

<https://medium.com/avmconsulting-blog/deploying-a-kubernetes-cluster-with-amazon-eks-9455e7e7828>

<https://docs.aws.amazon.com/eks/latest/userguide/getting-started-console.html>

[After installation Test](#)

[aws eks list-clusters](#)

## Step 0: Before you start

You will need to make sure you have the following components installed and set up before you start with Amazon EKS:

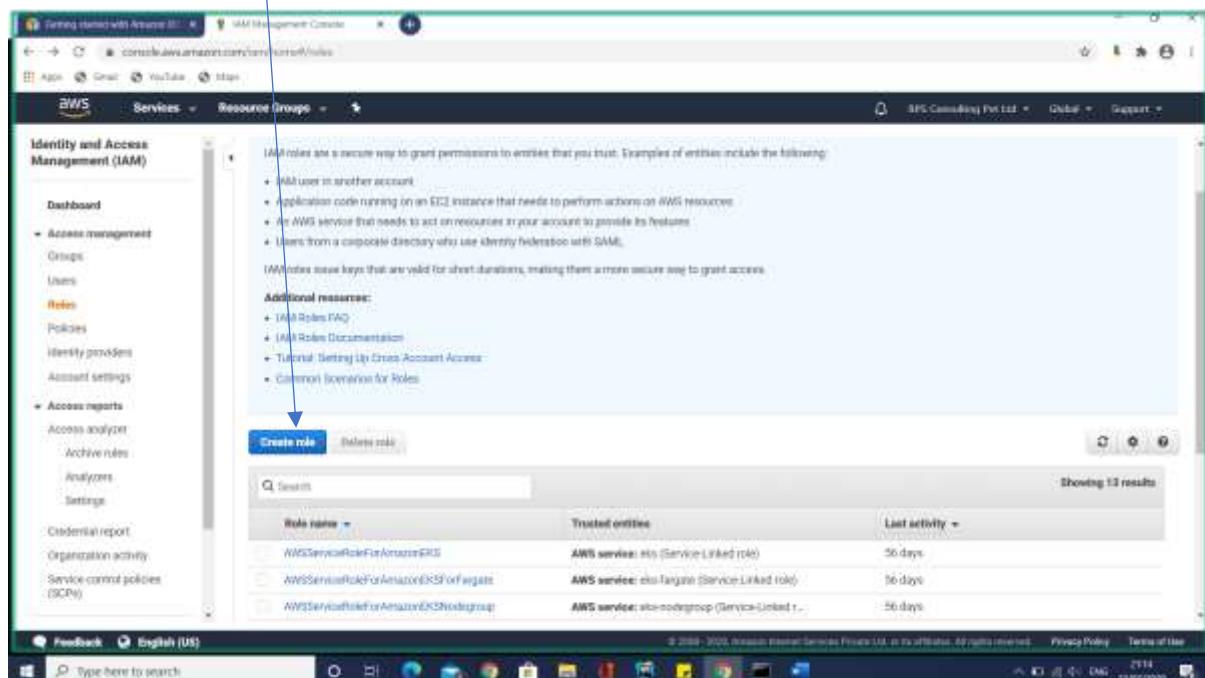
- AWS CLI – while you can use the AWS Console to create a cluster in EKS, the AWS CLI is easier. You will need version 1.16.73 at least.
- Kubectl – used for communicating with the cluster API server.. This endpoint is public by default, but is secured by proper configuration of a VPC.
- AWS-IAM-Authenticator – to allow IAM authentication with the Kubernetes cluster.

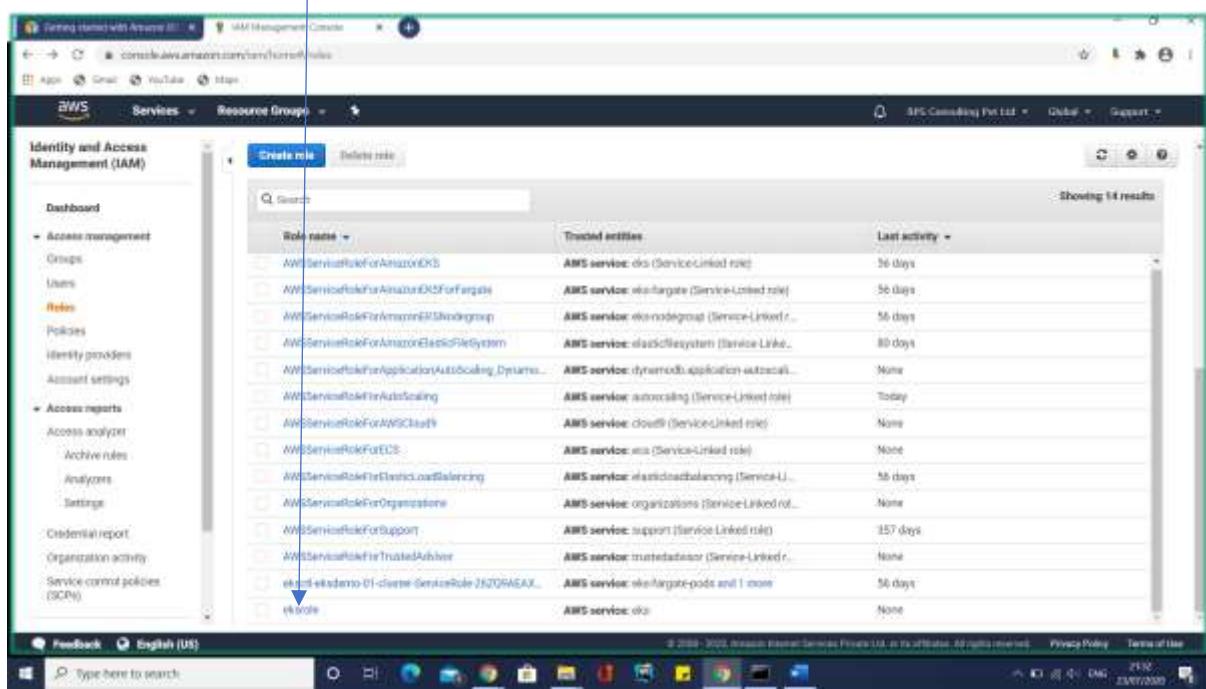
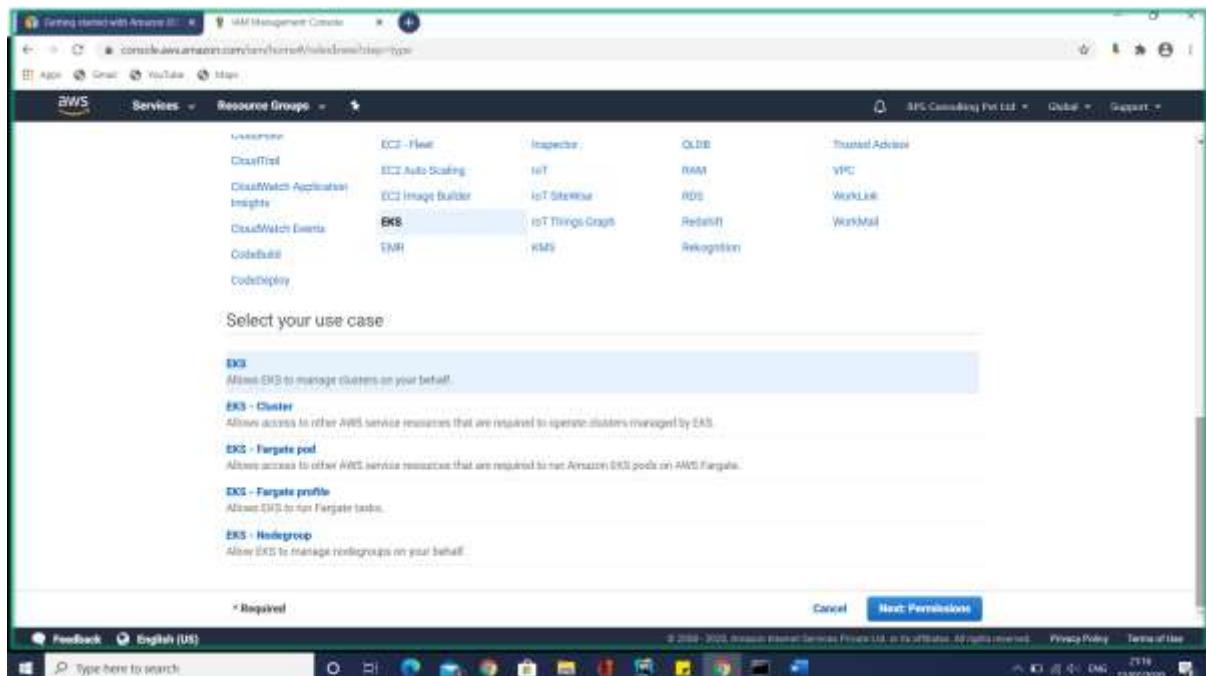
# Step 1: Creating an EKS role

Our first step is to set up a new IAM role with EKS permissions.

Open the [IAM console](#), select **Roles** on the left and then click the **Create Role** button at the top of the page.

From the list of AWS services, select **EKS** and then **Next: Permissions** at the bottom of the page.





The screenshot shows the AWS IAM Management Console with the 'eksrole' role selected. The left sidebar shows the navigation menu for Identity and Access Management (IAM). The main content area displays the 'Summary' tab for the 'eksrole'. Key details shown include:

- Role ARN:** arn:aws:iam::567749627506:role/eksrole
- Role description:** Allows access to other AWS service resources that are required to operate clusters managed by EKS.
- Instance Profile ARNs:** (empty)
- Path:** /
- Creation time:** 2020-07-22 21:32 UTC+0330
- Last activity:** Not accessed in the tracking period
- Maximum CLI/API session duration:** 1 hour

The 'Permissions' tab is active, showing the attached policy:

- Policy name: AmazonEKSContainerPolicy
- Policy type: AWS managed policy

## Role ARN

arn:aws:iam::567749627506:role/eksrole

arn:aws:iam::256544469827:role/eksrole

## Step 2: Creating a VPC for EKS

Next, we're going to create a separate VPC—a Virtual Private Cloud that protects communication between worker nodes and the AWS Kubernetes API server—for our EKS cluster. To do this, we're going to use a CloudFormation template that contains all the necessary EKS-specific ingredients for setting up the VPC.

Open up [CloudFormation](#), and click the **Create new stack** button.

On the **Select template** page, enter the URL of the CloudFormation YAML in the relevant section:

<https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2019-01-09/amazon-eks-vpc-sample.yaml>

## Search in Service Cloud Formation

The screenshot shows the AWS CloudFormation console with the URL <https://ap-south-1.console.aws.amazon.com/cloudformation/stacks?region=ap-south-1&stackFilteringText=&filterStatuses=active&sortOrder=lastUpdated>. The page displays a list of 7 stacks:

Stack name	Status	Created time	Description
lamp	CREATE_COMPLETE	2018-09-14 22:24:54 UTC+0530	AWS CloudFormation Sample Template LAMP_Single_Instance: Create a LAMP ...
rajesh6	ROLLBACK_COMPLETE	2018-08-30 16:41:27 UTC+0530	Demo template to create an S3 Bucket
Stack1-53	ROLLBACK_COMPLETE	2018-08-30 15:35:25 UTC+0530	Demo template to create an S3 Bucket
user1-stack	CREATE_COMPLETE	2018-08-23 16:11:37 UTC+0530	-
user5-chipoc2	CREATE_COMPLETE	2018-08-23 15:23:44 UTC+0530	VPC Creation
UserCreation	CREATE_COMPLETE	2018-08-23 11:12:34 UTC+0530	IAM User Creation
Ep21-STACK	DELETE_FAILED	2018-07-18 14:01:56 UTC+0530	Substack AWS CloudFormation

The screenshot shows the AWS CloudFormation console with the URL <https://ap-south-1.console.aws.amazon.com/cloudformation/stacks/rpstack?region=ap-south-1&stackFilteringText=&filterStatuses=active&sortOrder=lastUpdated>. The page displays the events for the rpstack stack:

Timestamp	Logical ID	Status	Status reason
2019-07-23 21:57:50 UTC+0530	rpstack	CREATE_COMPLETE	-
2020-07-23 21:57:49 UTC+0530	Submit01RouteTableAssociation	CREATE_COMPLETE	-
2020-07-23 21:57:49 UTC+0530	Submit02RouteTableAssociation	CREATE_COMPLETE	-
2020-07-23 21:57:48 UTC+0530	Route	CREATE_COMPLETE	-
2020-07-23 21:57:54 UTC+0530	Submit01RouteTableAssociation	CREATE_IN_PROGRESS	Resource creation initiated
2020-07-23 21:57:55 UTC+0530	Submit02RouteTableAssociation	CREATE_IN_PROGRESS	Resource creation initiated
2020-07-23 21:57:55 UTC+0530	Submit01RouteTableAssociation	CREATE_IN_PROGRESS	-
2020-07-23 21:57:55 UTC+0530	Submit02RouteTableAssociation	CREATE_IN_PROGRESS	-

Screenshot of the AWS CloudFormation console showing the 'Stacks' page. The left sidebar lists several stacks, including 'rpsstack' (Active, CREATE\_COMPLETE), 'lamb', 'rpsnet', 'StackT-S5', and 'user21-stack'. The main panel displays the 'rpsstack' stack details:

Key	Value	Description
SecurityGroups	sg-05ae18c1d8f9a94a	Security group for the cluster control plane communication with worker nodes
SubnetIds	subnet-09a04605a32e661f, subnet-0x0fb5226dc2a4ef	All subnets in the VPC
VpcId	vpc-0a2eb04f6aff51c1	The VPC Id

Screenshot of the AWS CloudFormation console showing the 'Outputs' tab for the 'rpsstack' stack. The table lists three outputs:

Key	Value	Description
SecurityGroups	sg-05ae18c1d8f9a94a	Security group for the cluster control plane communication with worker nodes
SubnetIds	subnet-09a04605a32e661f, subnet-0x0fb5226dc2a4ef	All subnets in the VPC
VpcId	vpc-0a2eb04f6aff51c1	The VPC Id

SecurityGroups	sg-05ae18c1d8f3ba94a	Security group for the cluster contr
SubnetIds	subnet-09a4c4803a32e661f,subnet-0c0fb522fcdc2c4ef	All subnets in the VPC
VpcId	vpc-0a2eeb04f6af651c1	The VPC Id

## Step 3: Creating the EKS cluster

[Navigate to EKS cluster](#)

**Elastic Kubernetes Service (Amazon EKS)**  
Fully managed Kubernetes control plane

Amazon EKS is a managed service that makes it easy for you to use Kubernetes on AWS without needing to install and operate your own Kubernetes control plane.

**Create EKS cluster**

Cluster name:

**Pricing**

EKS Control Plane: [EKS Pricing](#)  
Worker nodes: [EC2 Pricing](#)

**Getting started**

For more details, see the [Amazon EKS product page](#).  
For a walkthrough of deploying an EKS cluster, see the [AWS Documentation](#).

## Configure Cluster

**Step 1: Configure cluster**

**Step 2: Specify networking**

**Step 3: Configure logging**

**Step 4: Review and create**

**Configure cluster**

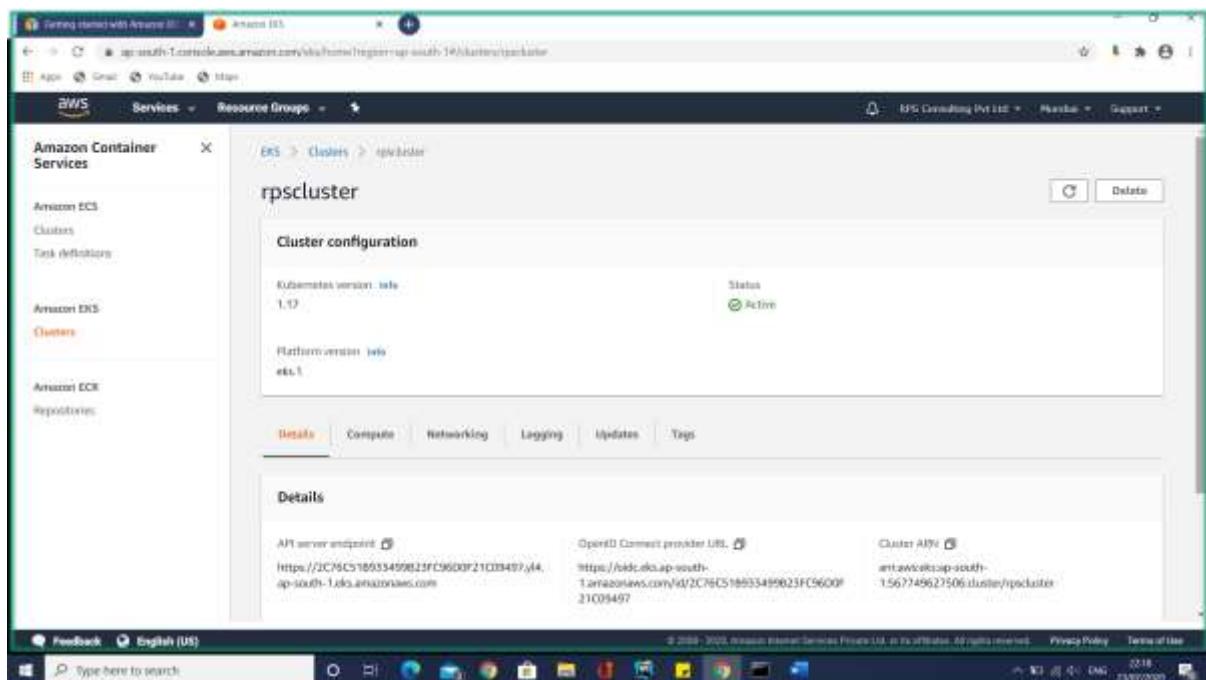
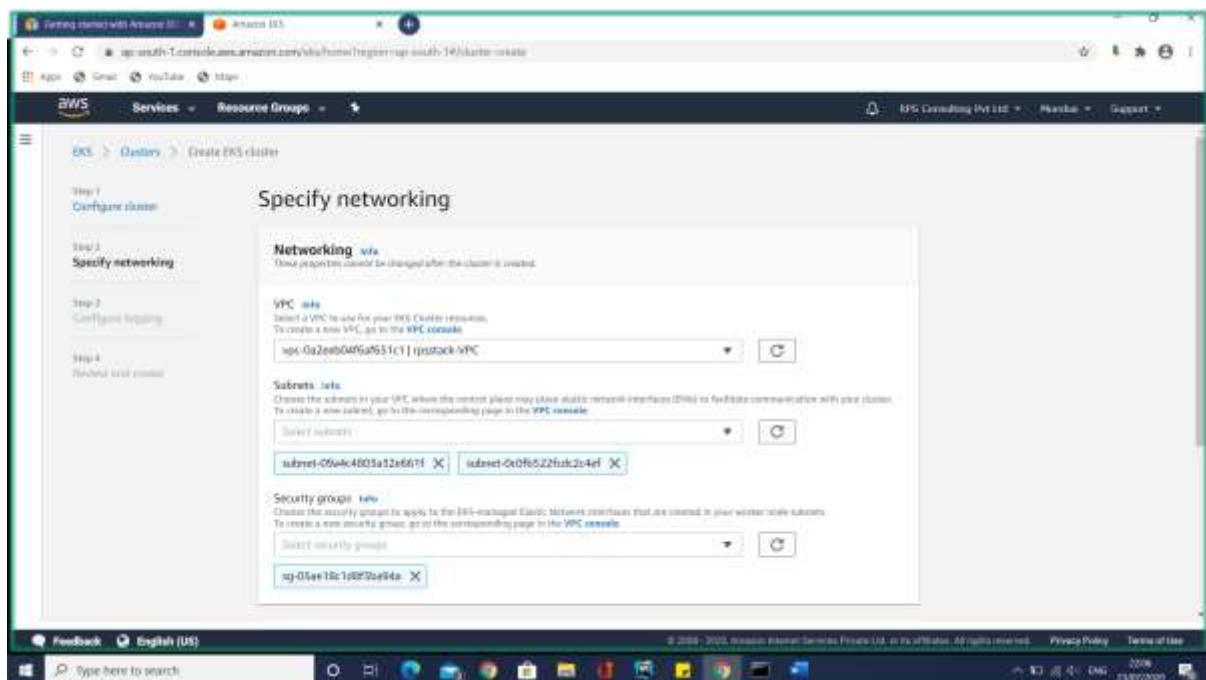
**Cluster configuration**

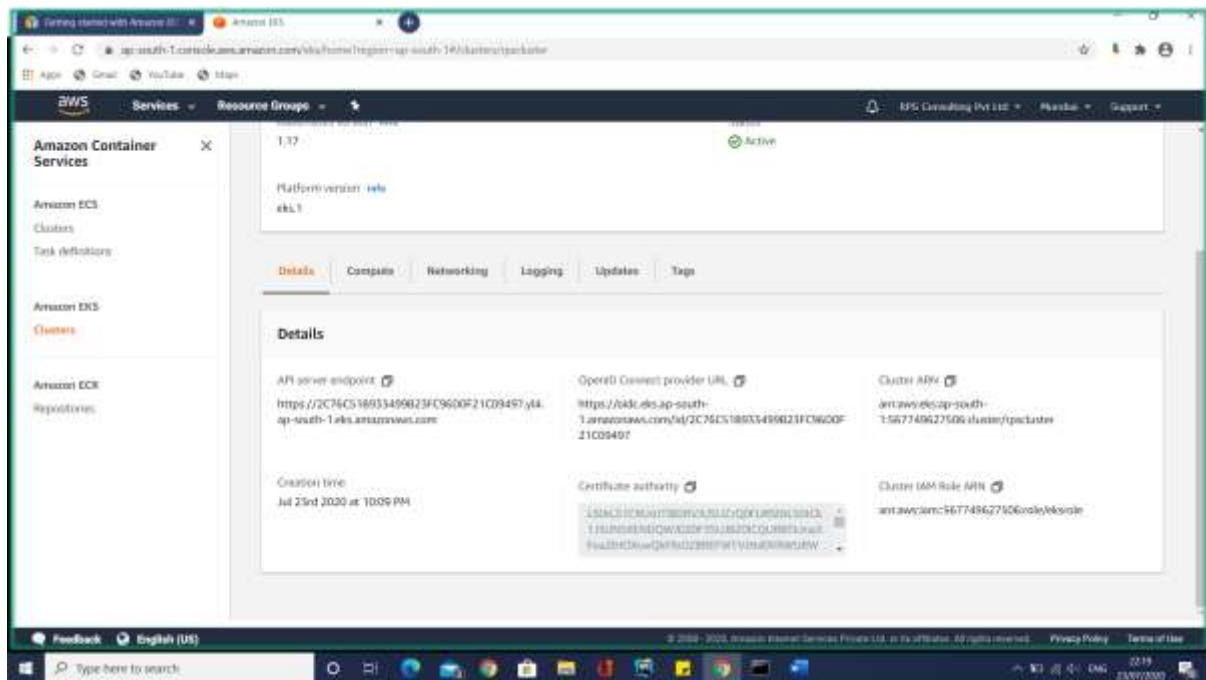
Name:  ipcluster

Kubernetes version:  1.17

Cluster Service Role:  eks - test default after creation

Secrets encryption:  Enable envelope encryption of Kubernetes secrets using KMS





## API server End point

<https://2C76C51B933499B23FC96D0F21C09497.y14.ap-south-1.eks.amazonaws.com>

Open Id Connect URL

<https://oidc.eks.ap-south-1.amazonaws.com/id/2C76C51B933499B23FC96D0F21C09497>

## Certificate Authority

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0V  
RWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJd01EY3lnekUyTkRZd01sb1hEVE13TURjeU1URTJORFI3TWxvd0ZURVRNQk  
VHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVIKS29aSWh2Y05BUUVCQIFBRGdnRVBBRENDQVFvQ2dnRUJBSy9aCkN2d  
TRCTThIRS9iekZiYnZLTlpIeDNXVnJGUZN0cTRDd3BuanlsU1ExekNsZWZhbtI4empCY1k5OUZ1K2JKQ2wKdVp3QjJxeXV4Ymd0d00ycVR  
PUVorUmZoR3V2ZGVYMGcyNWtVMmRBmjU2akZWWWYxNjV2Mm50YWVleS9Vd1N6dQo3U3VpZ3RsZExhb3ZsbHRYbEJPaEtUN1dtRH  
dHM1RKSnsZESEJFaWlJWk8zdW5HRhdIRXZKQTZiaG1Rd2FEbTzUClB3UDVwZjZKdUhthk9ZRE1EOFBKQmV6T2UrZmUwWDFUbW5tV  
ExUd2tEY3ZCU1lnMVN5MndCcU1xR0tjVS9xYzIKWXRkM0diZ0dFT2hMNHZ6RUpHSXdkRmFHZmcvEHsWEhBbWdCVnZrQloyOY2M  
W1laFZVV3d2ZDg1bzQvRUVBaQpTb2JJZGpjV0RXbXg4UGwzb0trQ0F3RUFBYU1qTUNFd0RnWURWUjBQQVFIL0JBUURBZ0trTUE4R0E  
xVWRFd0VCCi93UUZNQU1CQWY4d0RRWUpLb1pJaHZjTkFRRUxCUUF EZ2dFQkFFWWlnMIBiUIJQK0M2S05GeUdzM05sUDFqeGgKbzF  
XWWxRYWEvT25CbUZ6OWtMU05HTml3VDdFWnptS09RUDVteVZiUnljN3JqMFpja0NLTTBQbk1qTGjmK0tHzwpOWHkwNHcwOGpoTm  
ZoNFdvOTJMcoItWIFSKzvUEd3dG9MQ3AxWDRIK0xHYmo3THM2aXpSV0FyVGk0Y2dQbTIIClZhb1pQmk9IRndHSjc1RIZ2Q2dJSUVweV  
Z0bStHUVNNR1Qza01MVm1Tdml6dnlpGrFMnlhYU4wYzgvUzExbTMKd1ozc20vQ3JybjNWNkg1VVdsTEcxNjFTSy90bFE2NjVucklEdjFTZ

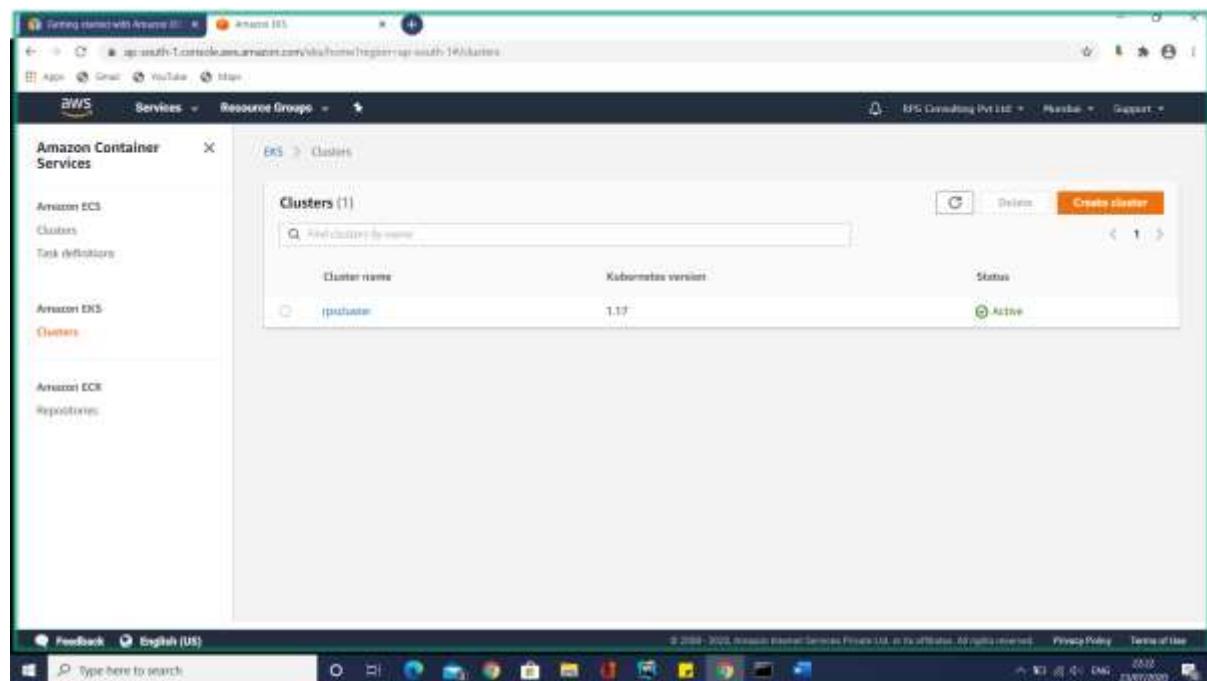
EJOTHpqckdoL0V0OHB6dmxERU84UUFnMwpmNDBNSUZ0TVN0emxjRk1PVndwTXAzV0M2VE1uQ3NudEM2VjVtYIM3QUxRQkZxSmw3  
N3BvR1oyWTNORT0KLS0tLS1FTkQgQ0VSVEIGSUNBVEUtLS0tLQo=

## Cluster ARN

arn:aws:eks:ap-south-1:567749627506:cluster/rpscluster

## Iam Role

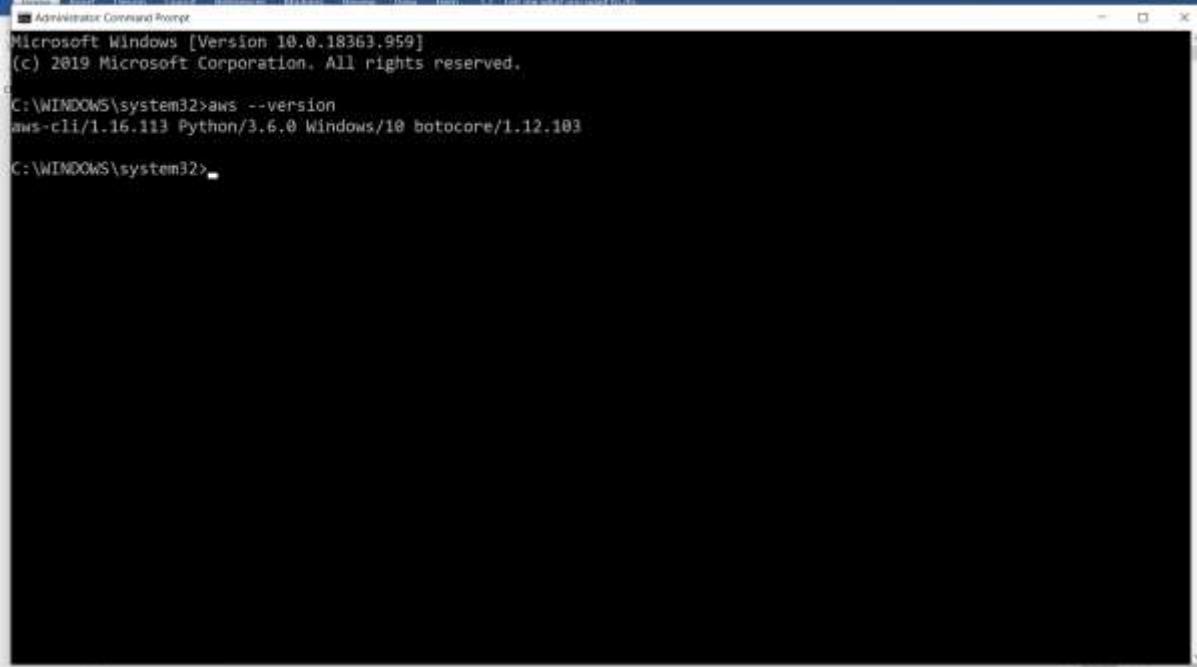
arn:aws:iam::567749627506:role/eksrole



## To install the AWS CLI version 2

If you don't have either version 1.18.97 or later, or version 2.0.30 or later installed, then install the AWS CLI version 2 using the following steps. For other installation options, or to upgrade your currently installed version 2, see [Upgrading the AWS CLI version 2 on Windows](#).

1. Download the AWS CLI MSI installer for Windows (64-bit) at <https://awscli.amazonaws.com/AWSCLIV2.msi>
2. Run the downloaded MSI installer and follow the onscreen instructions. By default, the AWS CLI installs to C:\Program Files\Amazon\AWSCLIV2.



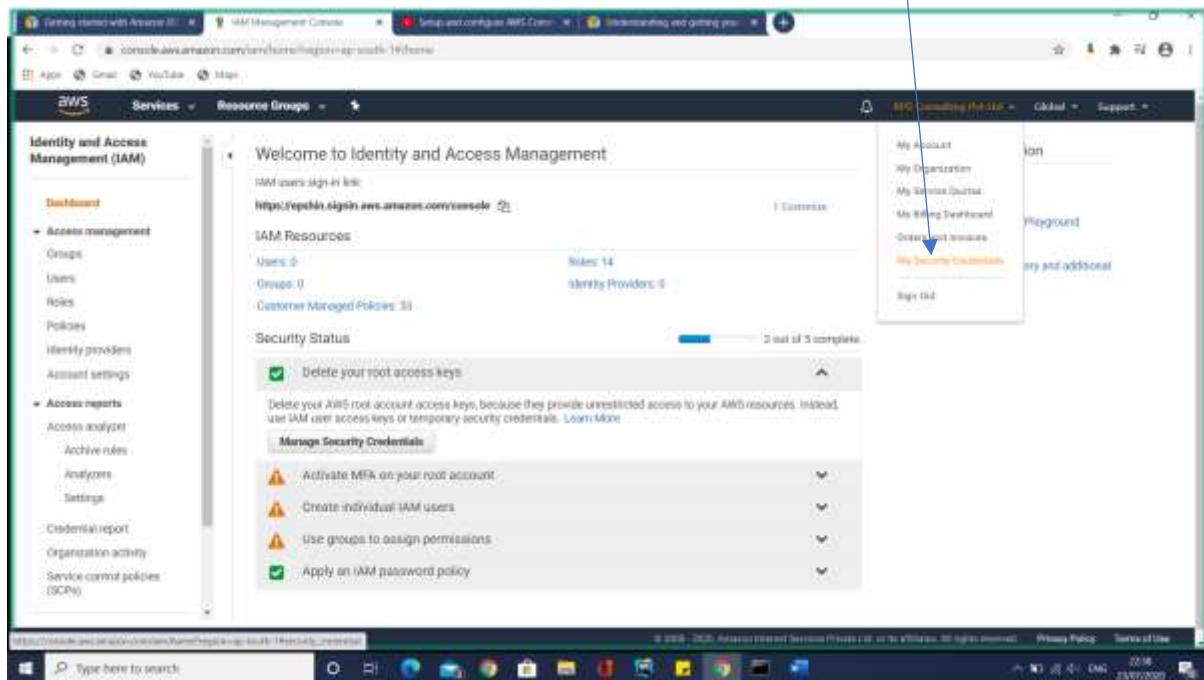
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18363.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>aws --version
aws-cli/1.16.113 Python/3.6.0 Windows/10 botocore/1.12.103

C:\WINDOWS\system32>
```

### To create an access key when signed in as the root user

1. Sign in to the AWS Management Console as the root user. For more information, see [Sign in as the root user](#) in the *IAM User Guide*.
2. In the navigation bar on the upper right, choose your account name or number and then choose **My Security Credentials**.
3. Expand the **Access keys (access key ID and secret access key)** section.
4. Choose **Create New Access Key**. If you already have two access keys, this button is disabled.
5. When prompted, choose **Show Access Key** or **Download Key File**. This is your only opportunity to save your secret access key.
6. After you've saved your secret access key in a secure location, chose **Close**.

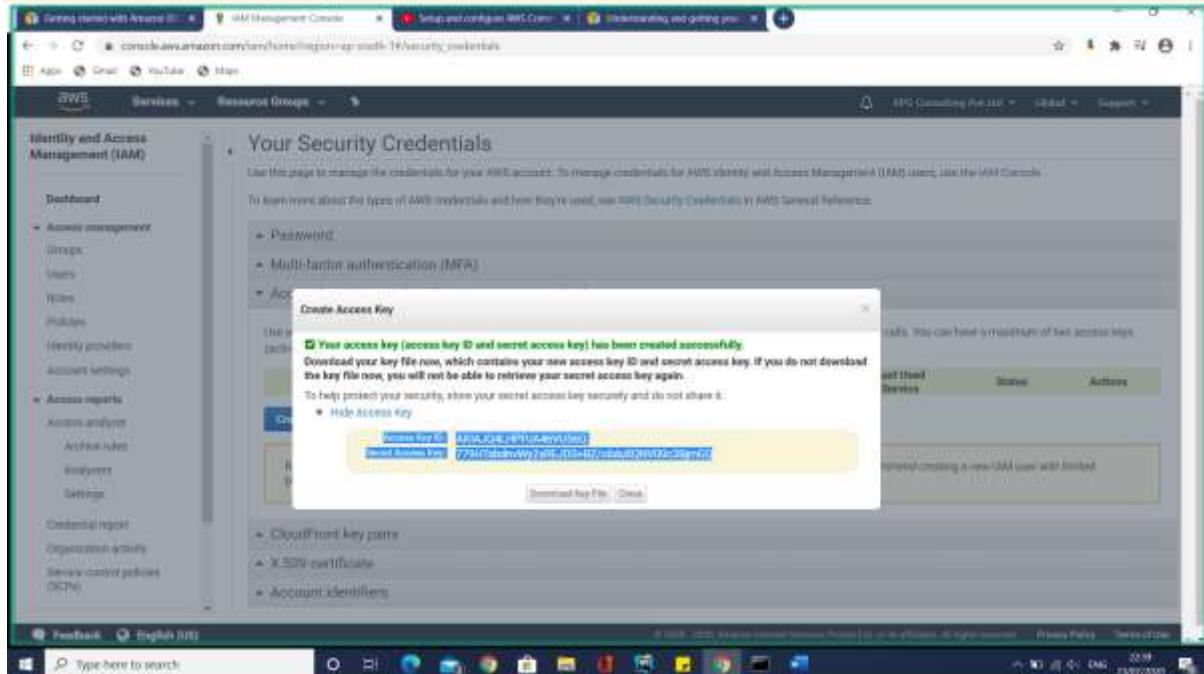


Access Key ID:

AKIAJQ4LHPPJA46VU56Q

Secret Access Key:

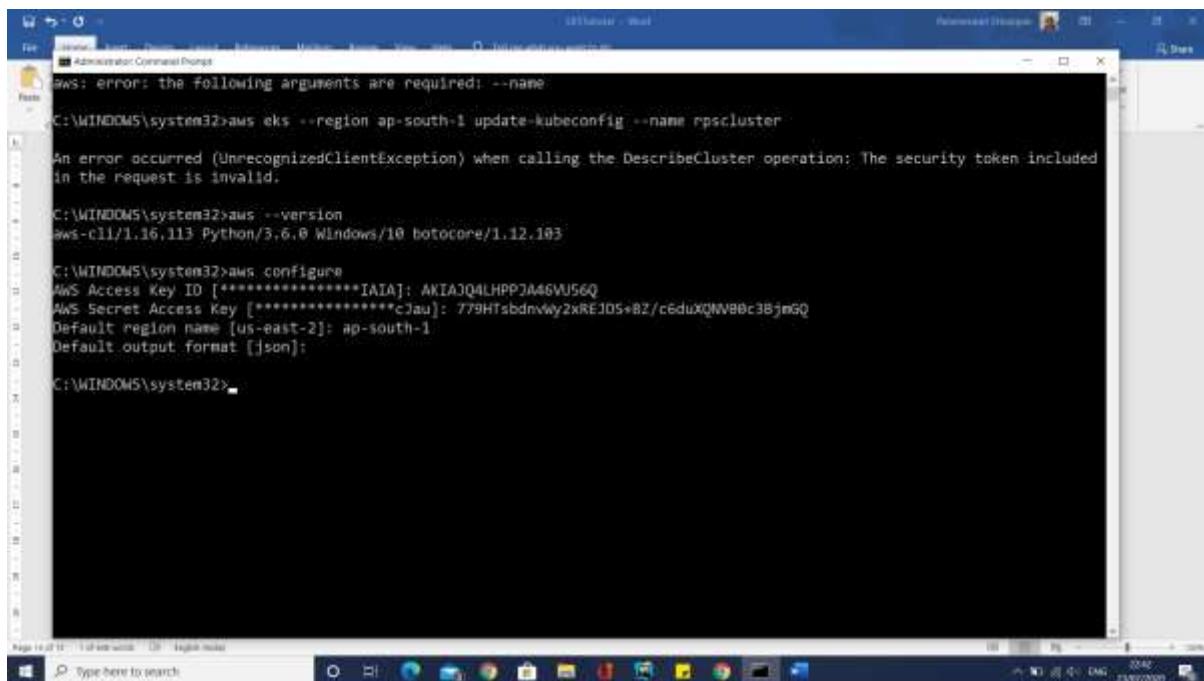
779HTsbdnvWy2xREJDS+BZ/c6duXQNV00c3BjmGQ



## To create an access key when signed in as an IAM user

1. Sign in to the AWS Management Console as an IAM user. For more information, see [Sign in as an IAM user](#) in the *IAM User Guide*.
2. In the navigation bar on the upper right, choose your user name and then choose **My Security Credentials**.
3. Choose **AWS IAM credentials**, **Create access key**. If you already have two access keys, the console displays a "Limited exceeded" error.
4. When prompted, choose **Download .csv file** or **Show secret access key**. This is your only opportunity to save your secret access key.
5. After you've saved your secret access key in a secure location, chose **Close**.

## AWS configure



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The command entered is:

```
aws eks --region ap-south-1 update-kubeconfig --name rpscluster
```

The output indicates an error:

```
aws: error: the following arguments are required: --name
```

An error occurred (UnrecognizedClientException) when calling the DescribeCluster operation: The security token included in the request is invalid.

Other commands shown include:

```
C:\WINDOWS\system32>aws --version
aws-cli/1.16.113 Python/3.6.0 Windows/10 botocore/1.12.103

C:\WINDOWS\system32>aws configure
AWS Access Key ID [*****]: AKIAJQ4LHPP3M46VU56Q
AWS Secret Access Key [*****]: 779HTsbdnwy2xREJ05+8Z/c6duXQW80c3BjmGQ
Default region name [us-east-1]: ap-south-1
Default output format [json]:
```

The command prompt ends with:

```
C:\WINDOWS\system32>
```

ap-south-1

```
aws eks --region ap-south-1 update-kubeconfig --name rpscluster
```

```
C:\WINDOWS\system32>aws eks --region ap-south-1 update-kubeconfig --name rpscluster
aws: error: the following arguments are required: --name
An error occurred (UnrecognizedClientException) when calling the DescribeCluster operation: The security token included in the request is invalid.

C:\WINDOWS\system32>aws --version
aws-cli/1.16.113 Python/3.6.0 Windows/10 botocore/1.12.103

C:\WINDOWS\system32>aws configure
AWS Access Key ID [*****]: AKIAJQ4LHPP3A46VU56Q
AWS Secret Access Key [*****]: 779HTsbdnvW2xREJOS+BZ/c6duXQNV80c3BjmGQ
Default region name [us-east-2]: ap-south-1
Default output format [json];

C:\WINDOWS\system32>aws eks --region ap-south-1 update-kubeconfig --name rpscluster
Added new context arn:aws:eks:ap-south-1:567749627506:cluster/rpscluster to C:/Users/Balasubramaniam/.kube/config

C:\WINDOWS\system32>
```

We can now test our configurations using the kubectl get svc command:

Kubectl get svc

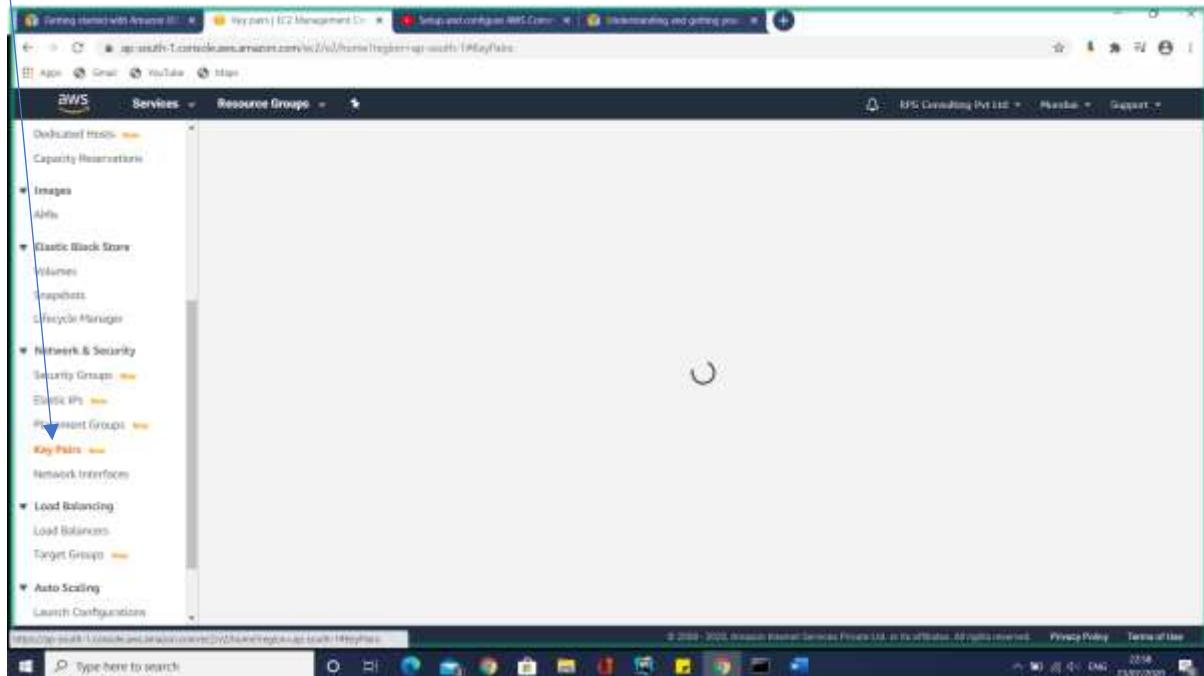
```
C:\WINDOWS\system32>kubectl get svc
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP  10.100.0.1   <none>        443/TCP   29m

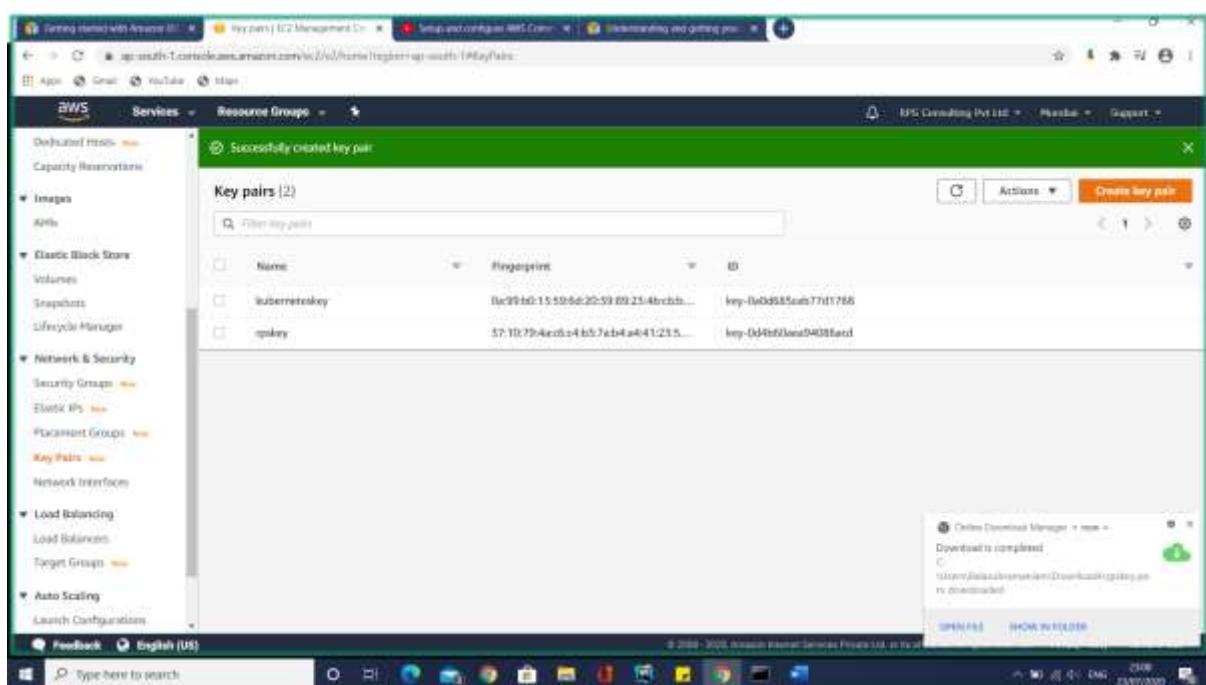
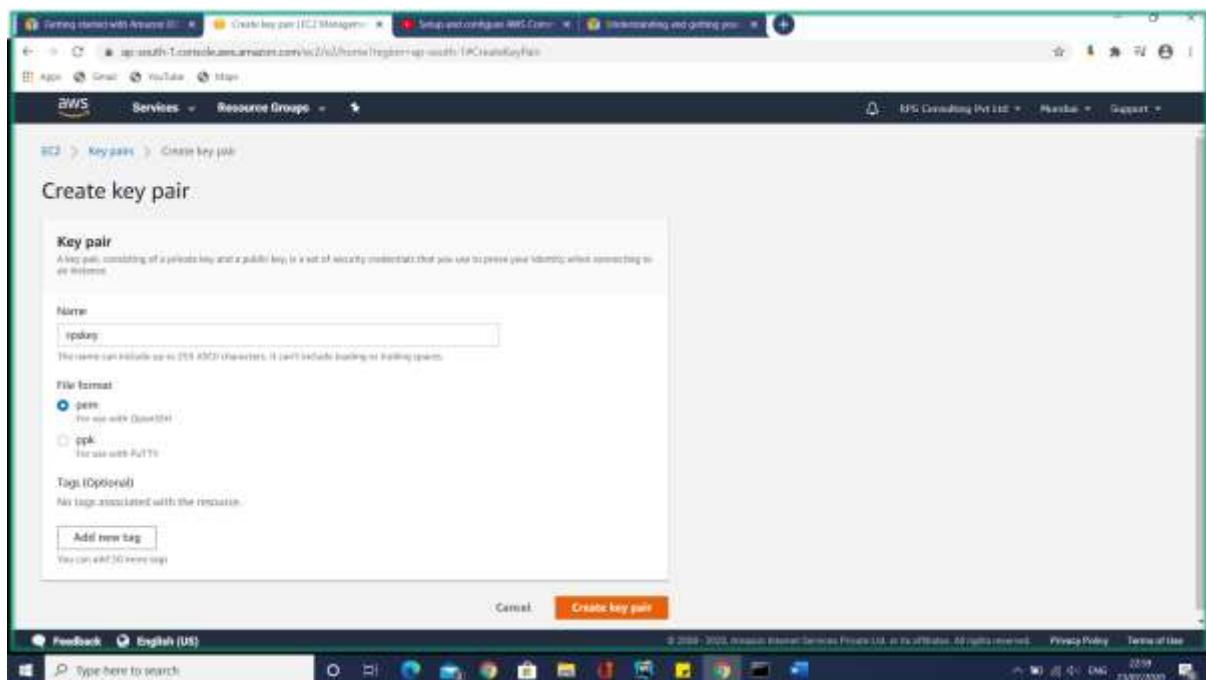
C:\WINDOWS\system32>
```

## Step 4

### Create Key Pair using Console

- Login to AWS Management Console and choose an AWS Region to create a key pair.
- Click on services and find EC2 under Compute services.
- Go to Network & Security > Key Pairs, and then choose Create Key Pair.





# Step 5: Launching Kubernetes worker nodes

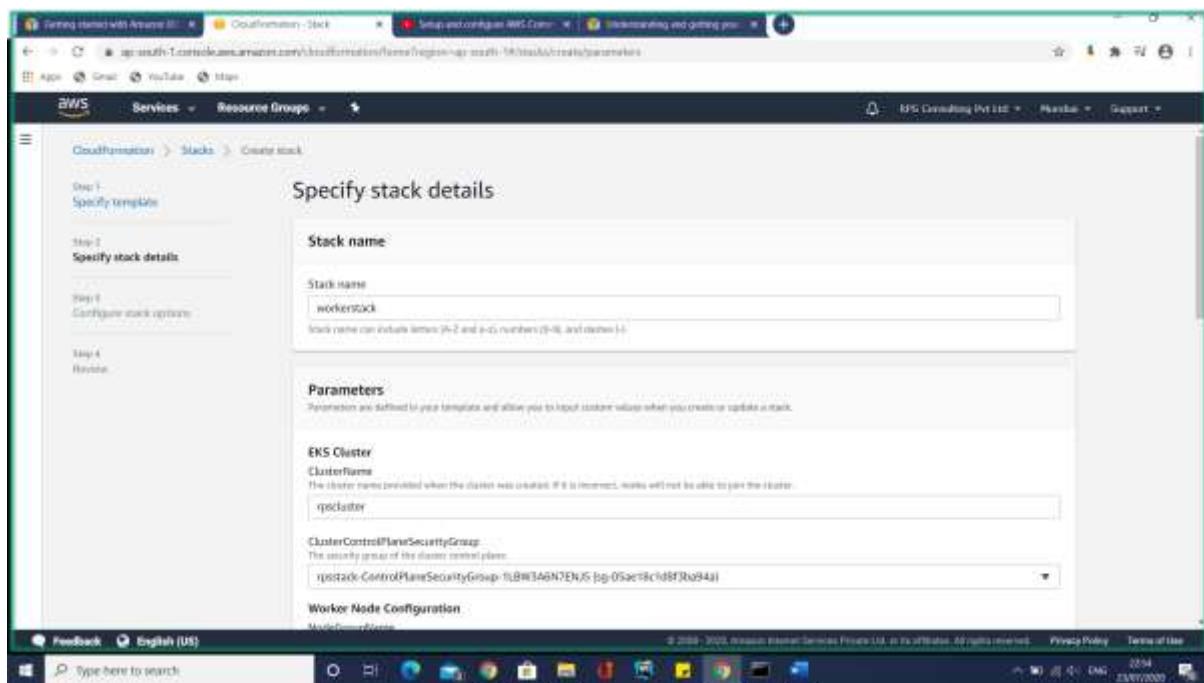
Now that we've set up our cluster and VPC networking, we can now launch Kubernetes worker nodes. To do this, we will again use a CloudFormation template.

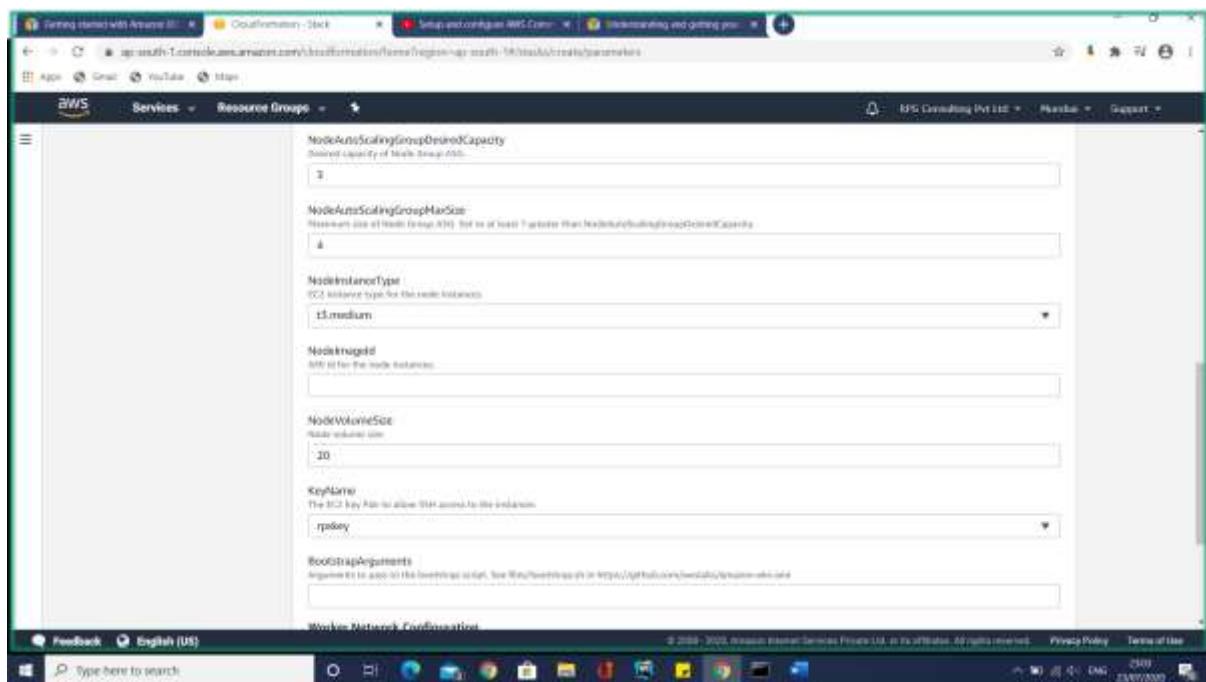
Open CloudFormation, click **Create Stack**, and this time use the following template URL:

<https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2019-01-09/amazon-eks-nodegroup.yaml>

- **ClusterName** – the name of your Kubernetes cluster (e.g. demo)
- **ClusterControlPlaneSecurityGroup** – the same security group you used for creating the cluster in previous step.
- **NodeGroupName** – a name for your node group.
- **NodeAutoScalingGroupMinSize** – leave as-is. The minimum number of nodes that your worker node Auto Scaling group can scale to.
- **NodeAutoScalingGroupDesiredCapacity** – leave as-is. The desired number of nodes to scale to when your stack is created.
- **NodeAutoScalingGroupMaxSize** – leave as-is. The maximum number of nodes that your worker node Auto Scaling group can scale out to.

- **NodeInstanceType** – leave as-is. The instance type used for the worker nodes.
- **NodeImageId** – the Amazon EKS worker node AMI ID for the region you’re using. For us-east-1, for example: ami-0c5b63ec54dd3fc38
- **KeyName** – the name of an Amazon EC2 SSH key pair for connecting with the worker nodes once they launch.
- **BootstrapArguments** – leave empty. This field can be used to pass optional arguments to the worker nodes bootstrap script.
- **VpcId** – enter the ID of the VPC you created in Step 2 above.
- **Subnets** – select the three subnets you created in Step 2 above.





Go to the link to view AMI Image Id

<https://docs.aws.amazon.com/eks/latest/userguide/eks-optimized-ami.html>

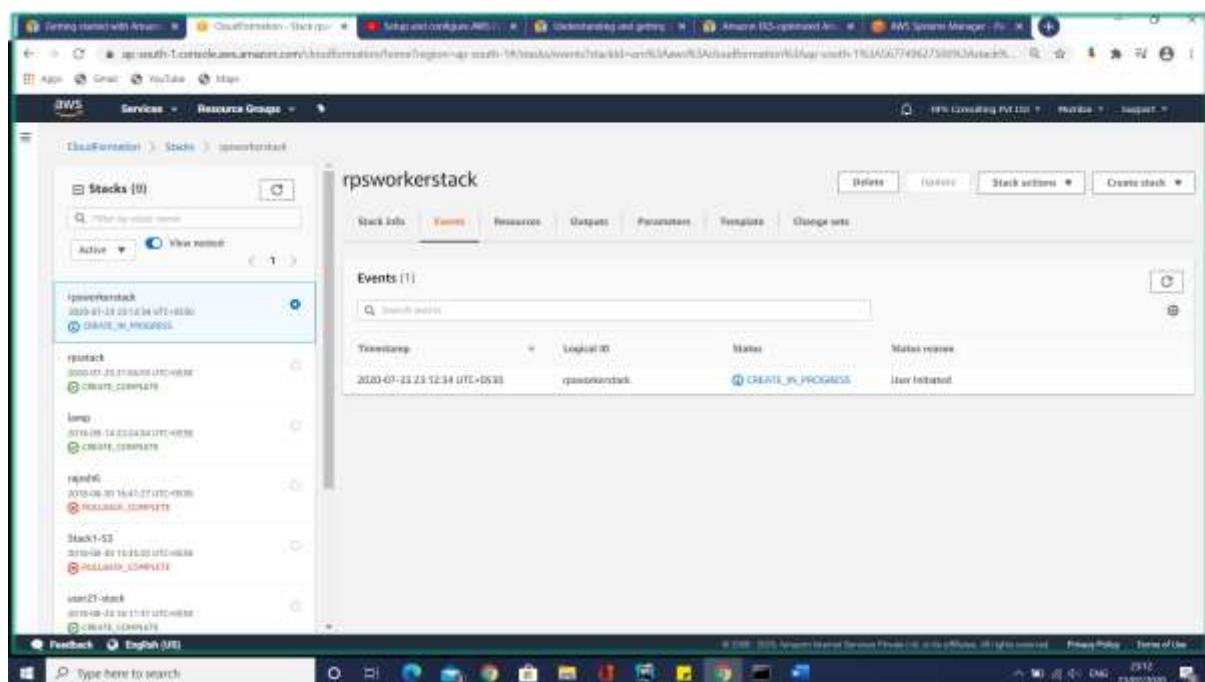
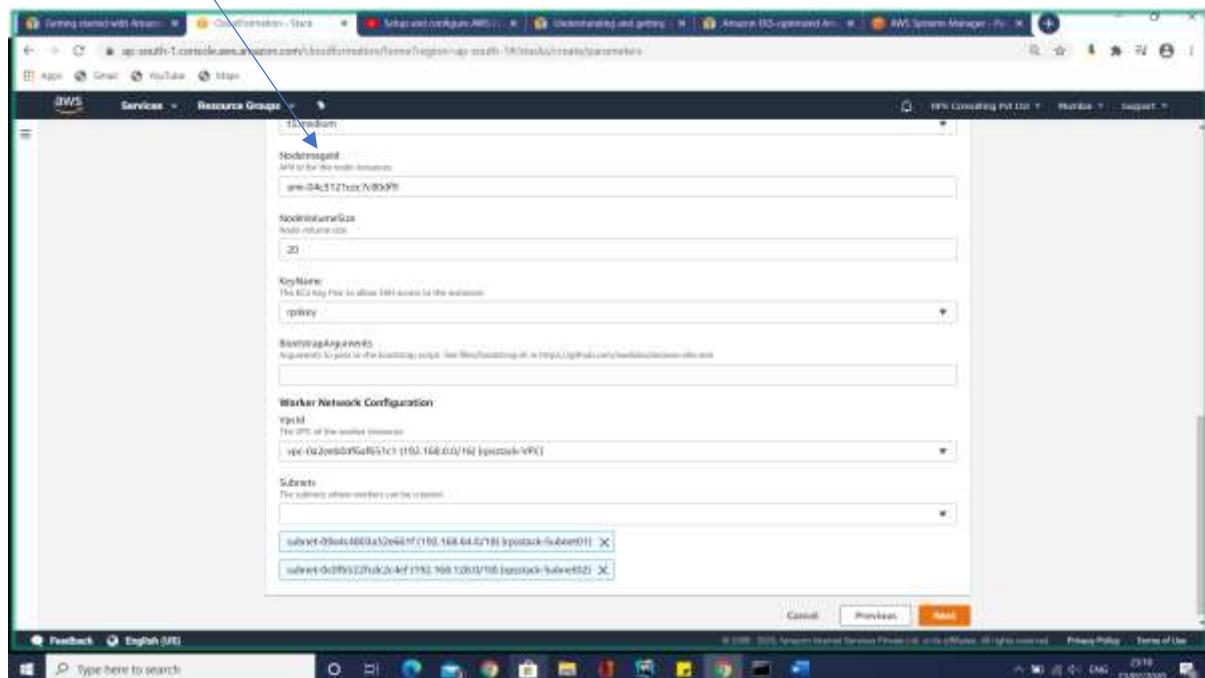
The screenshot shows the 'Amazon EKS-optimized AMIs' section of the AWS EKS User Guide. It lists AMIs for different regions:

Region	Amazon EKS-optimized Amazon Linux 2 AMI	Accelerated
east-1	View AMI ID	
Asia Pacific (Mumbai) (ap-south-1)	View AMI ID	
Asia Pacific (Tokyo) (ap-northeast-1)	View AMI ID	
Asia Pacific (Seoul) (ap-northeast-2)	View AMI ID	
Asia Pacific (Singapore) (ap-southeast-1)	View AMI ID	
Asia Pacific (Sydney) (ap-southeast-2)	View AMI ID	
Canada (Central) (ca-central-1)	View AMI ID	

A tooltip at the bottom left states: "Important: These AMIs require the latest AWS CloudFormation node template. You can't use these AMIs."

AMI ID for asia pacific south

ami-04c3121cec7c80df9



The screenshot shows the AWS CloudFormation console with the 'rpworkerstack' stack selected. The left sidebar lists several stacks, including 'rpworkerstack' (Status: CREATE\_COMPLETE), 'rpsstack' (Status: CREATE\_COMPLETE), 'long' (Status: CREATE\_COMPLETE), 'rpsnode6' (Status: RELEASE\_COMPLETE), 'Stack1-53' (Status: RELEASE\_COMPLETE), and 'short27-stack' (Status: CREATE\_COMPLETE). The main panel displays the 'rpworkerstack' stack's Overview, which includes the Stack ID, Description (Amazon DCS - Node Group), Status (Status: ACTIVE), Creation time (2020-01-23 23:12:34 UTC+0530), Last update time (2020-01-23 23:12:34 UTC+0530), Birth status (Status: NOT\_DELETED), and Termination protection (Disabled). The 'Outputs' tab is active, showing two outputs: 'NodeInstanceRole' (Value: arn:aws:iam::567749627506:role/rpworkerstack-NodeInstanceRole-1AC9FSSBOFRZC) and 'NodeSecurityGroup' (Value: sg-0c7d859b4097a577e).

This screenshot is identical to the one above, showing the AWS CloudFormation console with the 'rpworkerstack' stack selected. The left sidebar and main panel are the same, but the 'Outputs' tab is now explicitly highlighted in red, indicating it is the active tab.

## NodeInstanceRole

arn:aws:iam::567749627506:role/rpworkerstack-NodeInstanceRole-1AC9FSSBOFRZC

## NodeSecurityGroup

sg-0c7d859b4097a577e

Download

```
https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2019-01-09  
/aws-auth-cm.yaml
```

Open the yml file

Open the file and replace the rolearn with the ARN of the *NodeInstanceRole* created above:

```
apiVersion: v1  
kind: ConfigMap  
metadata:  
  name: aws-auth  
  namespace: kube-system  
data:  
  mapRoles: |  
    - rolearn: <ARN of instance role>  
      username: system:node:{{EC2PrivateDNSName}}  
      groups:  
        - system:bootstrappers  
        - system:nodes
```

Go to g:\local disk\docker\aws

```
kubectl apply -f aws-auth-cm.yaml
```

```
C:\WINDOWS\system32>g
G:\Local disk\Dockers\AWS>kubectl apply -f aws-auth-cm.yaml
configmap/aws-auth created
G:\Local disk\Dockers\AWS>
```

Use kubectl to check on the status of your worker nodes:

```
kubectl get nodes -watch
```

```
kubectl get nodes
```

## Step 6: Installing a demo app on Kubernetes

```
Administrator: Command Prompt F:\vldt> run -it --image=mysql --restart=Never mysql -h mysql -ppassword
C:\WINDOWS\system32>cd F:\virtusausjune2828\appointmentdocker
C:\WINDOWS\system32>f:
F:\virtusausjune2828\appointmentdocker>kubectl apply -f mysql-pv.yaml
persistentvolume/mysql-pv-volume created
persistentvolumeclaim/mysql-pv-claim created

F:\virtusausjune2828\appointmentdocker>kubectl apply -f mysql-deployment.yaml
service/mysql created
deployment.apps/mysql created

F:\virtusausjune2828\appointmentdocker>kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/mysql-78dc9c6b94-572pv     0/1     ContainerCreating   0          13s

NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes ClusterIP  10.100.0.1  <none>        443/TCP   75m
service/mysql   ClusterIP  None        <none>        3306/TCP  13s

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mysql  0/1     1           0           14s

NAME           DESIRED  CURRENT   READY   AGE
replicaset.apps/mysql-78dc9c6b94  1        1         0        14s

F:\virtusausjune2828\appointmentdocker>kubectl get services
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP  10.100.0.1  <none>        443/TCP   75m
mysql        ClusterIP  None        <none>        3306/TCP  42s

F:\virtusausjune2828\appointmentdocker>kubectl run -it --rm --image=mysql --restart=Never mysql-client -- mysql -h mysql -ppassword
if you don't see a command prompt, try pressing enter.
```

goto appointmenntdocker project in f:\virtusausjune

kubectl apply -f mysql-pv.yaml

kubectl apply -f mysql-deployment.yaml

kubectl get all

kubectl get services

kubectl describe deployment mysql

kubectl get pods -l app=mysql

kubectl describe pvc mysql-pv-claim

kubectl logs mysql-78dc9c6b94-vfgtq

kubectl get services

get ipaddress

kubectl run -it --rm --image=mysql --restart=Never mysql-client -- mysql -h mysql -ppassword

```
(don't change password it's ppassword)  
create database virtusausappointmentdb;  
show databases;
```

```
=====
```

Creating the secrets

```
kubectl create secret generic mysql-root-pass --from-  
literal=password=password
```

```
kubectl create secret generic mysql-user-pass --from-  
literal=username=demo_user --from-literal=password=demo_pass
```

```
kubectl create secret generic mysql-db-url --from-  
literal=database=virtusausappointmentdb --from-  
literal=url='jdbc:mysql://mysql:3306/virtusausappointmentdb?useSSL=false'
```

```
kubectl get secrets
```

```
kubectl describe secrets mysql-user-pass
```

```
kubectl get persistentvolumes
```

```
=====
```

```
kubectl create deployment demo --image=appointmentapp/v1 --dry-run -  
o=yaml > deployment.yaml
```

```
-----
```

```
kubectl create service clusterip demo --tcp=8070:8070 --dry-run -o=yaml >>  
deployment.yaml
```

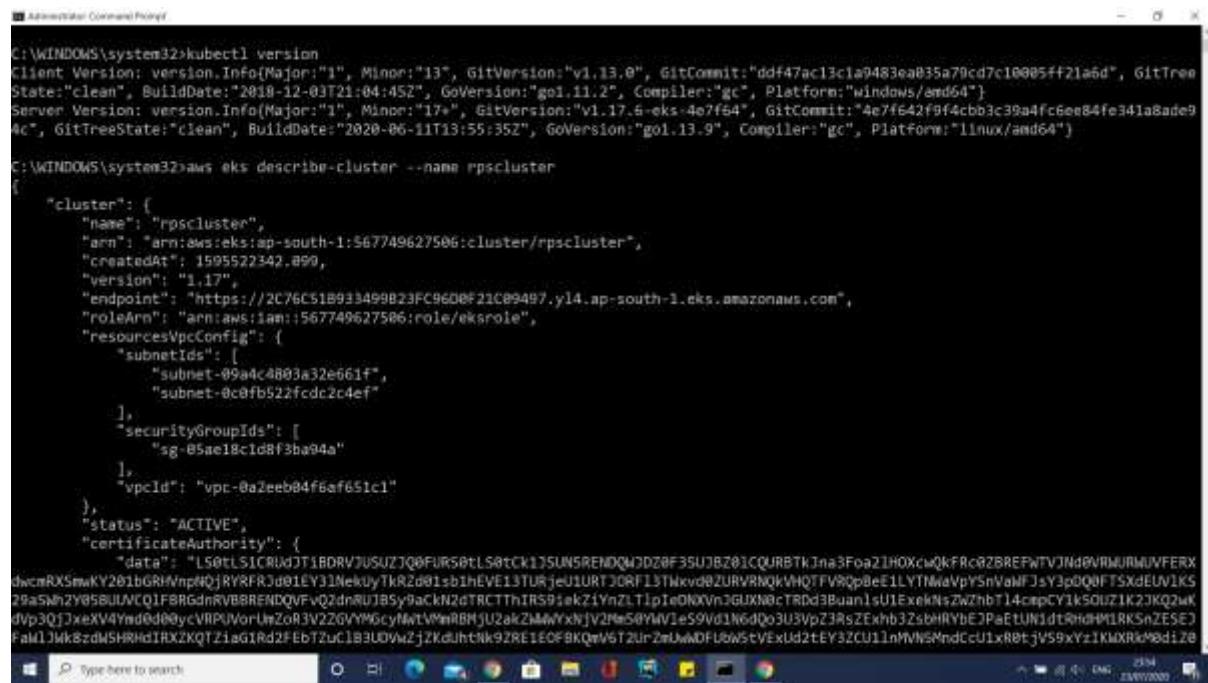
```
=====
```

```
kubectl apply -f deployment_v2.yaml
```

```
kubectl get pods
```

```
kubectl get services
```

```
kubectl port-forward service/demo 8070:8070
```



```
C:\WINDOWS\system32>kubectl version
Client Version: version.Info{Major:"1", Minor:"13", GitVersion:"v1.13.0", GitCommit:"dd47ac13c1a9483ea035a79cd710005ff21a6d", GitTreeState:"clean", BuildDate:"2018-12-03T21:04:45Z", GoVersion:"go1.11.2", Compiler:"gc", Platform:"windows/amd64"}
Server Version: version.Info{Major:"1", Minor:"17+", GitVersion:"v1.17.6-eks-4e7f64", GitCommit:"4e7f642f914ccb3c38a4fc6ee84fe341a8ade94c", GitTreeState:"clean", BuildDate:"2020-06-11T13:55:35Z", GoVersion:"go1.13.9", Compiler:"gc", Platform:"linux/amd64")

C:\WINDOWS\system32>aws eks describe-cluster --name rpscluster
{
  "cluster": {
    "name": "rpscluster",
    "arn": "arn:aws:eks:ap-south-1:567749627506:cluster/rpscluster",
    "createdAt": 1595522342.899,
    "version": "1.17",
    "endpoint": "https://2C76C518933499823FC9608F21C08497.y14.ap-south-1.eks.amazonaws.com",
    "roleArn": "arn:aws:iam::567749627506:role/eksrole",
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-09a4c4803a32e661f",
        "subnet-0cfb522fcfdc2c4ef"
      ],
      "securityGroupIds": [
        "sg-05ae18c1d8f3ba94a"
      ],
      "vpcId": "vpc-0a2eeb84f6af651c1"
    },
    "status": "ACTIVE",
    "certificateAuthority": {
      "data": "LS0tLS1CRUdjQTBDRVJUSUZJQ0FUR50tLS0tCK1JSUN5RENDQWDDZ0F35UJBZ01CQURBTkJna3Foa21H0Xcw0kFRc0ZBREFwTVJNdvVRMURMuUFERXdwcmRXSmwKY201bGRUWhpW0jRYRFRJ001Y32NeKuyTRkZd81sb1hEVE13TURjeU1URTJ0RF13TwvdbZURVRNQkVHQTFV80pBeE1LYTMwVpYSnVaMFjsYjpdQ0FTSXxEUV1KS29aswM2Yy5BuLVCQ1FBGdmRVB8RENDQVFvQ2dnRU1B5y9ackn2dTRCTThRS9iaK2iYnZLT1pted0XVnJGUKN8ctrDd3BuanisUIExeKNS2wZhbt14cepCY1kSOU1K2JQ2wKdVp3Qj2xeXVqYnd0d8@ycVIIPIUVorUmZoR3V22GVYMMcYmVtWmRRBhjlU2akZMaWYxNjV2Ma5@YmV1eS9vd1Ng0o3U3VpZ3Rs2Ewhb3ZsbHRYEBJpEtU1N1dtRhDHMLRKSNZESE)Fak1JMK8zdWSHRMdIRX2KQTZiaG1Rd2fEbT2uClB3UOVw2jZKduhtnk92RE1EOFBKQmWt2Urn2mUwMDFlUbWStVExUd2tEY32CU1lnMVNsMndCcU1xR8tjVS9xYzIKMXRkM8dize"
    }
  }
}
```

```
aws eks describe-cluster --name rpscluster
```

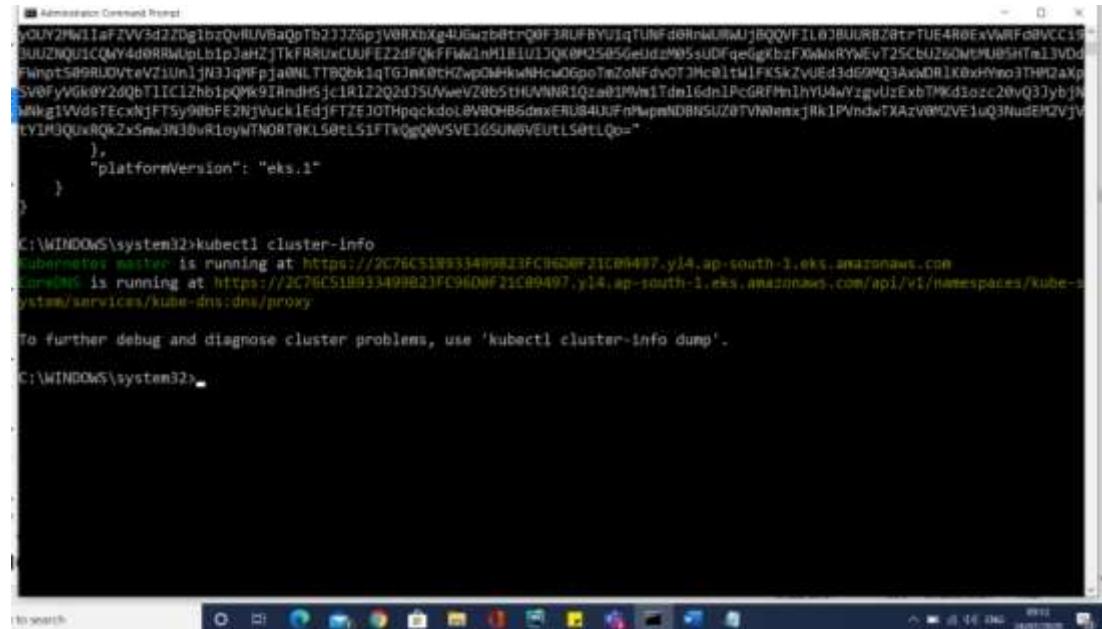
CLI commands

## To create Cluster

```
aws eks create-cluster --name prod \
--role-arn arn:aws:iam::012345678910:role/eks-service-role-AWSServiceRoleForAmazonEKS-J70NKE3BQ4PI \
--resources-vpc-config subnetIds=subnet-6782e71e,subnet-e7e761ac,securityGroupIds=sg-6979fe18
```

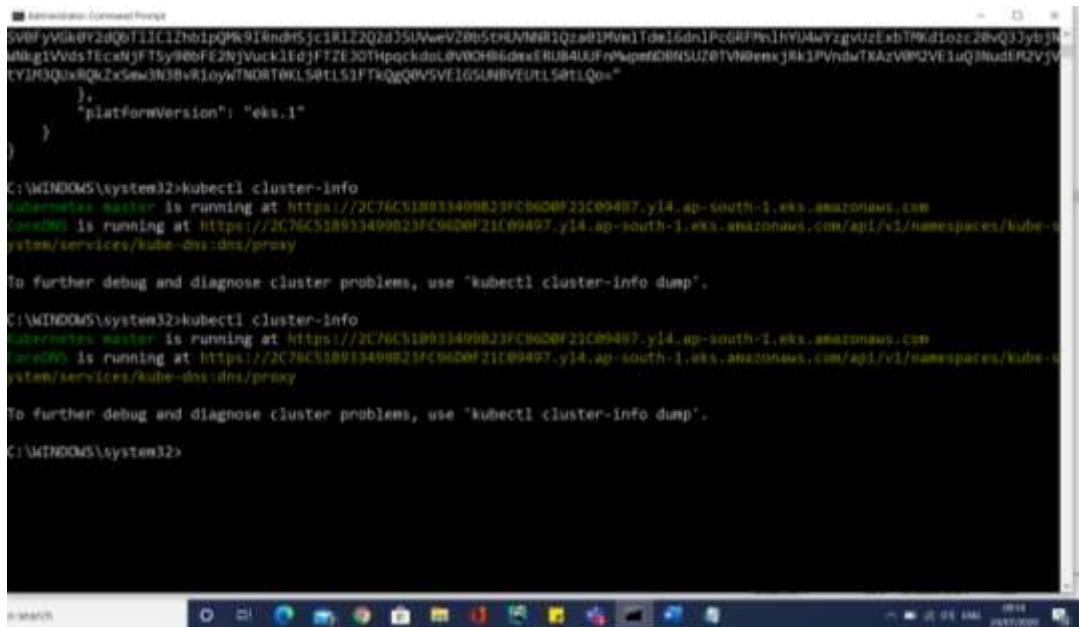
## To Delete Cluster

```
aws eks delete-cluster --name devel
```



```
C:\Windows\system32>kubectl cluster-info
Kubernetes master is running at https://207.61.189.33:443/v2/fetch?21C89497.y14.ap-south-1.eks.amazonaws.com
CoreDNS is running at https://207.61.189.33:443/v2/fetch?21C89497.y14.ap-south-1.eks.amazonaws.com/api/v1/namespaces/kube-system/services/kube-dns:proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```



```
Administrator: Command Prompt
yV8yV8kdn2dQ612Lc2he3pqH%u1Rn05Jc3RL2Q2d250Vwv2B65d8UvhmR1QzabUfve2TdvLsdr1PcGKfPx1hY4k7gvoxEb7Rfd3oec28hQ3jyb3j>
utkgp1W6sTEcxNjFTsy88fE2njvucK3E8jF2E3THpqckdab@VB0Hl1dex8RUB4UfN%pref0BfNSUZBTvN0ensk3jRk1PVndwTAzV8M2VE1uQ3nudEP2Vjv
tYj3Qux80K2c5ew3N38vR1oywTN0Rt0KLS0TLS3FTkoqQ0VsVE16SUNBVETLS@tLQo="
},
"platformVersion": "eks.1"
}

C:\WINDOWS\system32>kubectl cluster-info
kubernetes master is running at https://2C76C51803499823FC96D0F21C89497.y14.ap-south-1.eks.amazonaws.com
CoreDNS is running at https://2C76C51803499823FC96D0F21C89497.y14.ap-south-1.eks.amazonaws.com/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use "kubectl cluster-info dump".

C:\WINDOWS\system32>
```

To Analyse cluster dump

'kubectl cluster-info dump'

Dashboard

kubectl apply -f <https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.3.6/components.yaml>

```
kubectl get deployment metrics-server -n kube-system
```

```
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-
beta8/aio/deploy/recommended.yaml
```

```
kubectl apply -f eks-admin-service-account.yaml
```

```
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret  
| grep eks-admin | awk '{print $1}')
```