Create an Ec2 instance using configuration management tool Ansible.

Created two servers – one is working as a control plane, and another is the worker node.

```
root@server:~        root@server2:~
[root@server ~]#
```

```
root@server:~        root@server2:~
[root@server2 ~]#
```

Connection is built between both nodes.

```
root@server:~        root@server2:~
[root@server ~]# vim /etc/hosts
[root@server ~]# ping 172.31.90.152
PING 172.31.90.152 (172.31.90.152) 56(84) bytes of data.
64 bytes from 172.31.90.152: icmp_seq=1 ttl=127 time=2.15 ms
64 bytes from 172.31.90.152: icmp_seq=2 ttl=127 time=0.507 ms
64 bytes from 172.31.90.152: icmp_seq=3 ttl=127 time=0.714 ms
64 bytes from 172.31.90.152: icmp_seq=4 ttl=127 time=0.981 ms
^C
--- 172.31.90.152 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3046ms
rtt min/avg/max/mdev = 0.507/1.087/2.146/0.634 ms
[root@server ~]#
```

Copied ssh key in the both the nodes

```
root@server:~        root@server2:~
[root@server ~]# ssh-copy-id root@172.31.90.152
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '172.31.90.152 (172.31.90.152)' can't be established.
ED25519 key fingerprint is SHA256:9AqiAiA0+lBurdoex+Fi+08nXAV69XX5VSStgbZ52Zg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@172.31.90.152's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@172.31.90.152'"
and check to make sure that only the key(s) you wanted were added.

[root@server ~]#
```

```
[root@server2 ~]# ssh-copy-id root@172.31.85.199
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '172.31.85.199 (172.31.85.199)' can't be established.
ED25519 key fingerprint is SHA256:u/DNDpLZmQEo2zlNE9IdUk/PWKctVsRhmhIHQ4QS/pY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@172.31.85.199's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@172.31.85.199'"
and check to make sure that only the key(s) you wanted were added.

[root@server2 ~]#
```

Installed Ansible in the control plane.

```
[root@server ~]# rpmquery ansible-core
ansible-core-2.15.3-1.amzn2023.0.11.x86_64
[root@server ~]#
```

```
[server2]
server2
~
```

```
[root@server ~]# rpmquery ansible-core
ansible-core-2.15.3-1.amzn2023.0.11.x86_64
[root@server ~]# cd /etc/ansible
[root@server ansible]# vim ansible.cfg
[root@server ansible]# vim hosts
[root@server ansible]#
```

Ansible connection is successful.

```
[root@server ~]# ansible all -m ping
[WARNING]: Found both group and host with same name: server2
The authenticity of host 'server2 (172.31.90.152)' can't be established.
ED25519 key fingerprint is SHA256:9AqiAiA0+lBurdoex+Fi+08nXAV69XX5VSStgbZ52Zg.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: 172.31.90.152
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
[WARNING]: Platform linux on host server2 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of another
Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.15/reference_appendices/interpreter_discovery.html for more information.
server2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.9"
    },
    "changed": false,
    "ping": "pong"
}
[root@server ~]#
```

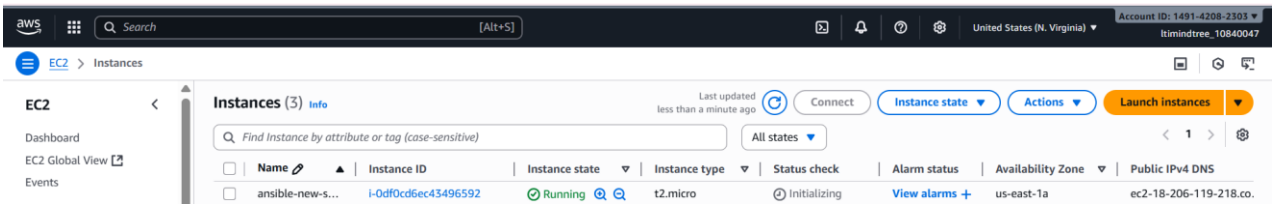
Wrote a playbook code to create an ec2 instance

```
root@server:/ansible-code   root@server2:~

[root@server ~]# mkdir /ansible-code
[root@server ~]# cd /ansible-code/
[root@server ansible-code]# vim playbook.yaml
```

```yaml
root@server:/ansible-code   root@server2:~

---
- name: Launch EC2 instance
  hosts: localhost
  connection: local
  gather_facts: false

  vars:
    region: us-east-1
    instance_type: t2.micro
    ami_id: ami-00ca32bbc84273381
    key_name: server-key
    security_group: my-sg

  tasks:
    - name: Launch instance
      amazon.aws.ec2_instance:
        name: ansible-new-server
        key_name: "{{ key_name }}"
        instance_type: "{{ instance_type }}"
        image_id: "{{ ami_id }}"
        wait: yes
        region: "{{ region }}"
        security_group: "{{ security_group }}"
        count: 1
        network:
          assign_public_ip: true
        tags:
          Environment: Devops
~
~
```

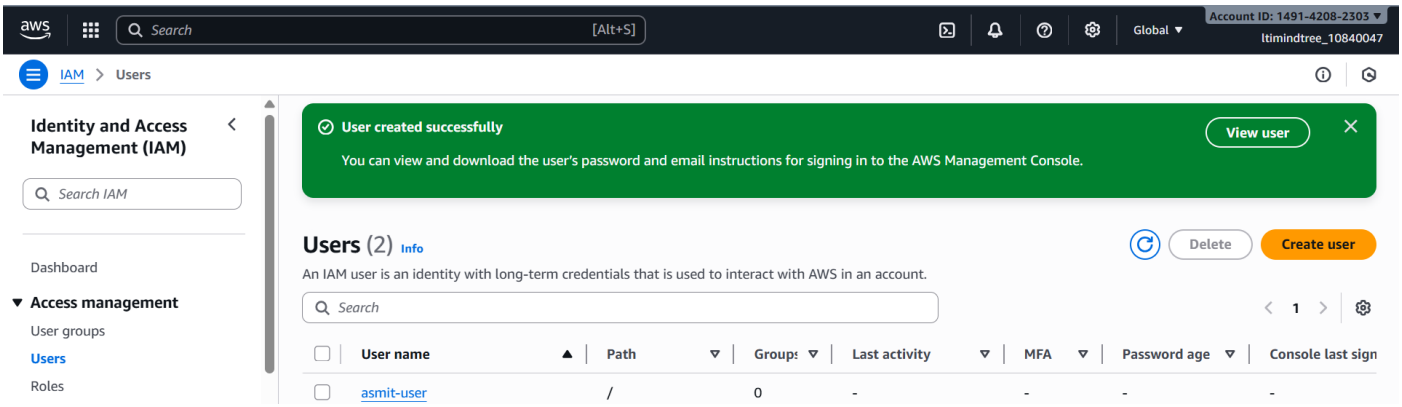
```
root@server:/ansible-code   root@server2:~

[root@server ~]# cd /ansible-code/
[root@server ansible-code]# ansible-playbook playbook.yaml --syntax-check
[WARNING]: Found both group and host with same name: server2

playbook: playbook.yaml
[root@server ansible-code]# ansible-playbook my-playbook.yaml -C
ERROR! the playbook: my-playbook.yaml could not be found
[root@server ansible-code]# ansible-playbook playbook.yaml -C
[WARNING]: Found both group and host with same name: server2

PLAY [Launch EC2 instance] *****************************************************************************************************

TASK [Launch instance] *********************************************************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[root@server ansible-code]# ansible-playbook playbook.yaml
[WARNING]: Found both group and host with same name: server2

PLAY [Launch EC2 instance] *****************************************************************************************************

TASK [Launch instance] *********************************************************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

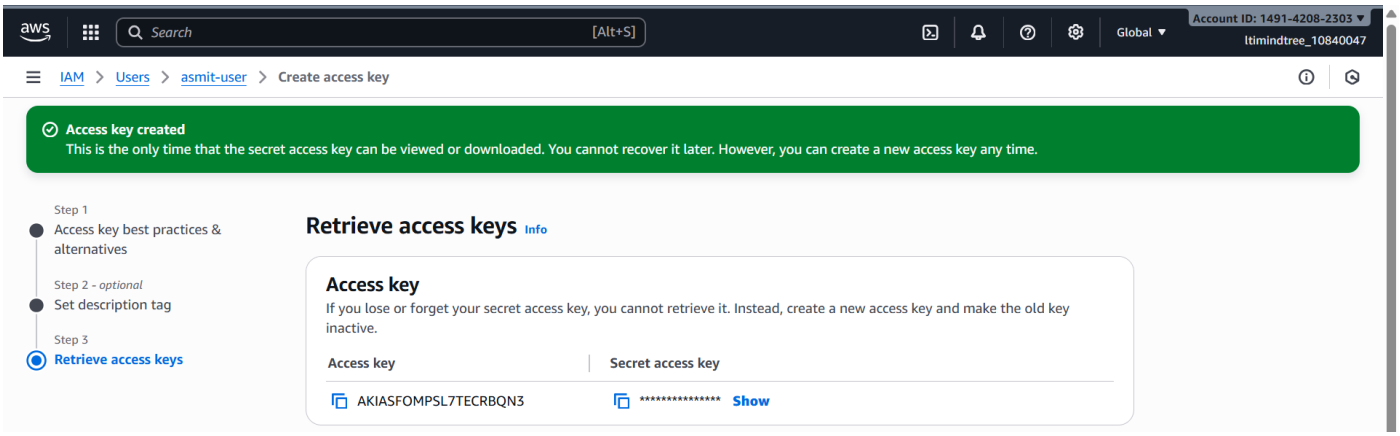
The instance named ansible-new-server got created.



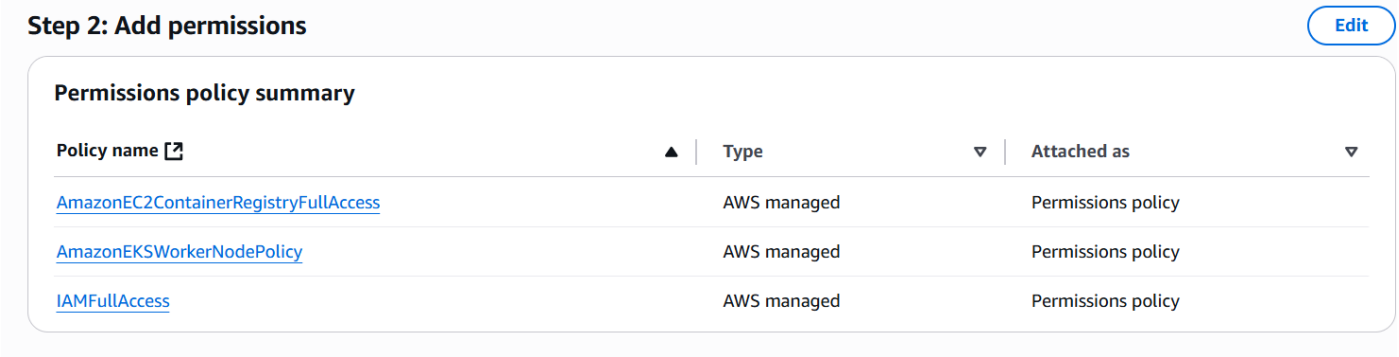
Deploy a Ngnix application on your Kubernetes cluster and it should be available across the cluster on port 80.

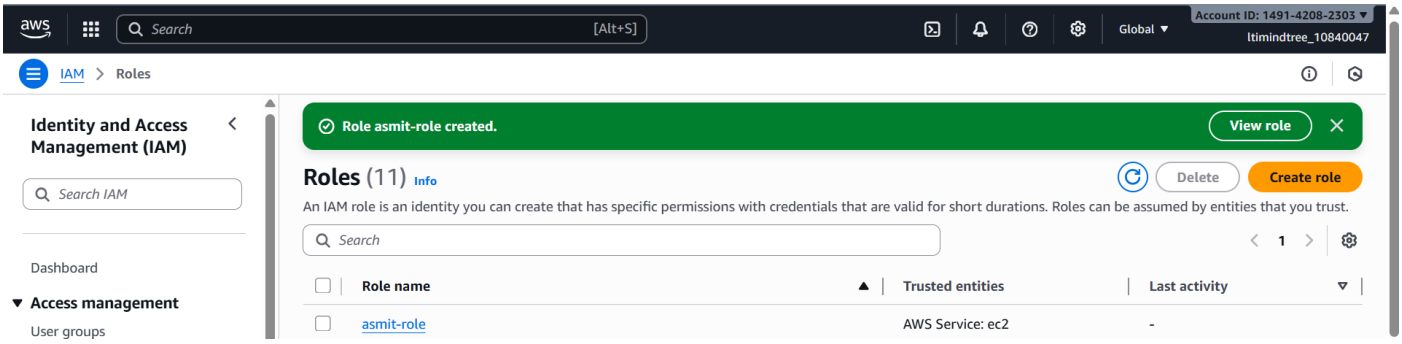
Created an ec2 instance and a user with policy **administrator access** and generated access key

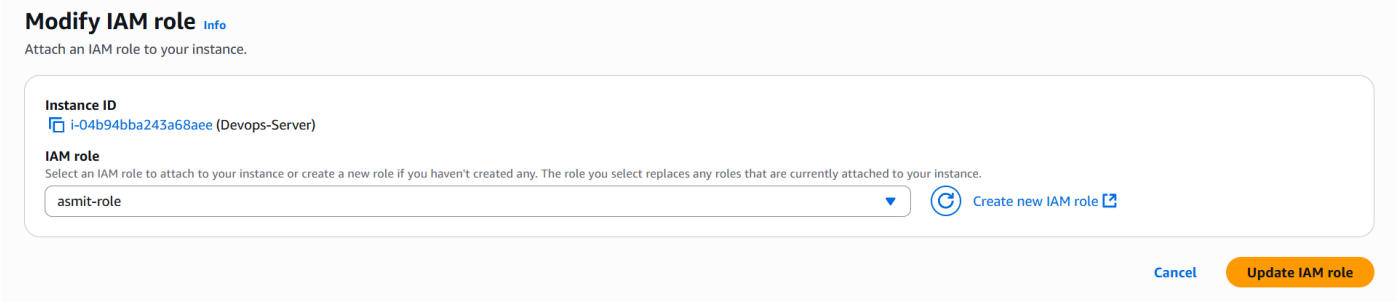




Created a role with below permissions and attached it to the instance.







Installed eks tool, kubectl and generated ssh-keys.

```
root@k8s:~
[root@k8s ~]# curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz
" | tar xz -C /tmp
[root@k8s ~]# sudo mv /tmp/eksctl /usr/local/bin
[root@k8s ~]# eksctl version
0.214.0
[root@k8s ~]#
```

```
root@k8s:~
[root@k8s ~]# curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-
release/release/stable.txt)/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 53.7M  100 53.7M    0     0  87.5M      0 --:--:-- --:--:-- --:--:-- 87.7M
[root@k8s ~]#  sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
[root@k8s ~]# kubectl version -client
error: extra arguments: [-client]
[root@k8s ~]# kubectl version --client
Client Version: v1.31.0
Kustomize Version: v5.4.2
[root@k8s ~]#
```

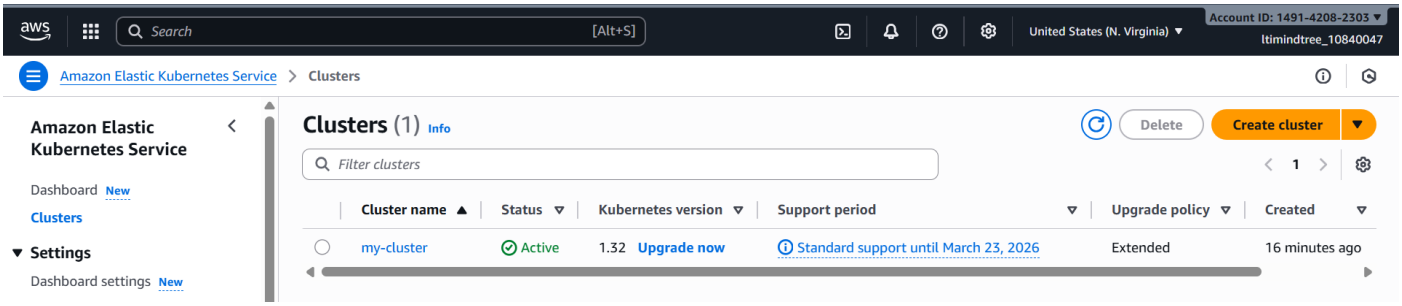
```
root@k8s:~
[root@k8s ~]# ssh-keygen
Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id_rsa): Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:g1yQHv6vYQyZbirgXPAkGsRoHvCsJC1EyHB30KsANOM root@k8s
The key's randomart image is:
+---[RSA 3072]----+
|%*. ooo.         |
|*@o. .+.         |
|*E=  o o.        |
|+*.. .+=         |
|o.=. .*.S        |
|o  o.. o..       |
|o..   o +.       |
| o.  o . ..      |
|   ..   ..       |
+----[SHA256]-----+
[root@k8s ~]#
```

Created a cluster and a nodegroup.

```
root@k8s:~
[root@k8s ~]# eksctl create cluster --name my-cluster --region us-east-1 --version 1.32 --vpc-public-subnets subnet-024ffa37df91f65c6
,subnet-0bec6ba741e27e1c0 --without-nodegroup
2025-09-05 05:47:59 [ℹ]  eksctl version 0.214.0
2025-09-05 05:47:59 [ℹ]  using region us-east-1
2025-09-05 05:47:59 [✔]  using existing VPC (vpc-096ee5d9060f72b59) and subnets (private:map[] public:map[us-east-1a:{subnet-0bec6ba7
41e27e1c0 us-east-1a 172.31.80.0/20 0 } us-east-1b:{subnet-024ffa37df91f65c6 us-east-1b 172.31.16.0/20 0 }])
2025-09-05 05:47:59 [!]  custom VPC/subnets will be used; if resulting cluster doesn't function as expected, make sure to review the
configuration of VPC/subnets
2025-09-05 05:47:59 [ℹ]  using Kubernetes version 1.32
2025-09-05 05:47:59 [ℹ]  creating EKS cluster "my-cluster" in "us-east-1" region with
2025-09-05 05:47:59 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-e
ast-1 --cluster=my-cluster'
2025-09-05 05:47:59 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "my-
cluster" in "us-east-1"
2025-09-05 05:47:59 [ℹ]  CloudWatch logging will not be enabled for cluster "my-cluster" in "us-east-1"
2025-09-05 05:47:59 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g
. all)} --region=us-east-1 --cluster=my-cluster'
2025-09-05 05:47:59 [ℹ]  default addons kube-proxy, coredns, metrics-server, vpc-cni were not specified, will install them as EKS add
ons
2025-09-05 05:47:59 [ℹ]
2 sequential tasks: { create cluster control plane "my-cluster",
    2 sequential sub-tasks: {
        1 task: { create addons },
        wait for control plane to become ready,
    }
}
2025-09-05 05:47:59 [ℹ]  building cluster stack "eksctl-my-cluster-cluster"
2025-09-05 05:48:00 [ℹ]  deploying stack "eksctl-my-cluster-cluster"
```

```
root@k8s:~
[root@k8s ~]# eksctl create nodegroup \
  --cluster my-cluster \
  --region us-east-1 \
  --name my-node-group \
  --node-ami-family Ubuntu2204 \
  --node-type t2.micro \
  --subnet-ids subnet-024ffa37df91f65c6,subnet-0bec6ba741e27e1c0 \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
  --ssh-access \
  --ssh-public-key /root/.ssh/id_rsa.pub
2025-09-05 05:59:44 [ℹ]  will use version 1.32 for new nodegroup(s) based on control plane version
2025-09-05 05:59:45 [ℹ]  nodegroup "my-node-group" will use "ami-0f31f4dfa334d856c" [Ubuntu2204/1.32]
2025-09-05 05:59:45 [ℹ]  using SSH public key "/root/.ssh/id_rsa.pub" as "eksctl-my-cluster-nodegroup-my-node-group-87:bf:b2:5f:34:6c
:19:c8:1b:31:23:b1:63:0c:49:06"
2025-09-05 05:59:46 [ℹ]  1 nodegroup (my-node-group) was included (based on the include/exclude rules)
2025-09-05 05:59:46 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "my-cluster"
2025-09-05 05:59:46 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "my-node-group" } }
}
2025-09-05 05:59:46 [ℹ]  checking cluster stack for missing resources
2025-09-05 05:59:46 [ℹ]  cluster stack has all required resources
2025-09-05 05:59:46 [ℹ]  building managed nodegroup stack "eksctl-my-cluster-nodegroup-my-node-group"
2025-09-05 05:59:46 [ℹ]  deploying stack "eksctl-my-cluster-nodegroup-my-node-group"
2025-09-05 05:59:46 [ℹ]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-node-group"
```

We can see here a cluster is created

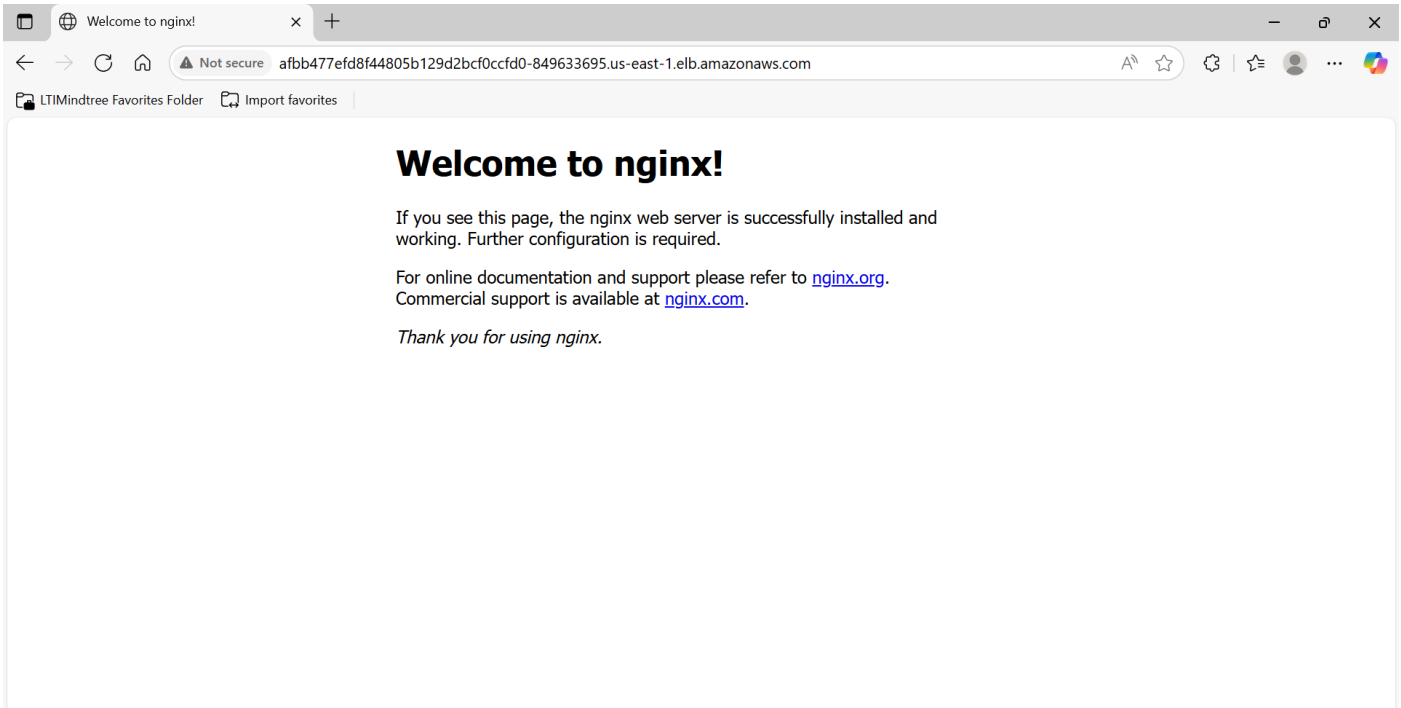


Wrote the yaml code to deploy a nginx application on port 80

```
root@k8s:/k8s
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.14.2
      ports:
        - containerPort: 80
```

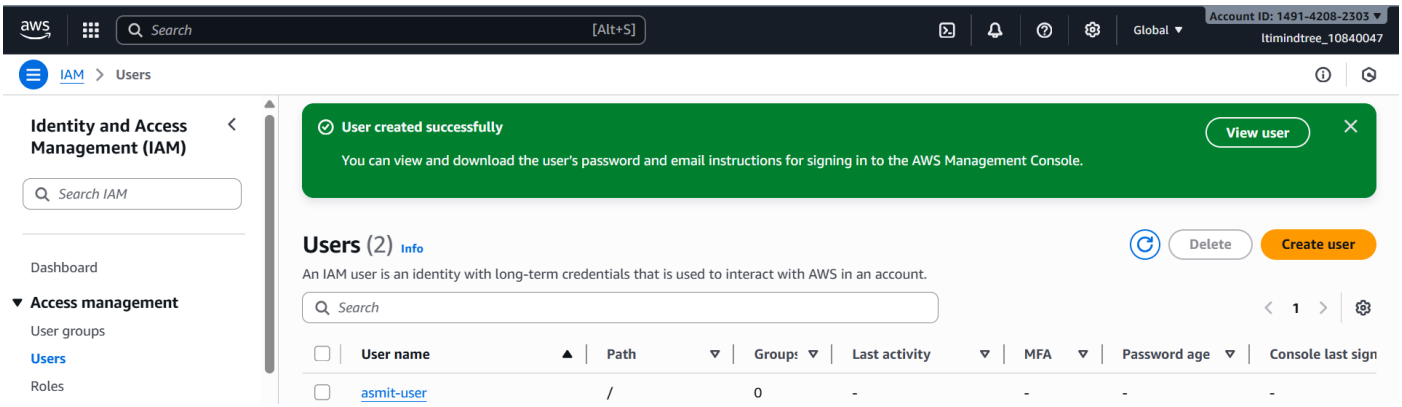
```
root@k8s:/k8s
[root@k8s k8s]# vim mypod.yaml
[root@k8s k8s]# kubectl apply -f mypod.yaml
pod/nginx created
[root@k8s k8s]# kubectl expose pod nginx --type=LoadBalancer --port=80
service/nginx exposed
[root@k8s k8s]# kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP                                                          PORT(S)
AGE
kubernetes   ClusterIP      10.100.0.1       <none>                                                               443/TCP
27m
nginx        LoadBalancer   10.100.130.235   afbb477efd8f44805b129d2bcf0ccfd0-849633695.us-east-1.elb.amazonaws.com   80:30649/TCP
18s
[root@k8s k8s]# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          93s
[root@k8s k8s]# kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP                                                          PORT(S)
AGE
kubernetes   ClusterIP      10.100.0.1       <none>                                                               443/TCP
28m
nginx        LoadBalancer   10.100.130.235   afbb477efd8f44805b129d2bcf0ccfd0-849633695.us-east-1.elb.amazonaws.com   80:30649/TCP
62s
```

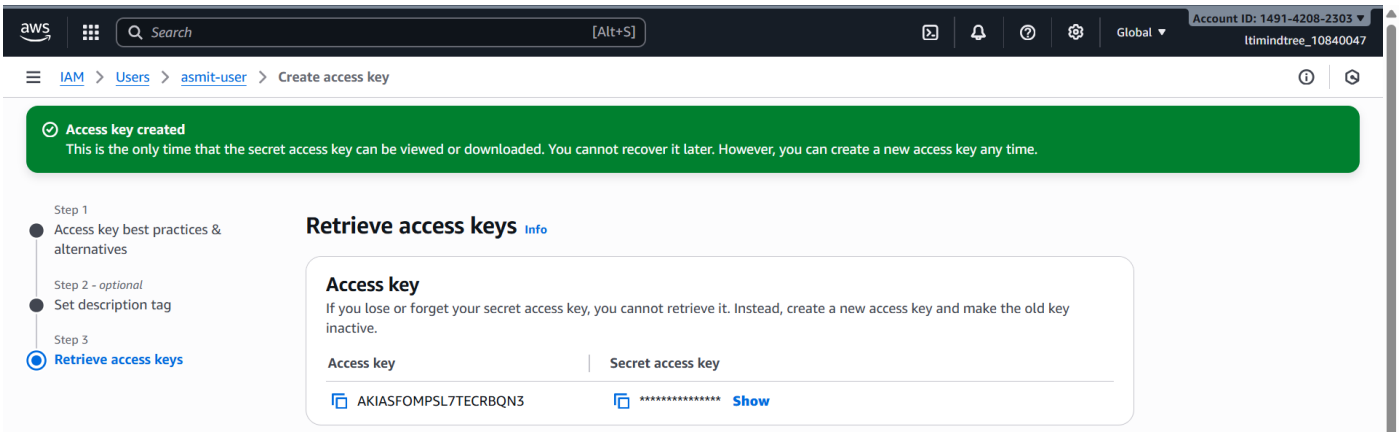
Copied the external Ip link and ran it with :80 (I allowed port 80 in my security group)



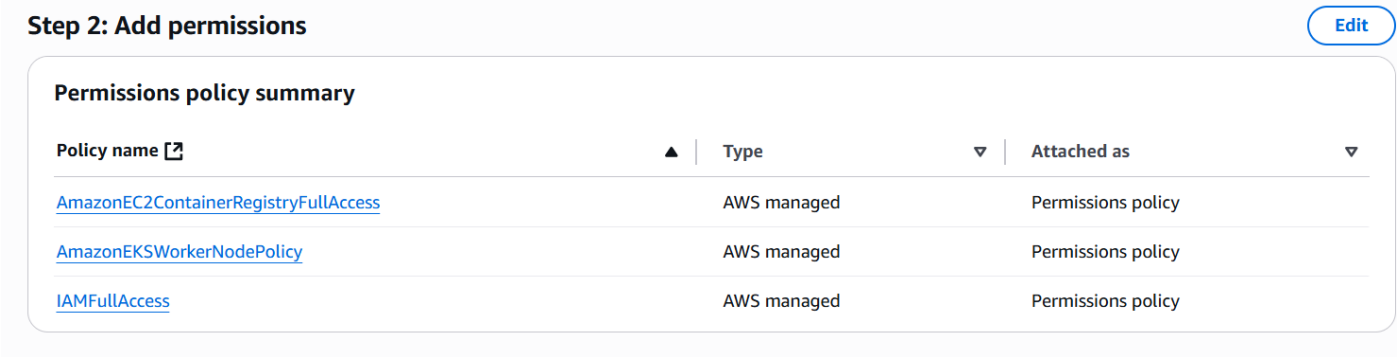
Deploy a web application in the kubernetes pod and create a replica set. In any case load is going to increase on your replica set, increase the number of replicas of the pods.

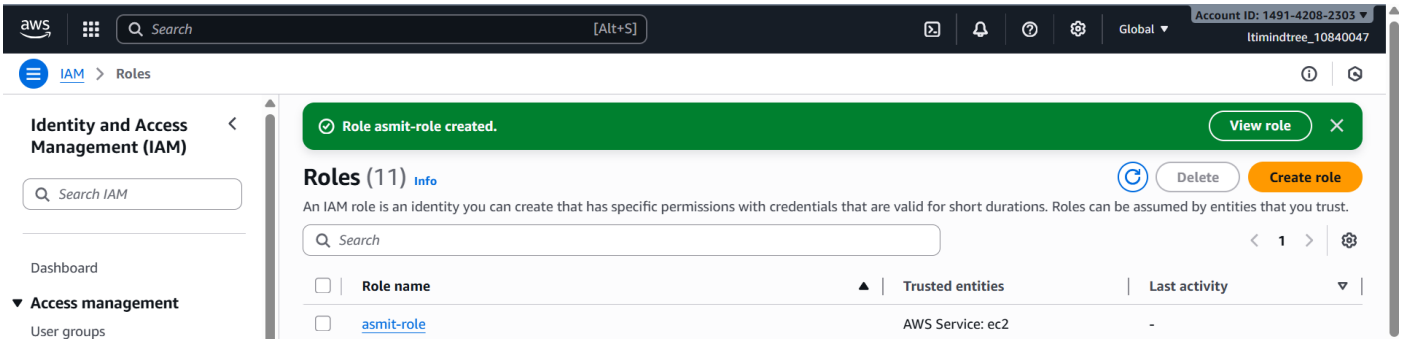
Created an ec2 instance and a user with policy **administrator access** and generated access key

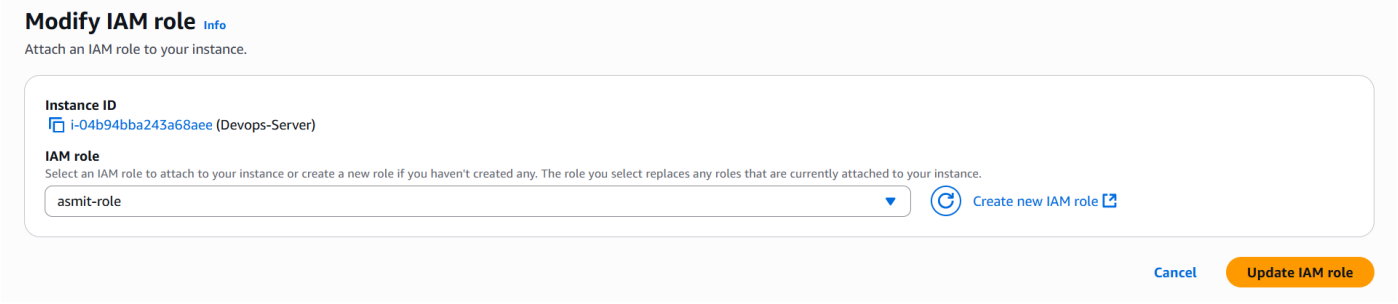




Created an role with below permissions and attached it to the instance.







Installed eks tool, kubectl and generated ssh-keys.

```
root@k8s:~
[root@k8s ~]# curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz
" | tar xz -C /tmp
[root@k8s ~]# sudo mv /tmp/eksctl /usr/local/bin
[root@k8s ~]# eksctl version
0.214.0
[root@k8s ~]#
```

```
root@k8s:~
[root@k8s ~]# curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-
release/release/stable.txt)/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 53.7M  100 53.7M    0     0  87.5M      0 --:--:-- --:--:-- --:--:-- 87.7M
[root@k8s ~]#  sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
[root@k8s ~]# kubectl version -client
error: extra arguments: [-client]
[root@k8s ~]# kubectl version --client
Client Version: v1.31.0
Kustomize Version: v5.4.2
[root@k8s ~]#
```

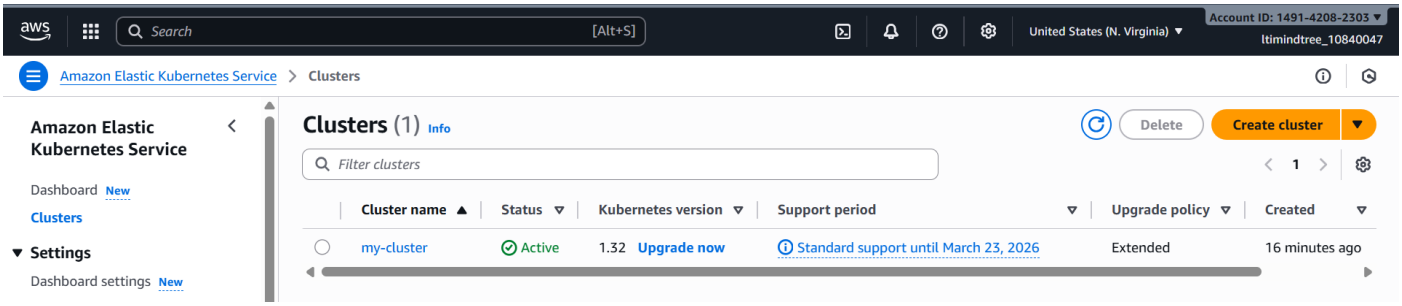
```
root@k8s:~
[root@k8s ~]# ssh-keygen
Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id_rsa): Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:g1yQHv6vYQyZbirgXPAkGsRoHvCsJC1EyHB30KsANOM root@k8s
The key's randomart image is:
+---[RSA 3072]----+
|%*. ooo.         |
|*@o. .+.         |
|*E=  o o.        |
|+*.. .+=         |
|o.=. .*.S        |
|o  o.. o..       |
|o..   o +.       |
| o.  o . ..      |
|   ..   ..       |
+----[SHA256]-----+
[root@k8s ~]#
```

Created a cluster and a nodegroup.

```
root@k8s:~
[root@k8s ~]# eksctl create cluster --name my-cluster --region us-east-1 --version 1.32 --vpc-public-subnets subnet-024ffa37df91f65c6
,subnet-0bec6ba741e27e1c0 --without-nodegroup
2025-09-05 05:47:59 [ℹ]  eksctl version 0.214.0
2025-09-05 05:47:59 [ℹ]  using region us-east-1
2025-09-05 05:47:59 [✔]  using existing VPC (vpc-096ee5d9060f72b59) and subnets (private:map[] public:map[us-east-1a:{subnet-0bec6ba7
41e27e1c0 us-east-1a 172.31.80.0/20 0 } us-east-1b:{subnet-024ffa37df91f65c6 us-east-1b 172.31.16.0/20 0 }])
2025-09-05 05:47:59 [!]  custom VPC/subnets will be used; if resulting cluster doesn't function as expected, make sure to review the
configuration of VPC/subnets
2025-09-05 05:47:59 [ℹ]  using Kubernetes version 1.32
2025-09-05 05:47:59 [ℹ]  creating EKS cluster "my-cluster" in "us-east-1" region with
2025-09-05 05:47:59 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-e
ast-1 --cluster=my-cluster'
2025-09-05 05:47:59 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "my-
cluster" in "us-east-1"
2025-09-05 05:47:59 [ℹ]  CloudWatch logging will not be enabled for cluster "my-cluster" in "us-east-1"
2025-09-05 05:47:59 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g
. all)} --region=us-east-1 --cluster=my-cluster'
2025-09-05 05:47:59 [ℹ]  default addons kube-proxy, coredns, metrics-server, vpc-cni were not specified, will install them as EKS add
ons
2025-09-05 05:47:59 [ℹ]
2 sequential tasks: { create cluster control plane "my-cluster",
    2 sequential sub-tasks: {
        1 task: { create addons },
        wait for control plane to become ready,
    }
}
2025-09-05 05:47:59 [ℹ]  building cluster stack "eksctl-my-cluster-cluster"
2025-09-05 05:48:00 [ℹ]  deploying stack "eksctl-my-cluster-cluster"
```

```
root@k8s:~
[root@k8s ~]# eksctl create nodegroup \
  --cluster my-cluster \
  --region us-east-1 \
  --name my-node-group \
  --node-ami-family Ubuntu2204 \
  --node-type t2.micro \
  --subnet-ids subnet-024ffa37df91f65c6,subnet-0bec6ba741e27e1c0 \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
  --ssh-access \
  --ssh-public-key /root/.ssh/id_rsa.pub
2025-09-05 05:59:44 [ℹ]  will use version 1.32 for new nodegroup(s) based on control plane version
2025-09-05 05:59:45 [ℹ]  nodegroup "my-node-group" will use "ami-0f31f4dfa334d856c" [Ubuntu2204/1.32]
2025-09-05 05:59:45 [ℹ]  using SSH public key "/root/.ssh/id_rsa.pub" as "eksctl-my-cluster-nodegroup-my-node-group-87:bf:b2:5f:34:6c
:19:c8:1b:31:23:b1:63:0c:49:06"
2025-09-05 05:59:46 [ℹ]  1 nodegroup (my-node-group) was included (based on the include/exclude rules)
2025-09-05 05:59:46 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "my-cluster"
2025-09-05 05:59:46 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "my-node-group" } }
}
2025-09-05 05:59:46 [ℹ]  checking cluster stack for missing resources
2025-09-05 05:59:46 [ℹ]  cluster stack has all required resources
2025-09-05 05:59:46 [ℹ]  building managed nodegroup stack "eksctl-my-cluster-nodegroup-my-node-group"
2025-09-05 05:59:46 [ℹ]  deploying stack "eksctl-my-cluster-nodegroup-my-node-group"
2025-09-05 05:59:46 [ℹ]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-node-group"
```

We can see here a cluster is created



Till here I have already done all these things for the above question that was to deploy a nginx application on Kubernetes.

So, after this I created 3 yaml files – deployment, service and an autoscaler.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp-deployment
  labels:
    app: webapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: webapp
        image: nginx:1.14.2
        ports:
        - containerPort: 80

        resources:
          requests:
            cpu: 100m
          limits:
            cpu: 200m
~
```

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: webapp-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: webapp-deployment
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
~
```

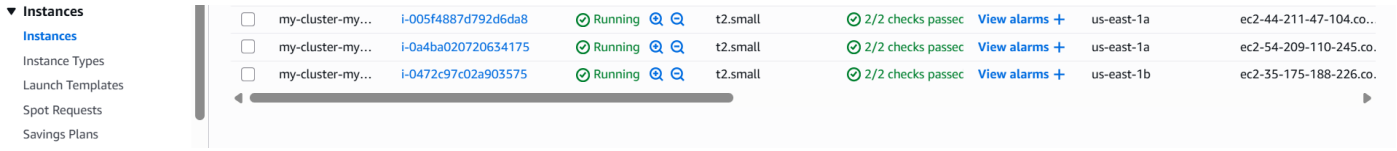
```yaml
apiVersion: v1
kind: Service
metadata:
  name: webapp-service
spec:
  selector:
    app.kubernetes.io/name: webapp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Applied all three of these.

```
[root@k8s k8s]# vim deployment.yaml
[root@k8s k8s]# vim service.yaml
[root@k8s k8s]# vim hpa.yaml
[root@k8s k8s]# vim hpa.yaml
[root@k8s k8s]# kubectl apply -f deployment.yaml
deployment.apps/webapp-deployment created
[root@k8s k8s]# kubectl apply -f service.yaml
service/webapp-service created
[root@k8s k8s]# kubectl apply -f hpa.yaml
horizontalpodautoscaler.autoscaling/webapp-hpa created
[root@k8s k8s]#
[root@k8s k8s]#
```

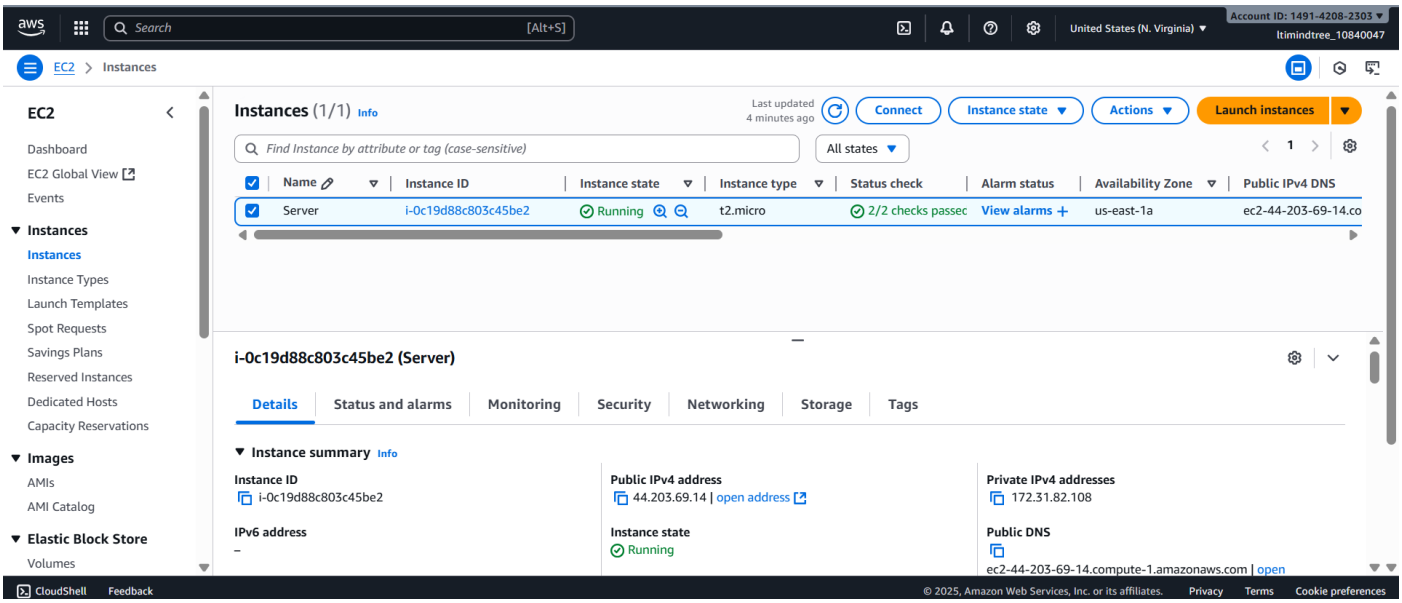
```
[root@k8s k8s]# kubectl get deployments
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
webapp-deployment    2/2     2            2           2m8s
[root@k8s k8s]# kubectl get pod -o wide
NAME                                   READY   STATUS    RESTARTS   AGE     IP              NODE                                       NOMINATED NODE   READINESS GATES
webapp-deployment-855bc84fb8-kq2p5     1/1     Running   0          2m18s   172.31.90.2     ip-172-31-81-187.ec2.internal   <none>           <none>
webapp-deployment-855bc84fb8-sldd2     1/1     Running   0          2m18s   172.31.92.106   ip-172-31-91-166.ec2.internal   <none>           <none>
[root@k8s k8s]# kubectl get rs
NAME                                 DESIRED   CURRENT   READY   AGE
webapp-deployment-855bc84fb8   2         2         2       2m32s
[root@k8s k8s]# kubectl get svc
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP                                                              PORT(S)        AGE
kubernetes      ClusterIP      10.100.0.1       <none>                                                                   443/TCP        46m
webapp-service  LoadBalancer   10.100.147.194   a7b97c810f25a4efa982b6a3cebb229d-1321803495.us-east-1.elb.amazonaws.com   80:31399/TCP   2m30s
[root@k8s k8s]# kubectl get hpa
NAME         REFERENCE                     TARGETS       MINPODS   MAXPODS   REPLICAS   AGE
webapp-hpa   Deployment/webapp-deployment   cpu: 0%/50%   2         10        2          2m36s
```

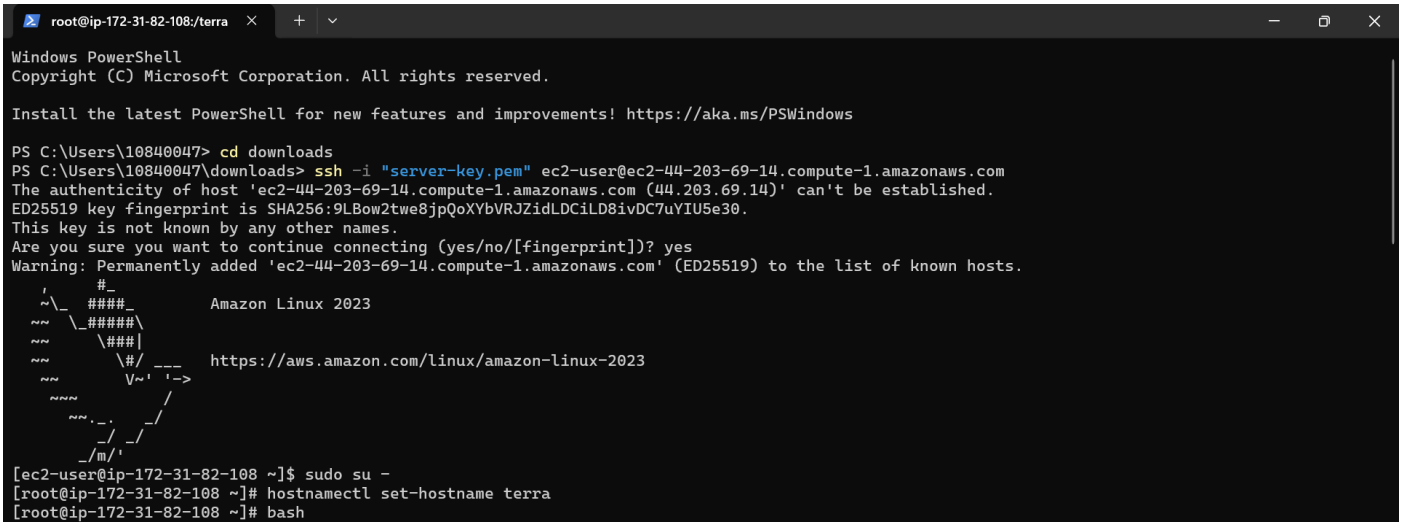
New instances got created



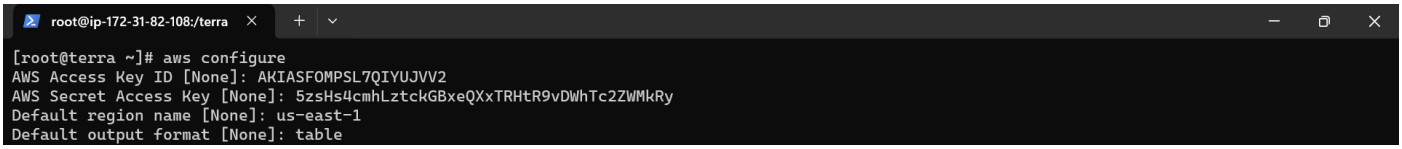
Create an EC2 instance using of terraform us-east-1a zone in North Virginia region and create of your security group which name is web_access and allow port 22 and 80 in the ingress. This security group should be attached with your EC2 instance.

Launched an instance and connected to the terminal.





Configured aws on the terminal



Installed Terraform.

```
root@ip-172-31-82-108:~

[root@terra ~]# terraform -v
Terraform v1.13.1
on linux_amd64
[root@terra ~]#
```

Wrote the terraform code

```
root@ip-172-31-82-108:/terra

provider "aws" {
  region     = "us-east-1"
}

#security group
resource "aws_security_group" "web_access" {
        name = "web_access"
        description = "allow ssh and http"

        ingress {
                from_port = 80
                to_port = 80
                protocol = "tcp"
                cidr_blocks = ["0.0.0.0/0"]
        }

        ingress {
                from_port = 22
                to_port = 22
                protocol = "tcp"
                cidr_blocks = ["0.0.0.0/0"]
        }

        egress {
                from_port = 0
                to_port = 0
                protocol = "-1"
                cidr_blocks = ["0.0.0.0/0"]
        }


}
```

```
resource "aws_instance" "web-server" {
  ami           = "ami-00ca32bbc84273381"
  availability_zone = "us-east-1a"
  instance_type = "t2.micro"
  security_groups = ["${aws_security_group.web_access.name}"]
  tags = {
    Name  = "asmit-terra"
    Stage = "test"
    Location = "Mumbai"
  }

}
```

Initialized and validated Terraform

```
root@ip-172-31-82-108:/terra

[root@terra ~]# cd /terra
[root@terra terra]# vim prov.tf
[root@terra terra]# terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.12.0...
- Installed hashicorp/aws v6.12.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[root@terra terra]#
```

```
root@ip-172-31-82-108:/terra

[root@terra terra]# terraform validate
Success! The configuration is valid.

[root@terra terra]#
```

Ran terraform apply command and it's successful.

```
root@ip-172-31-82-108:/terra

            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 80
              # (1 unchanged attribute hidden)
          },
        ]
      + name                   = "web_access"
      + name_prefix            = (known after apply)
      + owner_id               = (known after apply)
      + region                 = "us-east-1"
      + revoke_rules_on_delete = false
      + tags_all               = (known after apply)
      + vpc_id                 = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_security_group.web_access: Creating...
aws_security_group.web_access: Creation complete after 2s [id=sg-056feeaec09e6db0b]
aws_instance.web-server: Creating...
aws_instance.web-server: Still creating... [00m10s elapsed]
aws_instance.web-server: Still creating... [00m20s elapsed]
aws_instance.web-server: Still creating... [00m30s elapsed]
aws_instance.web-server: Creation complete after 32s [id=i-0897dc71e48269427]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[root@terra terra]#
```

An ec2 instance and a security group with name "web_access" is created.

EC2 instance:

Instances (2) Info

Last updated less than a minute ago | Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states | 1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Server	i-0c19d88c803c45be2	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-44-203-69-14.co
asmit-terra	i-0897dc71e48269427	Running	t2.micro	Initializing	View alarms +	us-east-1a	ec2-13-222-30-186.c

Security Group:

–	sg-056feeaec09e6db0b	web_access	vpc-096ee5d9060f72b59	allow ssh and http