

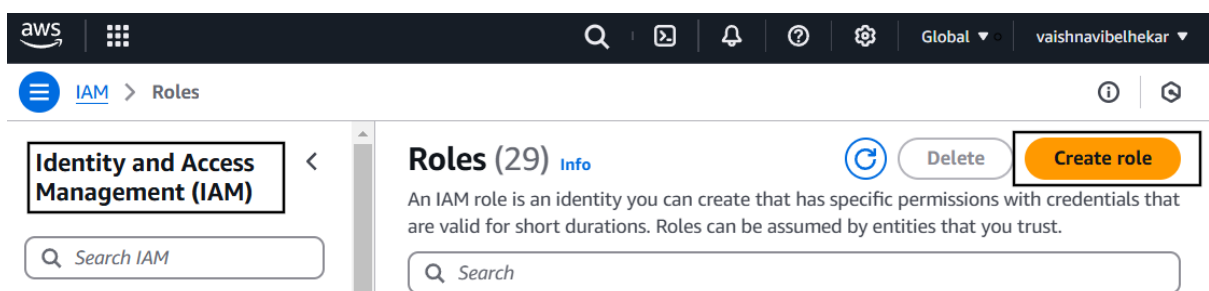
Elastic Kubernetes Service

What is EKS?

- Without using minikube.
- We can deploy image inside pods.
- Why we use prefer EKS because we can integrate it with other services like IAM.
- We can scale up and scale down using auto scaling.

Steps—

--> Search EKS--> Add cluster--> create --> provide name--> provide version(one before latest)--> duplicate Tab --> go in the IAM --> Roles--> create role--> usecase as EKS --> and select EKS cluster--> next--> provide role name--> create--> back to eks--> Refresh the role and select the role-->select next -->next-->next--> create.



Steps—



Azure DevOps

rushikeshkarpe03@gmail.com [Switch directory](#)

Almost done...

Name your Azure DevOps organization *

dev.azure.com/ rushikeshkarpe030255

We'll host your projects in

India

Enter the characters you see

New Audio



JS4LX

Continue

create a new role in IAM Service--> under usecase select ec2--> search CNI policies-->select 'amazon eks CNI policies'--> Search EC2 container and select 'amazon Ec2 container registry readonly'--> search worker node and select 'amazon EKS worker node policy'--> Give Role Name as workernodepolicyRole --> create role.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

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.

Cluster configuration

Name

Use the auto-generated name or enter a unique name for this cluster. This property cannot be changed after the cl

cluster-1



The cluster name should begin with letter or digit and can have any of the following characters: the set of Unicode underscores. Maximum length of 100.

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 ca is created. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

Role141814



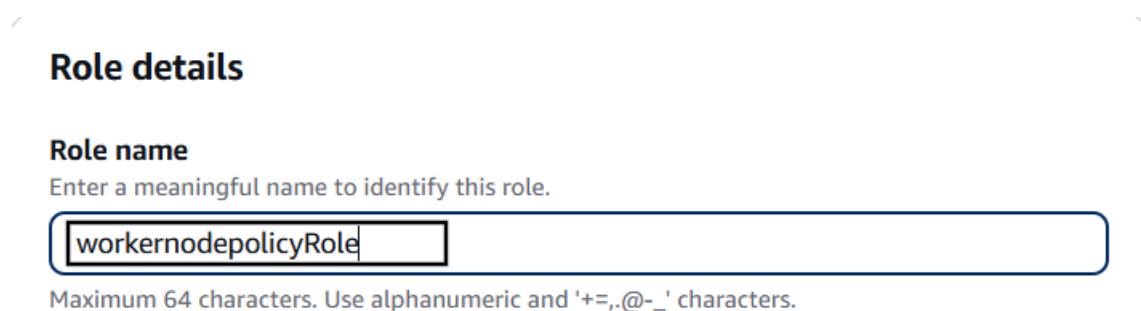
--> now we have to download aws cli so search AWS cli on google, copy first command and paste it in CMD.--> setup will open.

Steps—

-->When the Status becomes active--> go in compute tab for creating node group--> Select add node group--> give name myNodeGroup--> select worker node policy role--> next--> select instance type as t2.medium //skip t3.medium because charges are high--> next--> next-->create

--> under the node group desire size is 2--> therefore two nodes will be created above it-->

--> close the command prompt and open new CMD and check aws version using 'aws --version' command-->type 'aws configure' and type access key and secrete access key--> region as ap-south-1 --> output format as skip-->



Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+=,.,@-_' characters.

You can access the Access key and Secret Access key from the Security Credentials of the AWS Console. Download the Root key file for further use.

Provide the region name as = ap-south-1

Provide the output format= json

```
Command Prompt
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Vaishnavi>aws --version
aws-cli/2.23.6 Python/3.12.6 Windows/11 exe/AMD64

C:\Users\Vaishnavi>aws configure
AWS Access Key ID [*****BPS]: AKIA6GBMD5GMRG652BPS
AWS Secret Access Key [*****xJ9]: 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
Default region name [ap-south-1]: ap-south-1
Default output format [json]: json
```

Type the commands—

--> type aws and hit enter--> type kubectl --> Now we have to connect AWS cli to cluster

--> aws eks --region ap-south-1 update-kubeconfig --name myf13

--> kubectl get nodes

--> kubectl get svc

--> kubectl get pods

--> kubectl get ns

--> Now we have to create pod and bind it to port

--> kubectl run pod1 --image=nginx [to deploy image on pods]

--> kubectl get pods

--> kubectl expose pod pod1 --type=NodePort --port=80 --name=my-first-service

--> kubectl get svc

--> kubectl get pods -o wide

--> to check on which node is there--> go to console -->ec2--> and first and second Id to check private IP instance ID--> SO which instance's private Ip is matched copy the public IP of that instance and paste it in new Tab--> error is showing --> so check port number in command prompt and type in front of IP after colon --> still error then go to security group of instance and add inbound rule--> add rule --> All traffic--> CIDR block as 0000 --> save --> now refresh the page --> Ngnix will show there.

--> If you want one more image so create one more pod--> we can run multiple container in single pod but its not recommended because they are tightly coupled-->

-->for that 'kubectl run pod2 --image=nginx' repeat from this command

#Closing--> close command promt--> delete Node group--> delete autoscaling group --> delete cluster.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.
