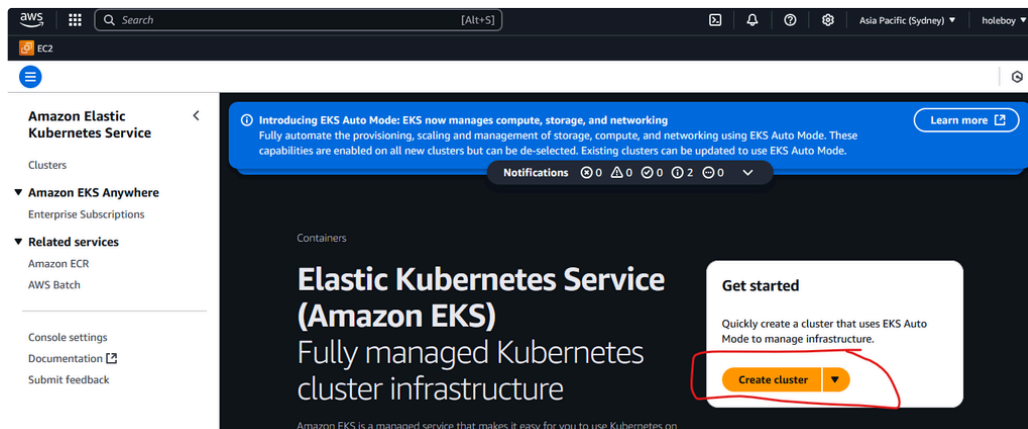


AWS Training: Basic AWS EKS deployment with NGINX

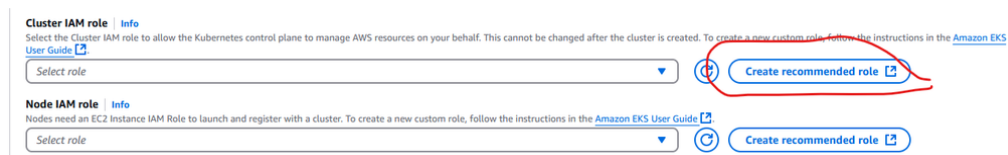
- Create Kubernetes cluster
 - Create cluster IAM role
 - Create node IAM role
 - Create a cluster
- Create kubeconfig
- NGINX manifest (ClusterIP)
- NGINX manifest (LoadBalancer)
 - Tag subnets if needed
- NGINX Config

Create Kubernetes cluster [↗](#)

Create cluster



Create cluster IAM role [↗](#)



EKS

Choose a use case for the specified service.

Use case

- ☐ **EKS - Service**
Allows EKS to manage clusters on your behalf.
- ☐ **EKS - Cluster**
Allows the cluster Kubernetes control plane to manage AWS resources on your behalf.
- ☐ **EKS - Nodegroup**
Allow EKS to manage nodegroups on your behalf.
- ☐ **EKS - Fargate pod**
Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.
- ☐ **EKS - Fargate profile**
Allows EKS to run Fargate tasks.
- ☐ **EKS - Connector**
Allows access to other AWS service resources that are required to connect to external clusters
- ☐ **EKS Local - Outpost**
Allows Amazon EKS Local to call AWS services on your behalf.
- ☐ **EKS - Pod Identity**
Allows pods running in Amazon EKS cluster to access AWS resources.
- ☒ **EKS - Auto Cluster**
Allows access to other AWS service resources that are required to operate Auto Mode clusters managed by EKS.

Permissions policies (5) [Info](#)

The type of role that you selected requires the following policy.

Policy name	Type
AmazonEKSBLOCKStoragePolicy	AWS managed
AmazonEKSClusterPolicy	AWS managed
AmazonEKSComputePolicy	AWS managed
AmazonEKSLoadBalancingPolicy	AWS managed
AmazonEKSNetworkingPolicy	AWS managed

Set permissions boundary - optional

Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting but can be used to delegate permission management to others. [Learn more about permission boundaries](#)

- ☒ Create role without a permissions boundary
- ☐ Use a permissions boundary to control the maximum role permissions

Use defaults if you don't know what you're doing

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and "+, @, _" characters.

Description
Add a short explanation for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: "-+.,@/[]!#\$%^&*(){}~:|;`"'

Step 1: Select trusted entities [Edit](#)

Create node IAM role [🔗](#)

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](#).

[Create recommended role](#)

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](#).

[Create recommended role](#)

EKS

Choose a use case for the specified service.

Use case

- ☐ EKS - Service
Allows EKS to manage clusters on your behalf.
- ☐ EKS - Cluster
Allows the cluster Kubernetes control plane to manage AWS resources on your behalf.
- ☐ EKS - Nodegroup
Allow EKS to manage nodegroups on your behalf.
- ☐ EKS - Fargate pod
Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.
- ☐ EKS - Fargate profile
Allows EKS to run Fargate tasks.
- ☐ EKS - Connector
Allows access to other AWS service resources that are required to connect to external clusters
- ☐ EKS Local - Outpost
Allows Amazon EKS Local to call AWS services on your behalf.
- ☐ EKS - Pod Identity
Allows pods running in Amazon EKS cluster to access AWS resources.
- ☐ EKS - Auto Cluster
Allows access to other AWS service resources that are required to operate Auto Mode clusters managed by EKS.
- ☒ EKS - Auto Node
Allows EKS nodes to connect to EKS Auto Mode clusters and to pull container images from ECR.

- Step 1
Select trusted entity
- Step 2
Add permissions
- Step 3
Name, review, and create

Add permissions [Info](#)Permissions policies (2) [Info](#)

The type of role that you selected requires the following policy.

Policy name	Type
AmazonEC2ContainerRegistryPullOnly	AWS managed
AmazonEKSWorkerNodeMinimalPolicy	AWS managed

[▶ Set permissions boundary - optional](#)

Cancel

Previous

Next

- Step 1
Select trusted entity
- Step 2
Add permissions
- Step 3
Name, review, and create

Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

AmazonEKSAutoNodeRole
Maximum 64 characters. Use alphanumeric and "+", "@", "-", "." characters.

Description

Add a short explanation for this role.

Allows EKS nodes to connect to EKS Auto Mode clusters and to pull container images from ECR.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: "_", "+", "@", "-", ".", ":", "=".

Step 1: Select trusted entities

[Edit](#)Create a cluster [🔗](#)Kubernetes version [Info](#)

Select Kubernetes version for this cluster.

1.31

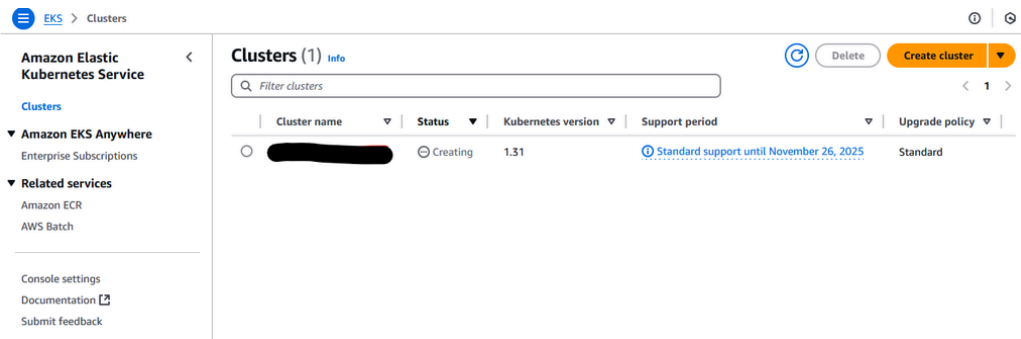
Cluster IAM role [Info](#)Select the Cluster IAM role to allow the Kubernetes control plane to manage AWS resources on your behalf. This cannot be changed after the cluster is created. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

AmazonEKSAutoClusterRole

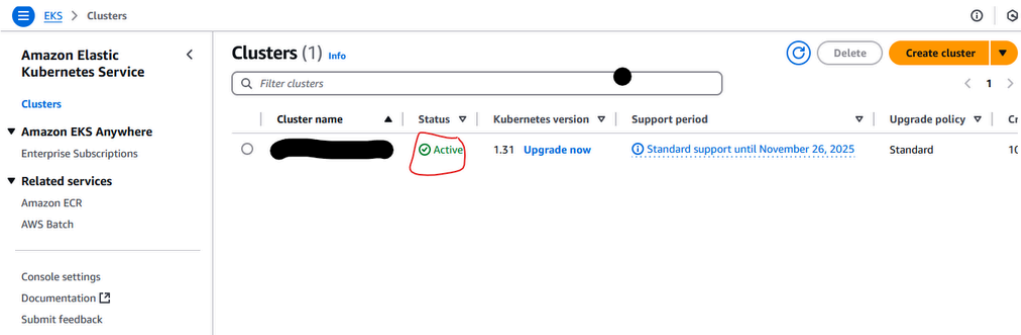
[Create recommended role](#)Node IAM role [Info](#)Nodes need an EC2 Instance IAM Role to launch and register with a cluster. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

AmazonEKSAutoNodeRole

[Create recommended role](#)

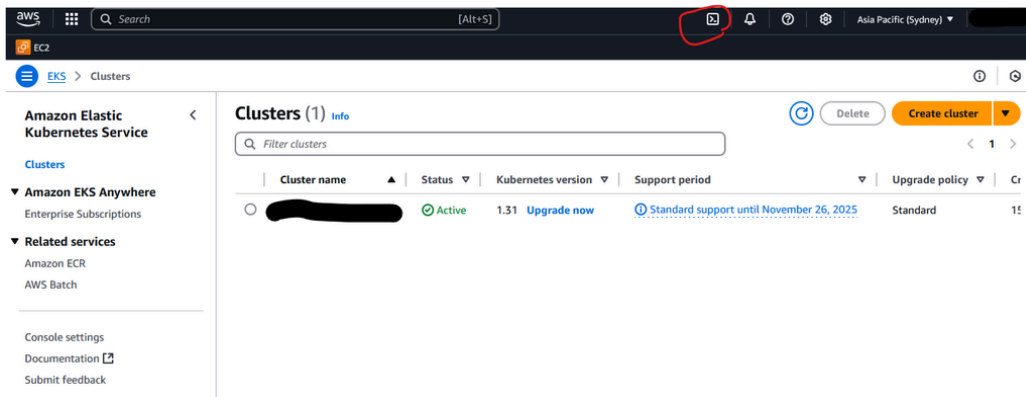


It can take up to 10-15 mins to provision a Kubernetes cluster



Create kubeconfig [↗](#)

Enter AWS cloudshell



Create a kubeconfig file using the AWS CLI

```
1 # Change region-code and my-cluster values
2 aws eks update-kubeconfig --region region-code --name my-cluster
3
4 # Test the connection
5 kubectl get svc
```

NGINX manifest (ClusterIP) [↗](#)

See [Deploy a sample application on Linux - Amazon EKS](#)

Create namespace:

```
1 kubectl create namespace eks-example
```

nginx-deployment.yaml

```
1 cat <<EOF > nginx-deployment.yaml
2 apiVersion: apps/v1
3 kind: Deployment
4 metadata:
5   name: nginx-deployment
6   namespace: eks-example
7   labels:
8     app: nginx
9 spec:
10   replicas: 1
11   selector:
12     matchLabels:
13       app: nginx
14   template:
15     metadata:
16       labels:
17         app: nginx
18     spec:
19       affinity:
20         nodeAffinity:
21           requiredDuringSchedulingIgnoredDuringExecution:
22             nodeSelectorTerms:
23               - matchExpressions:
24                 - key: kubernetes.io/arch
25                   operator: In
26                   values:
27                     - amd64
28                     - arm64
29       containers:
30         - name: nginx
31           image: public.ecr.aws/nginx/nginx:1.23
32           ports:
33             - name: http
34               containerPort: 80
35           imagePullPolicy: IfNotPresent
36       nodeSelector:
37         kubernetes.io/os: linux
38 EOF
39
40 kubectl apply -f nginx-deployment.yaml
41 kubectl get pods -l 'app=nginx' -o wide -n eks-example
```

nginx-service.yaml

```
1 cat <<EOF > nginx-service.yaml
2 apiVersion: v1
3 kind: Service
4 metadata:
5   name: nginx-service
6   namespace: eks-example
7   labels:
8     app: nginx
9 spec:
10   selector:
11     app: nginx
12   ports:
13     - protocol: TCP
```

```

14     port: 80
15     targetPort: 80
16 EOF
17
18 kubectl apply -f nginx-service.yaml
19 kubectl get service/nginx-service -n eks-example
20
21 kubectl -n default describe service nginx-service
22
23 kubectl get pods -l 'app=nginx' -o wide -n eks-example # Get a POD_ID
24 kubectl -n eks-example describe pod nginx-deployment-${POD_ID}
25
26 kubectl get all -n eks-example

```

Enter the pod

```

1 kubectl exec -it nginx-deployment-${POD_ID} -n eks-example -- /bin/bash
2
3 # Inside the pod
4 curl nginx-service
5
6 cat /etc/resolv.conf

```

Clean up everything

```

1 exit # If still inside the pod
2 kubectl delete ns eks-example

```

NGINX manifest (LoadBalancer) [↗](#)

See [Route TCP and UDP traffic with Network Load Balancers - Amazon EKS](#)

Tag subnets if needed [↗](#)

The screenshot shows the Amazon EKS console interface. The breadcrumb navigation at the top reads 'EKS > Clusters > ferocious-funk-hideout'. The left sidebar contains the 'Amazon Elastic Kubernetes Service' logo and a 'Clusters' section with links to 'Amazon EKS Anywhere', 'Enterprise Subscriptions', and 'Related services' (Amazon ECR, AWS Batch). Below these are links for 'Console settings', 'Documentation', and 'Submit feedback'. The main content area has a top bar with 'Cluster health issues' (0), 'Upgrade insights' (4), and 'Node health issues' (0). Below this is a tabbed interface with 'Networking' selected. The 'Networking' section includes a 'VPC' card (with a redacted ID), a 'Cluster IP address family' (IPv4), a 'Service IPv4 range' (10.100.0.0/16), a 'Subnets' card (with three redacted IDs), a 'Cluster security group' (with a redacted ID), and 'Additional security groups' (None). On the right, there are buttons for 'Manage VPC resources' and 'Manage endpoint access', and a section for 'API server endpoint access' (Public and private) and 'Public access source allowlist' (0.0.0.0/0).

The screenshot shows the AWS VPC console interface. The left sidebar contains navigation links for VPC, Subnets, and various VPC resources. The main content area displays the 'Tags' tab for a selected resource. The 'Tags' section is currently empty, showing a message 'No tags associated with this resource' and a 'Manage tags' button. Below the console, a detailed view of the 'Tags' interface is shown, including a search bar, a table with 'Key' and 'Value' columns, and a 'Value - optional' field with the value '1'. The 'Add new tag' button is also visible.

- For public subnets:
 - Key: `kubernetes.io/role/elb`
 - Value: `1`
- For private subnets:
 - Key: `kubernetes.io/role/internal-elb`
 - Value: `1`

NGINX Config [↗](#)

Create namespace:

```
1 kubectl create namespace eks-example
```

nginx-deployment.yaml

```
1 cat <<EOF > nginx-deployment.yaml
2 apiVersion: apps/v1
3 kind: Deployment
4 metadata:
5   name: nginx-deployment
6   namespace: eks-example
7   labels:
8     app: nginx
9 spec:
10   replicas: 1
11   selector:
12     matchLabels:
13       app: nginx
14   template:
15     metadata:
16       labels:
17         app: nginx
18   spec:
19     containers:
20       - name: nginx
21         image: public.ecr.aws/nginx/nginx:1.23
```

```

22     ports:
23     - name: tcp
24       containerPort: 80
25 EOF
26
27 kubectl apply -f nginx-deployment.yaml
28 kubectl get pods -l 'app=nginx' -o wide -n eks-example

```

nginx-service.yaml

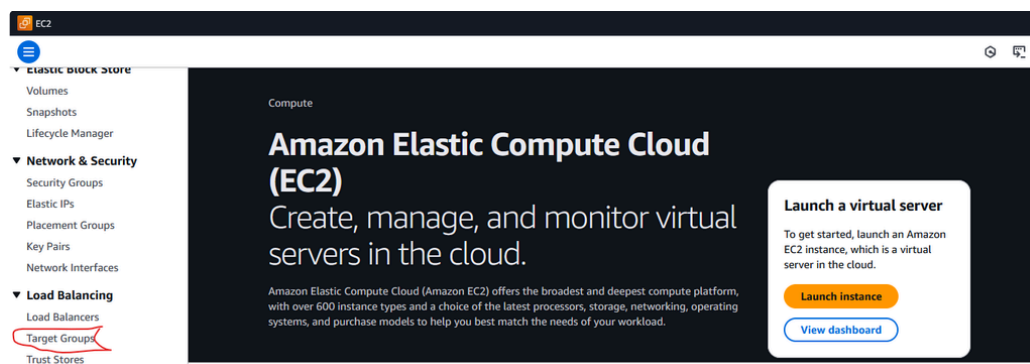
```

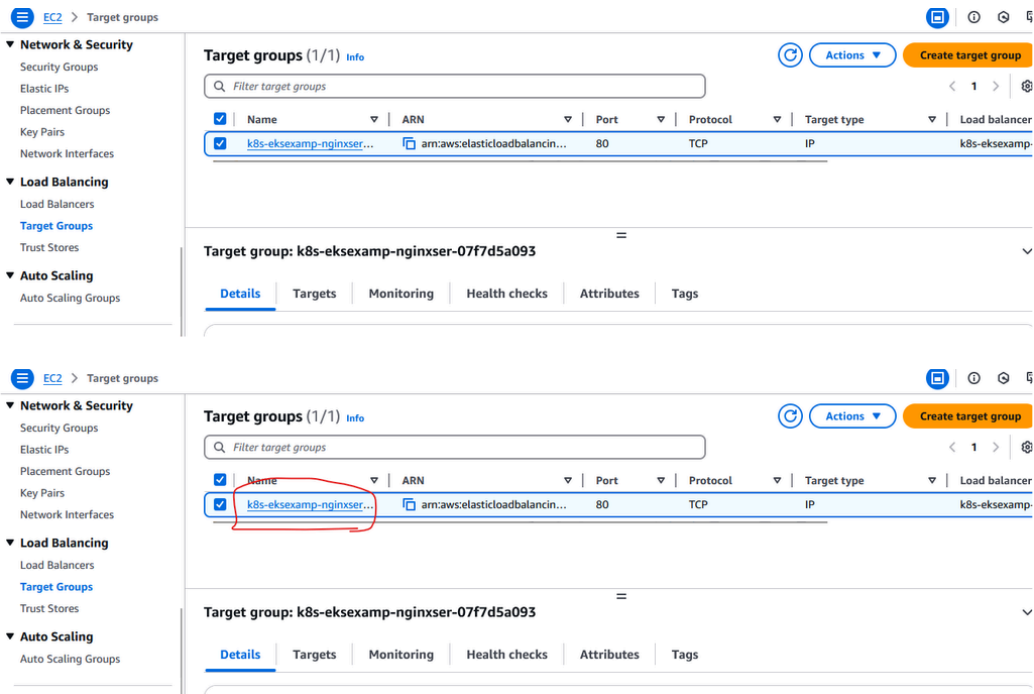
1 cat <<EOF > nginx-service.yaml
2 apiVersion: v1
3 kind: Service
4 metadata:
5   name: nginx-service
6   namespace: eks-example
7   annotations:
8     service.beta.kubernetes.io/aws-load-balancer-type: external
9     service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
10    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
11 spec:
12   ports:
13   - port: 80
14     targetPort: 80
15     protocol: TCP
16   type: LoadBalancer
17   selector:
18     app: nginx
19 EOF
20
21 kubectl apply -f nginx-service.yaml
22 kubectl get all -n eks-example
23 kubectl get svc nginx-service -n eks-example

```

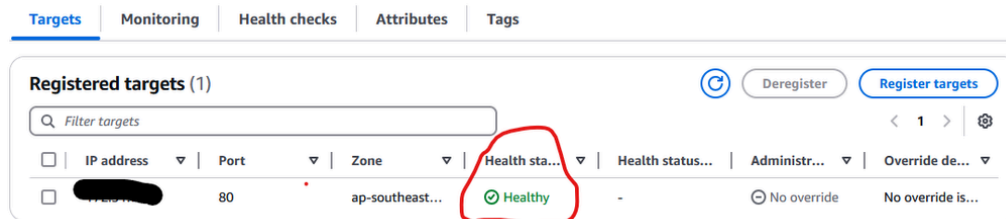
Open EC2 AWS Management console and select Target Groups under Load Balancing:

<https://console.aws.amazon.com/ec2>





Visit Registered targets, and wait until the status is healthy:



Then run the following in the CloudShell console:

```
1 curl k8s-eksexamp-nginxser-xxxxxxxx-xxxxxxxxxxxxxxxx.elb.${REGION_CODE}.amazonaws.com
```

You can also visit the above link in your browser

Clean up everything

```
1 kubectl delete ns eks-example
```

Don't forget to delete your cluster after testing