

Create the K8s EKS, further you have to do the deployment of Nginx application

## Step 1: Create a Ec2 Instance:

EC2 > Instances > Launch an instance

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

**Name and tags** [Info](#)

Name

K8s\_Task [Add additional tags](#)

**▼ Application and OS Images (Amazon Machine Image)** [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Recents

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

SUSE Linux

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type  
ami-03b6d83c60fc5f7c (64-bit (x86)) / ami-0416007a1d2fa2dd (64-bit (arm))  
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2024-02-07

Architecture

AMI ID

64-bit (x86)

ami-03b6d83c60fc5f7c

Verified provider

**▼ Summary**

Number of instances [Info](#)

1

Software Image (AMI)

Canonical, Ubuntu, 22.04 LTS, ...[read more](#)  
ami-03b6d83c60fc5f7c

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GB of bandwidth to the internet.

Cancel

Launch instance

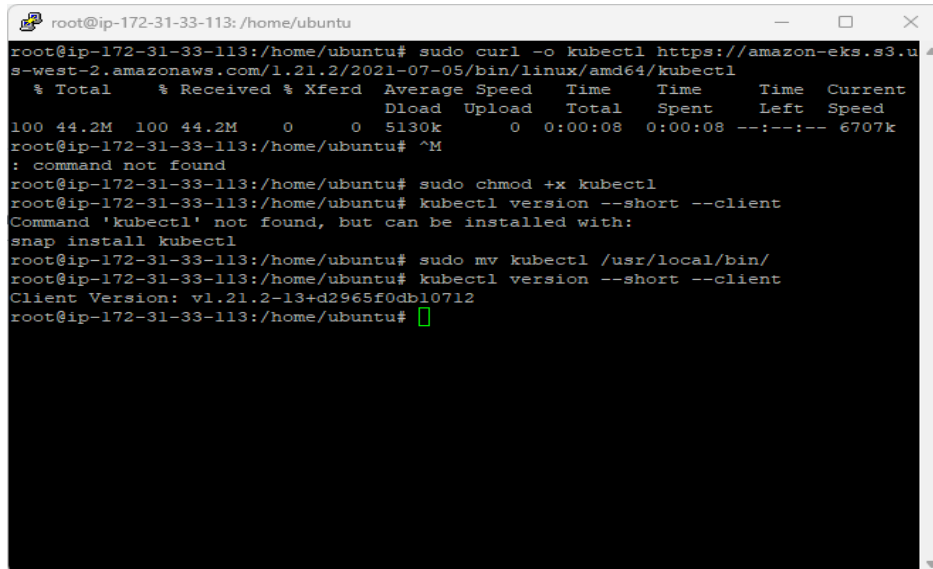
[Review commands](#)

## Step 2: Using Putty connect the instance and update.

```
root@ip-172-31-33-113: /home/ubuntu
Get:28 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multivers
e amd64 c-n-f Metadata [116 B]
Get:29 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [120
5 kB]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [219
kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Package
s [1476 kB]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-e
n [244 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages
[846 kB]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en
[161 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Met
adata [16.8 kB]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Package
s [37.1 kB]
Get:37 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-e
n [7476 B]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f M
etadata [260 B]
Fetched 29.8 MB in 6s (5280 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
34 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-33-113:/home/ubuntu#
```

Step 3: To install Kubernetes, Run the following command one by one.

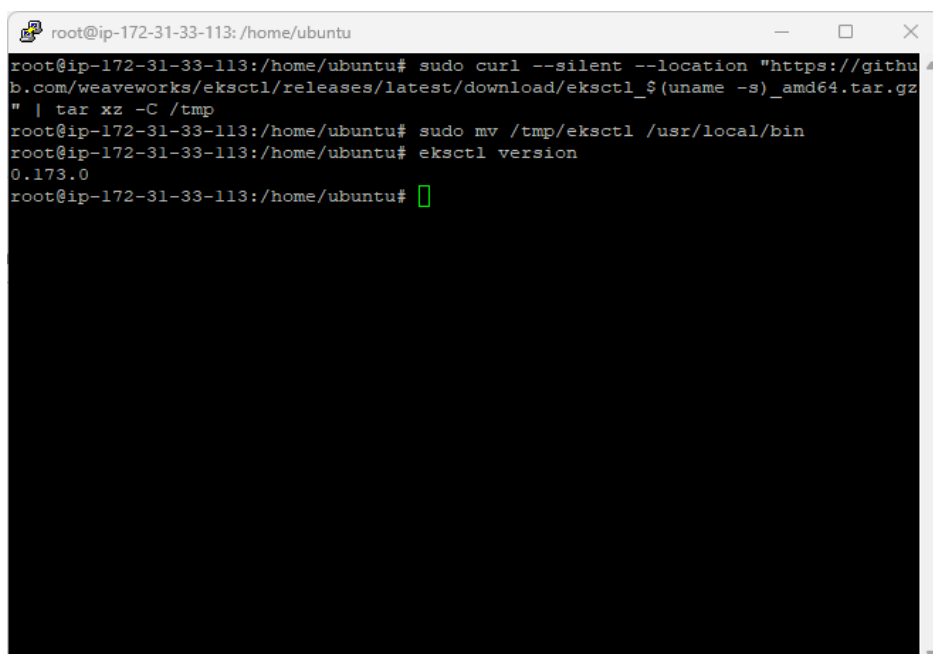
```
curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl
chmod +x ./kubectl
mv ./kubectl /usr/local/bin
kubectl version --short --client
```



```
root@ip-172-31-33-113: /home/ubuntu
root@ip-172-31-33-113:/home/ubuntu# sudo curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/linux/amd64/kubectl
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 44.2M 100 44.2M 0 0 5130k 0 0:00:08 0:00:08 --:--:-- 6707k
root@ip-172-31-33-113:/home/ubuntu# ^M
: command not found
root@ip-172-31-33-113:/home/ubuntu# sudo chmod +x kubectl
root@ip-172-31-33-113:/home/ubuntu# kubectl version --short --client
Command 'kubectl' not found, but can be installed with:
snap install kubectl
root@ip-172-31-33-113:/home/ubuntu# sudo mv kubectl /usr/local/bin/
root@ip-172-31-33-113:/home/ubuntu# kubectl version --short --client
Client Version: v1.21.2-13+d2965f0db10712
root@ip-172-31-33-113:/home/ubuntu#
```

Step 4: To install eksctl, Run the following command one by one.

```
curl --silent -location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
```



```
root@ip-172-31-33-113: /home/ubuntu
root@ip-172-31-33-113:/home/ubuntu# sudo curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
root@ip-172-31-33-113:/home/ubuntu# sudo mv /tmp/eksctl /usr/local/bin
root@ip-172-31-33-113:/home/ubuntu# eksctl version
0.173.0
root@ip-172-31-33-113:/home/ubuntu#
```

## Step 5: Create New User to connect AWS from Ec2 Instance.

IAM > Users > Create user

Step 1  
Specify user details

Step 2  
Set permissions

Step 3  
Review and create

### Specify user details

**User details**

User name

Node\_deploy

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and +, -, @, \_, . (pound)

☐ Provide user access to the AWS Management Console - optional  
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

☒ If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel Next

IAM > Users > Create user

Step 1  
Specify user details

Step 2  
Set permissions

Step 3  
Review and create

### Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

**Permissions options**

☐ Add user to group  
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions  
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly  
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

**Permissions policies (1/1177)**

Choose one or more policies to attach to your new user.

Q Search Filter by Type All types

Policy name	Type	Attached entities
<input type="checkbox"/> AccessAnalyzerServiceRolePolicy	AWS managed	0
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	3
<input type="checkbox"/> AdministratorAccess-Ampify	AWS managed	0
<input type="checkbox"/> AdministratorAccess-AWSStudioBuddi	AWS managed	0
<input type="checkbox"/> AmazonWorkSpacesDeviceSetup	AWS managed	0

Cancel Next

IAM > Users > Node\_deploy > Create access key

Step 1  
Access key best practices & alternatives

Step 2 - optional  
Set description tag

Step 3  
Retrieve access keys

### Access key best practices & alternatives

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

**Use case**

☒ Command Line Interface (CLI)  
You plan to use this access key to enable the AWS CLI to access your AWS account.

☐ Local code  
You plan to use this access key to enable application code in a local development environment to access your AWS account.

☐ Application running on an AWS compute service  
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

☐ Third-party service  
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

☐ Application running outside AWS  
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

☐ Other  
Your use case is not listed here.

**Alternatives recommended**

- Use [AWS CloudShell](#), a browser-based CLI to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

**Confirmation**

☒ I understand the above recommendation and want to proceed to create an access key.

Cancel Next

**Access key created**  
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Users > Node\_deploy > Create access key

Step 1  
Access key best practices & alternatives

Step 2 - optional  
Set description tag

Step 3  
Retrieve access keys

### Retrieve access keys

**Access key**

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
<input checked="" type="checkbox"/> AKIA6ODU4QSV77546DK3	<input type="checkbox"/> <a href="#">Show</a>

**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

Download .csv file Done

## Step 6 : Configure the New IAM user in the master node

```
root@ip-172-31-33-113: /home/ubuntu
root@ip-172-31-33-113:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIA6ODU4QSV77S46DK3
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
root@ip-172-31-33-113:/home/ubuntu#
```

## Step 7 : Create cluster in AWS using below commands.

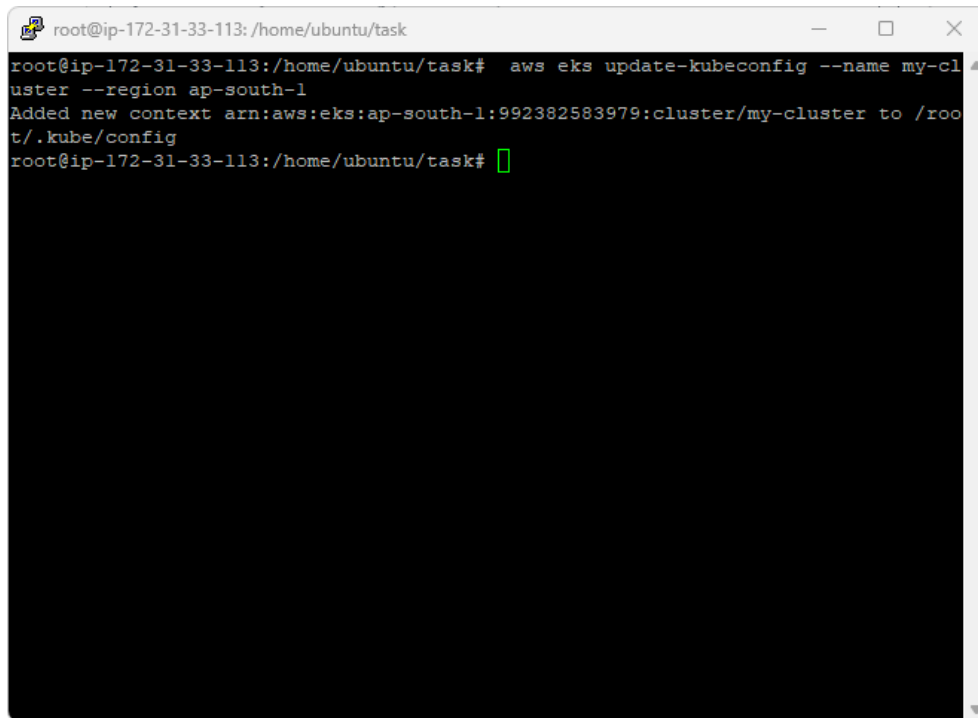
eksctl create cluster --name my-cluster --region ap-south-1 --node-type t2.micro --nodes 2

```
root@ip-172-31-33-113: /home/ubuntu/task
root@ip-172-31-33-113:/home/ubuntu/task# eksctl create cluster --name my-cluster --region ap-south-1 --node-type t2.micro --nodes 2
2024-03-01 07:55:18 [ ] eksctl version 0.173.0
2024-03-01 07:55:18 [ ] using region ap-south-1
2024-03-01 07:55:18 [ ] skipping ap-south-1c from selection because it doesn't support the following instance type(s): t2.micro
2024-03-01 07:55:18 [ ] setting availability zones to [ap-south-1a ap-south-1b]
2024-03-01 07:55:18 [ ] subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.64.0/19
2024-03-01 07:55:18 [ ] subnets for ap-south-1b - public:192.168.32.0/19 private:192.168.96.0/19
2024-03-01 07:55:18 [ ] nodegroup "ng-df2b36d9" will use "" [AmazonLinux2/1.29]
2024-03-01 07:55:18 [ ] using Kubernetes version 1.29
2024-03-01 07:55:18 [ ] creating EKS cluster "my-cluster" in "ap-south-1" region with managed nodes
2024-03-01 07:55:18 [ ] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2024-03-01 07:55:18 [ ] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=my-cluster'
2024-03-01 07:55:18 [ ] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "my-cluster" in "ap-south-1"
2024-03-01 07:55:18 [ ] CloudWatch logging will not be enabled for cluster "my-cluster" in "ap-south-1"
2024-03-01 07:55:18 [ ] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south-1 --cluster=my-cluster'
2024-03-01 07:55:18 [ ]
```

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	K8s_Task	i-07f61ad929ae24ce9	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a
<input type="checkbox"/>	my-cluster-ng-df2b36d9-Node	i-099e8ccbaa2169bd6	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a
<input type="checkbox"/>	my-cluster-ng-df2b36d9-Node	i-0641893316277dde9	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b

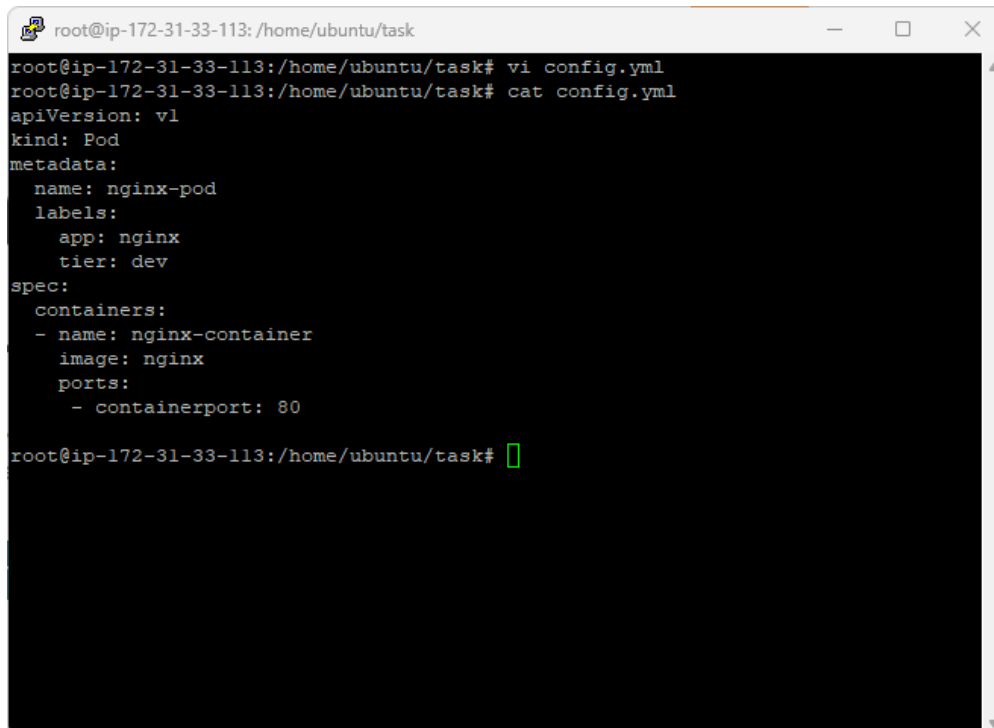
Step 8 : Update the Kubernetes configure using below command.

aws eks update-kubeconfig --name my-cluster --region ap-south-1

A terminal window with a dark background and light text. The title bar shows 'root@ip-172-31-33-113: /home/ubuntu/task'. The command 'aws eks update-kubeconfig --name my-cluster --region ap-south-1' has been entered and executed. The output shows 'Added new context arn:aws:eks:ap-south-1:992382583979:cluster/my-cluster to /root/.kube/config'. The prompt is now ready for the next command.

```
root@ip-172-31-33-113: /home/ubuntu/task# aws eks update-kubeconfig --name my-cluster --region ap-south-1
Added new context arn:aws:eks:ap-south-1:992382583979:cluster/my-cluster to /root/.kube/config
root@ip-172-31-33-113: /home/ubuntu/task#
```

Step 9: Create new directory and create yml file to deploy nginx pod.

A terminal window with a dark background and light text. The title bar shows 'root@ip-172-31-33-113: /home/ubuntu/task'. The user has created a file named 'config.yml' using 'vi' and then viewed its contents with 'cat'. The file contains a Kubernetes Pod definition for nginx.

```
root@ip-172-31-33-113: /home/ubuntu/task# vi config.yml
root@ip-172-31-33-113: /home/ubuntu/task# cat config.yml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
    tier: dev
spec:
  containers:
  - name: nginx-container
    image: nginx
    ports:
    - containerport: 80
root@ip-172-31-33-113: /home/ubuntu/task#
```

## Step 10: Create and display PODS:

```
kubectl create -f nginx-pod.yaml
kubectl get pod
kubectl get pod -o wide
kubectl get pod nginx-pod -o yaml
kubectl describe pod nginx-pod
```

```
root@ip-172-31-33-113: /home/ubuntu/task
root@ip-172-31-33-113:/home/ubuntu/task# vi config.yaml
root@ip-172-31-33-113:/home/ubuntu/task# kubectl create -f config.yaml
pod/nginx-pod created
root@ip-172-31-33-113:/home/ubuntu/task# kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
nginx-pod     1/1     Running   0           88s
root@ip-172-31-33-113:/home/ubuntu/task# kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
nginx-pod     1/1     Running   0           4m47s
root@ip-172-31-33-113:/home/ubuntu/task# kubectl get pod -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP              NODE
nginx-pod     1/1     Running   0           4m56s   192.168.19.0   ip-192-168-2-178.ap-south-1.compu
te.internal    <none>         <none>
root@ip-172-31-33-113:/home/ubuntu/task# kubectl get pod config -o yaml
Error from server (NotFound): pods "config" not found
root@ip-172-31-33-113:/home/ubuntu/task# kubectl get pod nginx-pod -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2024-03-01T08:19:03Z"
  labels:
    app: nginx
    tier: dev
  name: nginx-pod
  namespace: default
  resourceVersion: "3341"
  uid: e29e8917-e10d-465e-827e-500018fca0dd
```

```
root@ip-172-31-33-113: /home/ubuntu/task
root@ip-172-31-33-113:/home/ubuntu/task# kubectl describe pod nginx-pod
Name:          nginx-pod
Namespace:     default
Priority:       0
Node:          ip-192-168-2-178.ap-south-1.compute.internal/192.168.2.178
Start Time:    Fri, 01 Mar 2024 08:19:03 +0000
Labels:        app=nginx
               tier=dev
Annotations:   <none>
Status:        Running
IP:            192.168.19.0
IPs:
  IP: 192.168.19.0
Containers:
  nginx-container:
    Container ID:  containerd://817b76d4e68be3c0f20ee6da25dc285531aca9a888d0895754d7409c9e1b9747
    Image:         nginx
    Image ID:      docker.io/library/nginx@sha256:c26ae7472d624balfafd296e73cecc4f93f853088e6a9c
13c0d52f6ca5865107
    Port:         80/TCP
    Host Port:    0/TCP
    State:        Running
      Started:    Fri, 01 Mar 2024 08:19:12 +0000
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-tsz86 (ro)
```

Step 11 : Create HTML file inside pod and expose the port to 80.

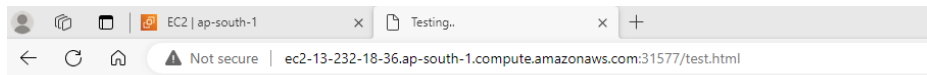
kubectl expose pod nginx-pod --type=NodePort --port=80

```
root@ip-172-31-33-113: /home/ubuntu/task
root@ip-172-31-33-113:/home/ubuntu/task# kubectl exec -it nginx-pod -- /bin/sh
# cat <<EOF > /usr/share/nginx/html/test.html
<!DOCTYPE html>
<html>
<head>
<title>Testing..</title>
</head>
<body>
<h1 style="color:rgb(90,70,250);">Hello, Kubernetes...!</h1>
<h2>Congratulations, you passed :-)</h2>
</body>
</html>
EOF
exit
> > > > > > > > # root@ip-172-31-33-113:/home/ubuntu/task# kubectl expose pod nginx-pod --t
type=NodePort --port=80 kubectl expose pod nginx-pod --type=NodePort --port=80
service/nginx-pod exposed
root@ip-172-31-33-113:/home/ubuntu/task#
```

Step 12: Display Service and find Node Port:

```
root@ip-172-31-33-113: /home/ubuntu/task
root@ip-172-31-33-113:/home/ubuntu/task# kubectl describe svc nginx-pod
Name: nginx-pod
Namespace: default
Labels: app=nginx
        tier=dev
Annotations: <none>
Selector: app=nginx,tier=dev
Type: NodePort
IP Family Policy: SingleStack
IP Families: IPv4
IP: 10.100.209.89
IPs: 10.100.209.89
Port: <unset> 80/TCP
TargetPort: 80/TCP
NodePort: <unset> 31544/TCP
Endpoints: 192.168.19.0:80
Session Affinity: None
External Traffic Policy: Cluster
Events: <none>
root@ip-172-31-33-113:/home/ubuntu/task#
```

Step 13: find the Node port number from the display service and using that port number in cluster DNS to view the created HTML page



**Hello, Kubernetes...!**

**Congratulations, you passed :-)**