   0/1     Pending             0          26s
autoscale-deployment-demo-9bfc7d8cd-5prjp   0/1     Pending             0          29s
autoscale-deployment-demo-9bfc7d8cd-5rlph   1/1     Running             0          30s
autoscale-deployment-demo-9bfc7d8cd-5sjwp   0/1     Pending             0          26s
autoscale-deployment-demo-9bfc7d8cd-5vkdw   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-64gxc   1/1     Running             0          30s
autoscale-deployment-demo-9bfc7d8cd-6jd8m   0/1     ContainerCreating   0          30s
autoscale-deployment-demo-9bfc7d8cd-6stvx   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-7cqqd   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-7f4wn   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-7n5hm   0/1     Pending             0          26s
autoscale-deployment-demo-9bfc7d8cd-7nwjf   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-8qxwz   0/1     ContainerCreating   0          29s
autoscale-deployment-demo-9bfc7d8cd-8rfwb   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-8wt6x   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-bt5n4   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-bxvfs   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-c67t4   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-c7r67   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-cf45v   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-cv6kh   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-cwrp4   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-djw52   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-dm8q9   1/1     Running             0          30s
autoscale-deployment-demo-9bfc7d8cd-dxmwt   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-f7nln   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-fhtls   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-flplc   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-fpzz6   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-g8z7h   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-gmzt6   0/1     Pending             0          26s
autoscale-deployment-demo-9bfc7d8cd-gv7j9   0/1     Pending             0          26s
autoscale-deployment-demo-9bfc7d8cd-gxbhn   1/1     Running             0          30s
autoscale-deployment-demo-9bfc7d8cd-h5hsb   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-h96mg   0/1     Pending             0          26s
autoscale-deployment-demo-9bfc7d8cd-hqwbm   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-hrfpp   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-htx9s   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-jjfn2   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-jtkzv   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-kdgj6   0/1     Pending             0          28s
autoscale-deployment-demo-9bfc7d8cd-kg8ml   0/1     Pending             0          26s
autoscale-deployment-demo-9bfc7d8cd-kp29t   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-ksg8r   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-l4mdp   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-l5887   1/1     Running             0          29s
autoscale-deployment-demo-9bfc7d8cd-lznj6   0/1     Pending             0          27s
autoscale-deployment-demo-9bfc7d8cd-mddwb   0/1     Pending             0          27s
```

When traffic increases EKS cluster scaled up:



```
(base) [redacted]EKS-autoscaler-demo$ kubectl get nodes -w
NAME                                           STATUS   ROLES    AGE   VERSION
ip-192-168-16-27.us-east-2.compute.internal    Ready    <none>   47m   v1.25.13-eks-43840fb
ip-192-168-187-95.us-east-2.compute.internal   Ready    <none>   47m   v1.25.13-eks-43840fb
ip-192-168-59-27.us-east-2.compute.internal    Ready    <none>   61m   v1.25.13-eks-43840fb
ip-192-168-70-219.us-east-2.compute.internal   Ready    <none>   61m   v1.25.13-eks-43840fb
ip-192-168-16-27.us-east-2.compute.internal    Ready    <none>   48m   v1.25.13-eks-43840fb
ip-192-168-187-95.us-east-2.compute.internal   Ready    <none>   48m   v1.25.13-eks-43840fb
ip-192-168-59-27.us-east-2.compute.internal    Ready    <none>   62m   v1.25.13-eks-43840fb
^C(base) [redacted]/EKS-autoscaler-demo$ date
Sunday 29 October 2023 06:24:21 PM IST
(base) [redacted]/EKS-autoscaler-demo$ 
```

When traffic decreased EKS cluster scaled down:



```
Sunday 29 October 2023 06:24:21 PM IST
(base) [redacted]/EKS-autoscaler-demo$ kubectl get nodes -w
NAME                                          STATUS   ROLES    AGE   VERSION
ip-192-168-16-27.us-east-2.compute.internal   Ready    <none>   53m   v1.25.13-eks-43840fb
ip-192-168-187-95.us-east-2.compute.internal  Ready    <none>   53m   v1.25.13-eks-43840fb
ip-192-168-59-27.us-east-2.compute.internal   Ready    <none>   67m   v1.25.13-eks-43840fb
ip-192-168-70-219.us-east-2.compute.internal  Ready    <none>   67m   v1.25.13-eks-43840fb
```

After demo:
##eksctl delete cluster --name clusteratoscales --region us-east-2