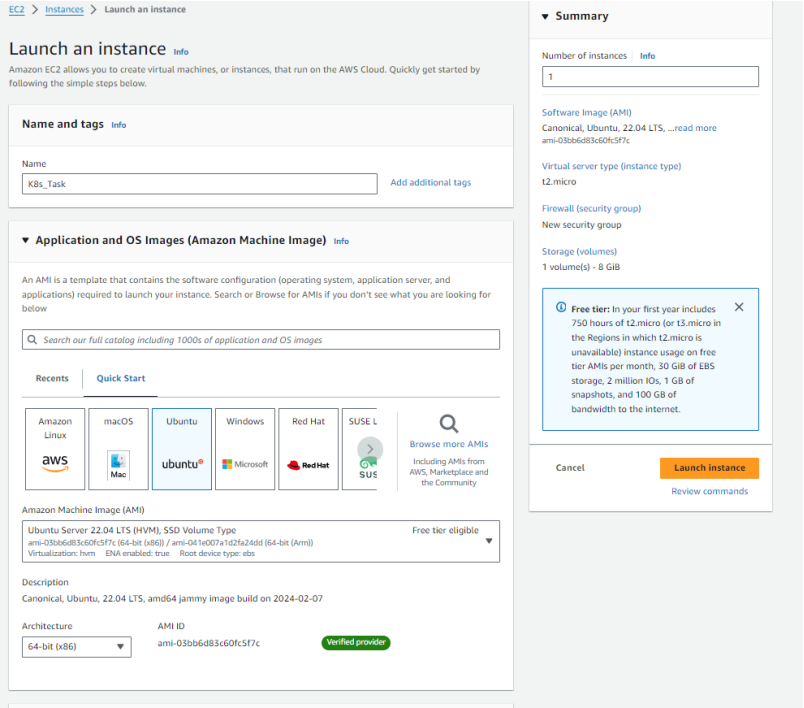
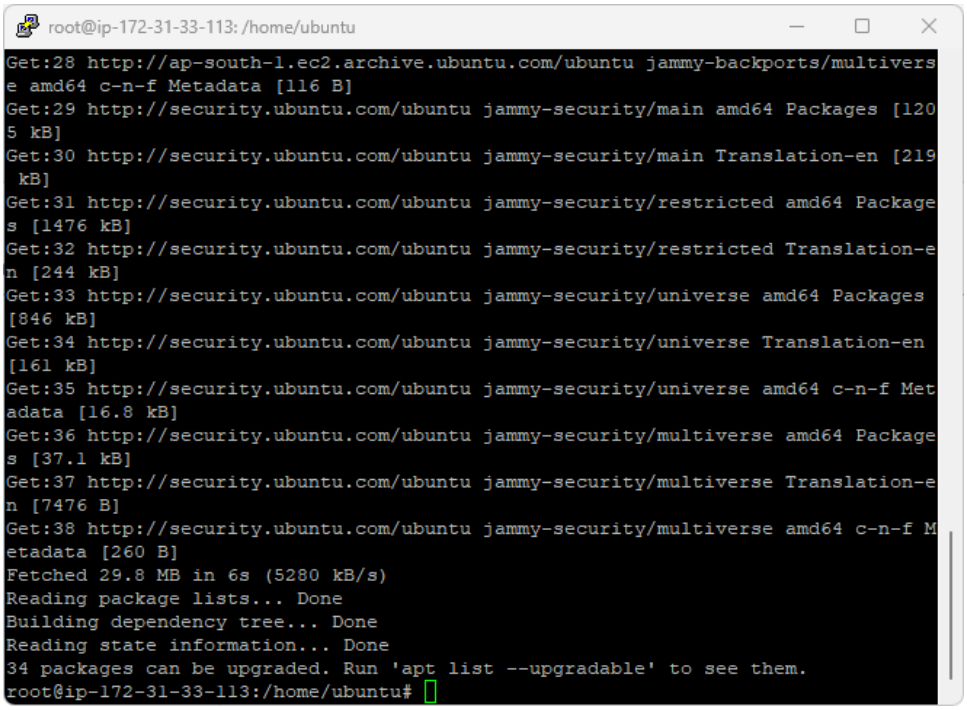
Create the K8s EKS,further you have to do the deployment of Nginx application

Step 1: Create a Ec2 Instance:

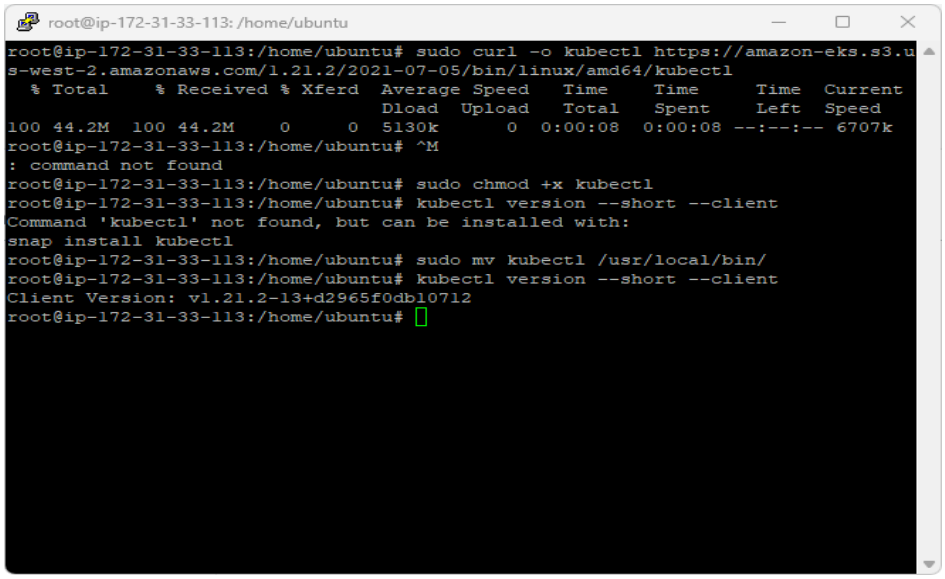


Step 2: Using Putty connect the instance and update.



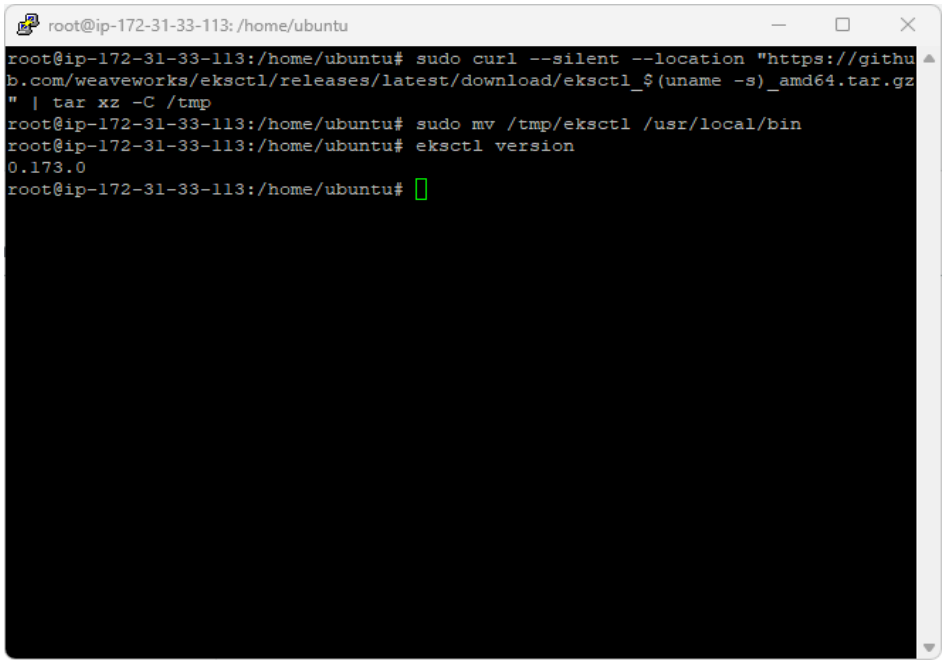
Step 3: To install Kubernetes, Run the following command one by one.

```
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
```

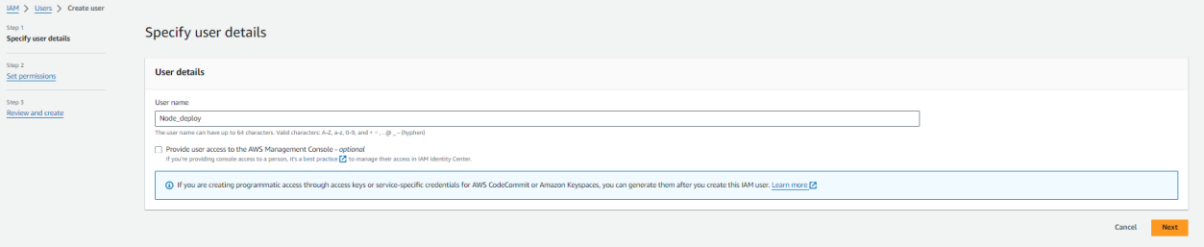


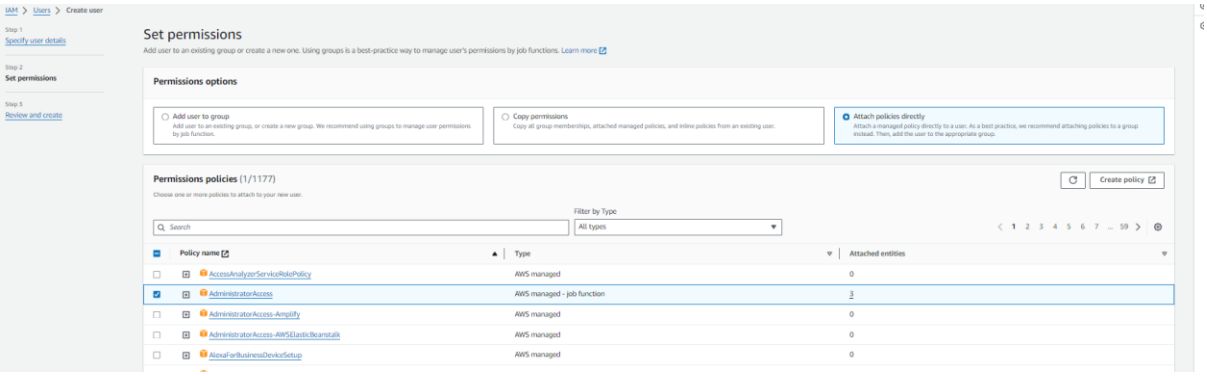
Step 4: To install eksctl, Run the following command one by one.

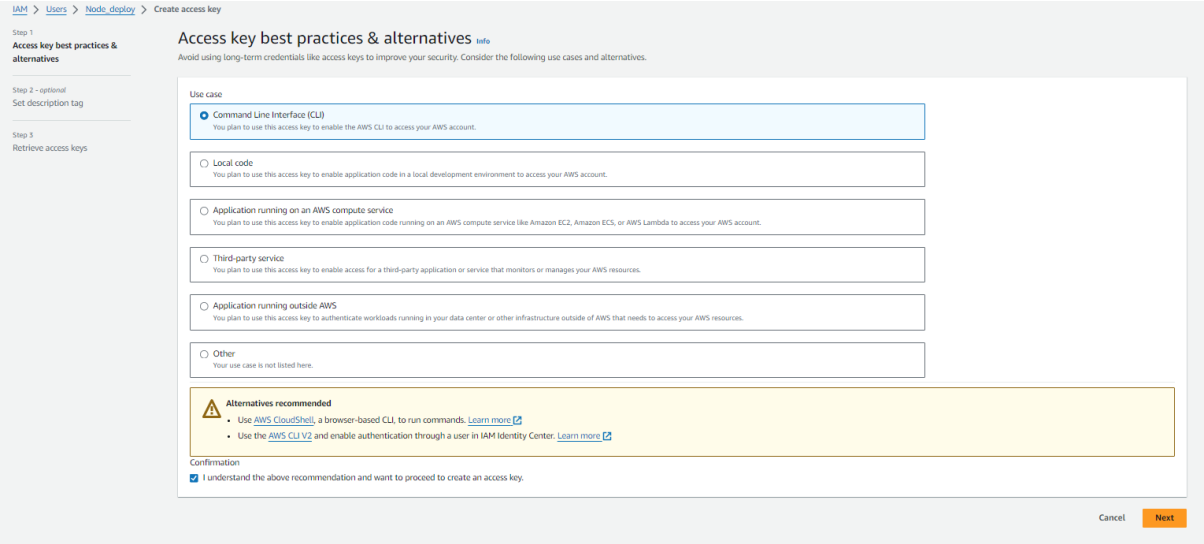
```
curl --silent -location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
```

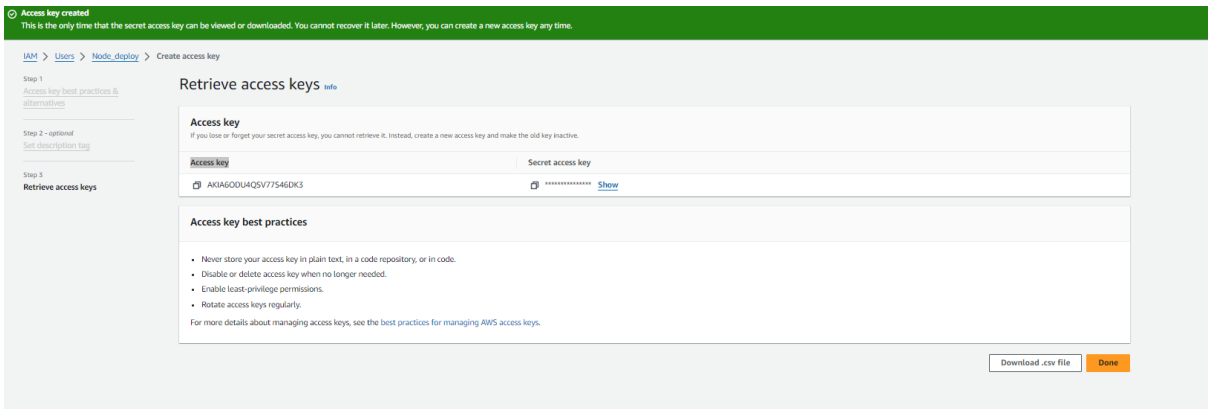


Step 5: Create New User to connect AWS from Ec2 Instance.









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

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

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

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

```
root@ip-172-31-33-113: /home/ubuntu/task
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.yml
root@ip-172-31-33-113:/home/ubuntu/task# kubectl create -f config.yml
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
              NOMINATED NODE   READINESS GATES
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:c26ae7472d624ba1fafd296e73cecc4f93f853088e6a9c
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
ype=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