Are you exploring Kubernetes and looking to set up your first Amazon EKS cluster? Here's a straightforward guide to get you started with AWS Elastic Kubernetes Service (EKS). Whether you're a beginner or looking for a quick reference, this post has got you covered!

Step 1: Create an EKS Management Host

Start by launching an **Ubuntu EC2 instance** (a `t2.micro` works perfectly) on AWS. Then, install the required tools:

1- **Install kubectl**

```
#curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.6/2024-11-15/bin/
linux/amd64/kubectl

#chmod +x ./kubectl

#sudo mv ./kubectl /usr/local/bin

#kubectl version --short --client
```

2- **Install AWS CLI (Latest Version):**

```
#sudo apt update && sudo apt install unzip -y

#curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"

#unzip awscliv2.zip

#sudo ./aws/install

#aws --version
```

3- **Install eksctl:**

```
#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 2: Configure IAM Role for EKS Management Host

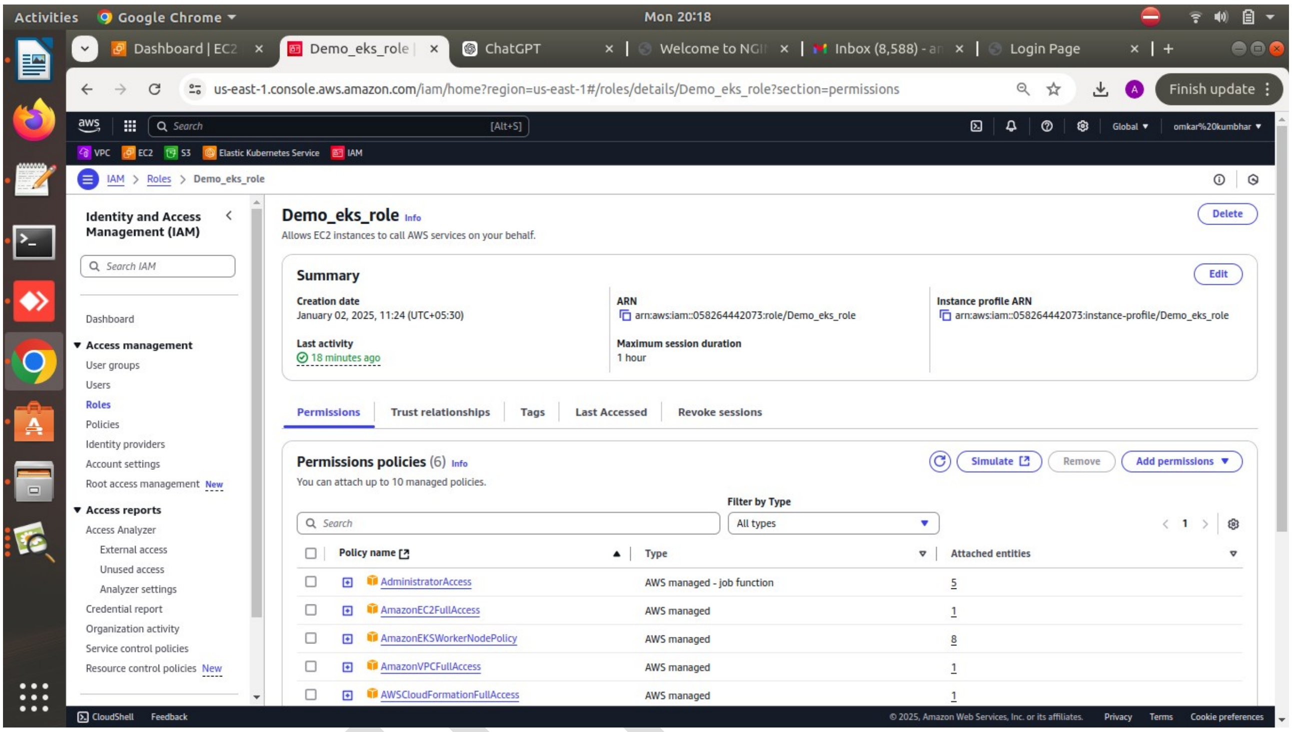
To manage your EKS cluster, create an IAM role with the necessary permissions:

- IAM: Full Access
- VPC: Full Access
- EC2: Full Access
- CloudFormation: Full Access
- Administrator Access

Role Name: `Demo-eks-role`

Attach Role to EC2 Instance:

1. Navigate to **EC2 Dashboard** → Select your instance → **Security** → **Modify IAM Role**.
2. Attach the `Demo-eks-role`.



Demo_eks_role Info

Allows EC2 instances to call AWS services on your behalf.

Delete

Summary

Edit

Creation date: January 02, 2025, 11:24 (UTC+05:30)

Last activity: 18 minutes ago

ARN: arn:aws:iam::058264442073:role/Demo_eks_role

Maximum session duration: 1 hour

Instance profile ARN: arn:aws:iam::058264442073:instance-profile/Demo_eks_role

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (6) Info

You can attach up to 10 managed policies.

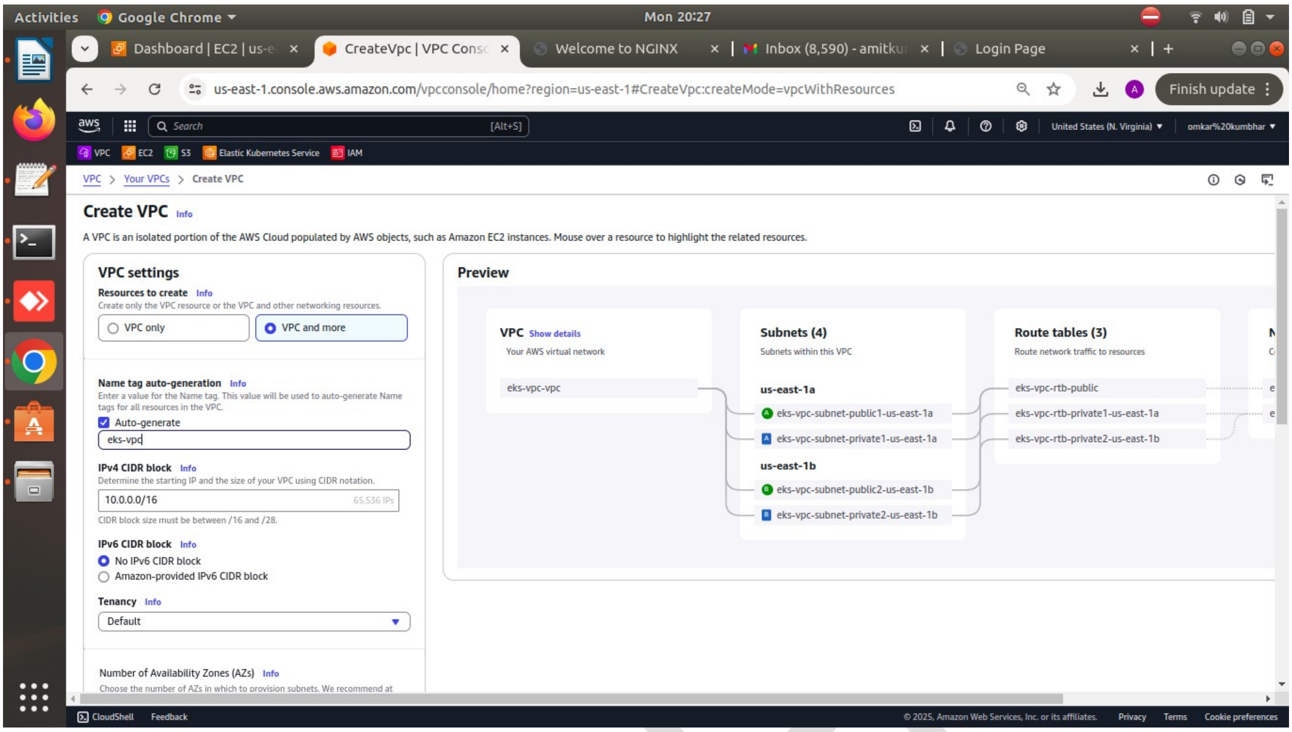
Simulate | Remove | Add permissions

Filter by Type: All types

□	Policy name	Type	Attached entities
□	AdministratorAccess	AWS managed - job function	5
□	AmazonEC2FullAccess	AWS managed	1
□	AmazonEKSWorkerNodePolicy	AWS managed	8
□	AmazonVPCFullAccess	AWS managed	1
□	AWSCloudFormationFullAccess	AWS managed	1

Step 3: Create a VPC for Your EKS Cluster

Go to the **VPC Dashboard** and create a new VPC for your cluster. Configure it based on your requirements, such as CIDR block, subnets, and route tables.


Activities
Google Chrome
Mon 20:27
Dashboard | EC2 | us-e
CreateVpc | VPC Conso
Welcome to NGINX
Inbox (8,590) - amitku
Login Page
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateVpc:createMode=vpcWithResources
Finish update
Search [Alt+S]
United States (N. Virginia)
omkar%20kumbhar
VPC EC2 S3 Elastic Kubernetes Service IAM
VPC > Your VPCs > Create VPC
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create VPC Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances. Mouse over a resource to highlight the related resources.

VPC settings

Resources to create Info

Create only the VPC resource or the VPC and other networking resources.

○ VPC only ● VPC and more

Name tag auto-generation Info

Enter a value for the Name tag. This value will be used to auto-generate Name tags for all resources in the VPC.

☑ Auto-generate

eks-vpc

IPv4 CIDR block Info

Determine the starting IP and the size of your VPC using CIDR notation.

10.0.0.0/16 65,536 IPs

CIDR block size must be between /16 and /28.

IPv6 CIDR block Info

● No IPv6 CIDR block

○ Amazon-provided IPv6 CIDR block

Tenancy Info

Default

Number of Availability Zones (AZs) Info

Choose the number of AZs in which to provision subnets. We recommend at

Preview

VPC Show details

Your AWS virtual network

eks-vpc-vpc

Subnets (4)

Subnets within this VPC

us-east-1a

eks-vpc-subnet-public1-us-east-1a

eks-vpc-subnet-private1-us-east-1a

us-east-1b

eks-vpc-subnet-public2-us-east-1b

eks-vpc-subnet-private2-us-east-1b

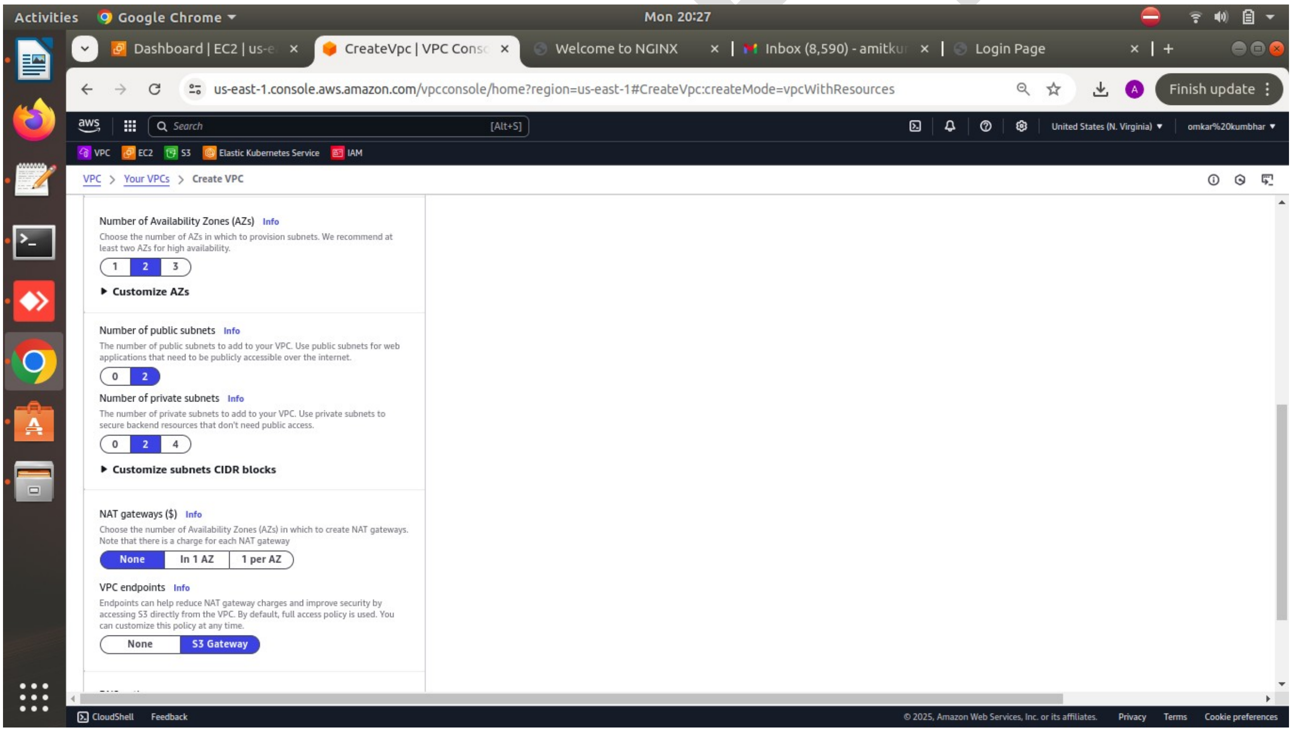
Route tables (3)

Route network traffic to resources

eks-vpc-rtb-public

eks-vpc-rtb-private1-us-east-1a

eks-vpc-rtb-private2-us-east-1b


Activities
Google Chrome
Mon 20:27
Dashboard | EC2 | us-e
CreateVpc | VPC Conso
Welcome to NGINX
Inbox (8,590) - amitku
Login Page
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateVpc:createMode=vpcWithResources
Finish update
Search [Alt+S]
United States (N. Virginia)
omkar%20kumbhar
VPC EC2 S3 Elastic Kubernetes Service IAM
VPC > Your VPCs > Create VPC
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Number of Availability Zones (AZs) Info

Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability.

1 | [✓ 2] | 3

▸ Customize AZs

Number of public subnets Info

The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.

0 | [✓ 2]

Number of private subnets Info

The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access.

0 | [✓ 2] | 4

▸ Customize subnets CIDR blocks

NAT gateways ($) Info

Choose the number of Availability Zones (AZs) in which to create NAT gateways. Note that there is a charge for each NAT gateway

[✓ None] | In 1 AZ | 1 per AZ

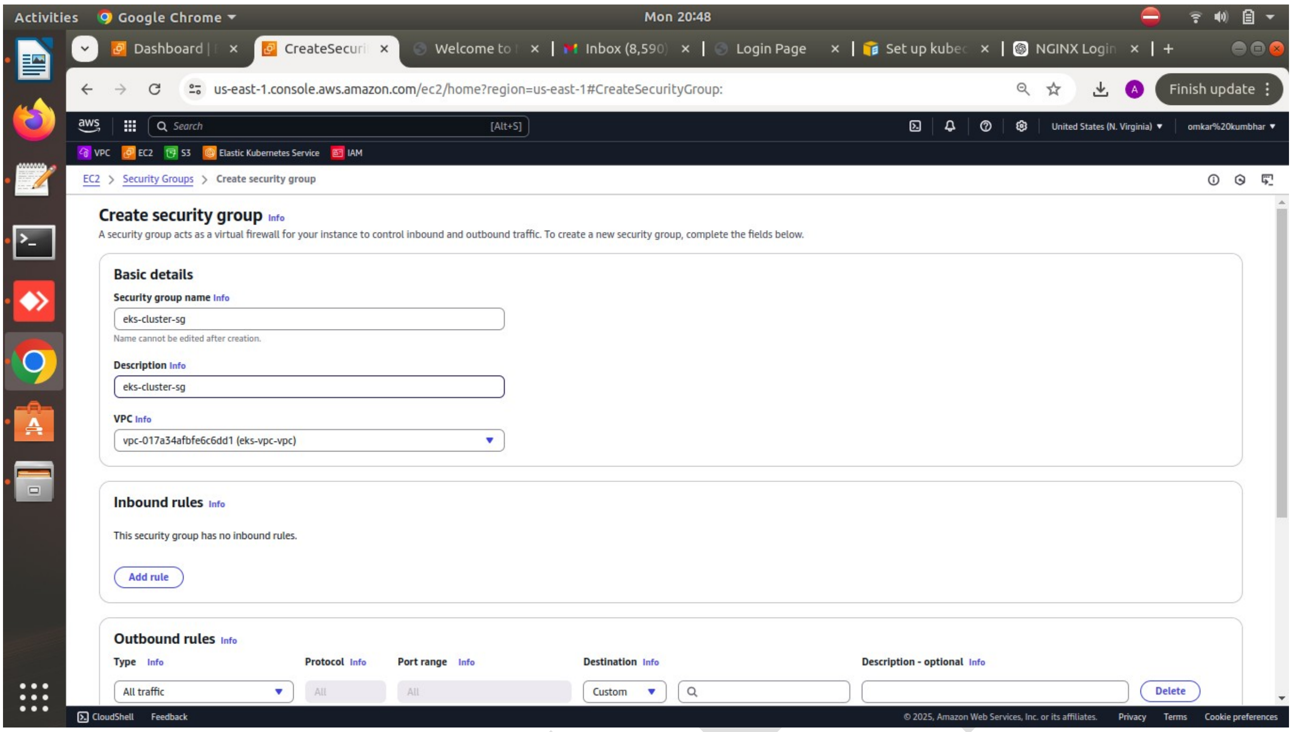
VPC endpoints Info

Endpoints can help reduce NAT gateway charges and improve security by accessing S3 directly from the VPC. By default, full access policy is used. You can customize this policy at any time.

None | [✓ S3 Gateway]

Step 4: Create a Security Group for the Cluster

Set up a security group for your EKS cluster and configure **inbound rules** to allow the required traffic (e.g., SSH, HTTP, HTTPS).



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateSecurityGroup:

EC2 > Security Groups > Create security group

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: eks-cluster-sg

Name cannot be edited after creation.

Description: eks-cluster-sg

VPC: vpc-017a34afbfe6c6dd1 (eks-vpc-vpc)

Inbound rules

This security group has no inbound rules.

Add rule

Outbound rules

Type	Protocol	Port range	Destination		Description - optional	
All traffic	All	All	Custom			Delete

© 2025, Amazon Web Services, Inc. or its affiliates.

Step 5: Create an EKS Cluster

Use `eksctl` to create your EKS cluster. Here's an example command for the us region (us-east-`1`):

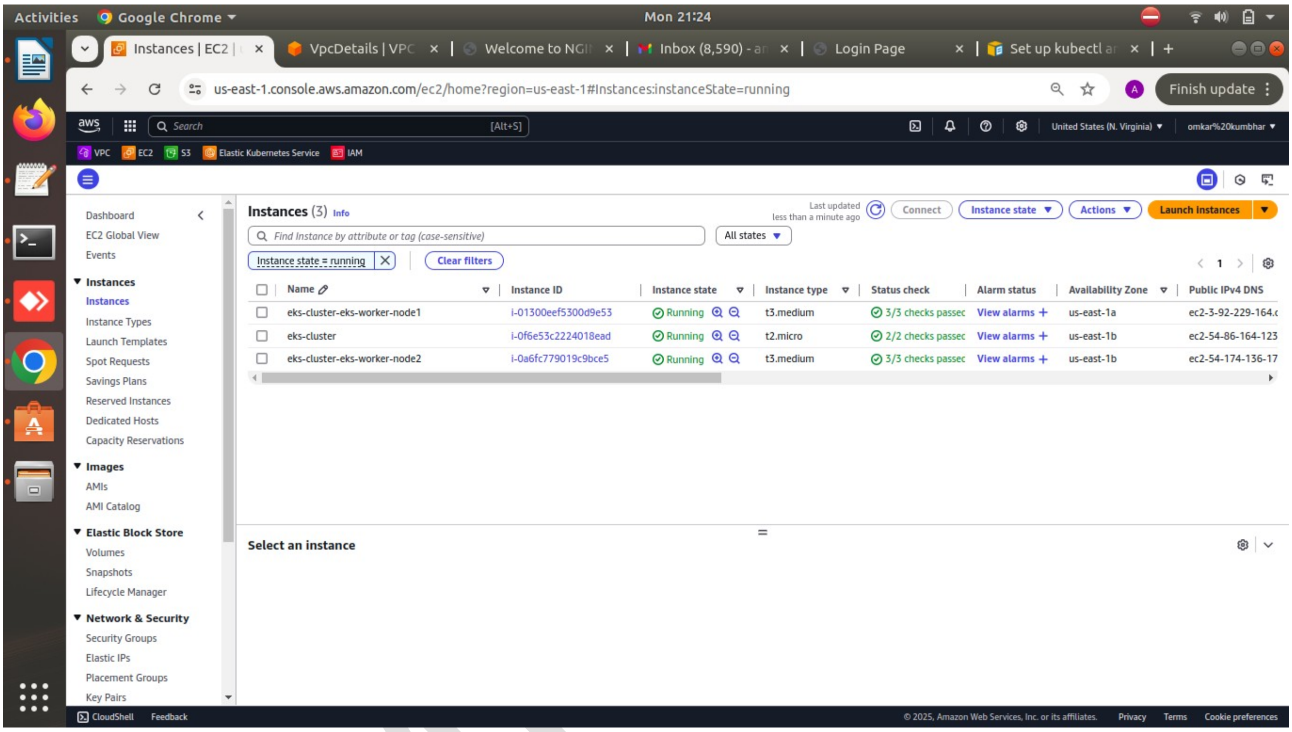
```
eksctl create cluster \
  --name eks-cluster \
  --region us-east-1 \
  --nodegroup-name eks-nodegroup \
  --node-type t3.medium \
  --nodes-min 2 \
  --nodes-max 2 \
  --vpc-public-subnets subnet-0275a83eb493a97a8,subnet-0dd08dc8f4d5cd7b4 \
  --vpc-security-group-ids sg-0fa6391a37249a7e6 \
  --zones us-east-1a,us-east-1b,us-east-1c
```

Use `eksctl` to create your EKS cluster. Here's an example command for the Mumbai region (ap-south-1):

```
eksctl create cluster \
  --name eks-cluster \
  --region ap-south-1 \
  --nodegroup-name eks-nodegroup \
  --node-type t3.medium \
  --nodes-min 2 \
  --nodes-max 2 \
  --vpc-public-subnets subnet-0275a83eb493a97a8,subnet-0dd08dc8f4d5cd7b4 \
  --vpc-security-group-ids sg-0fa6391a37249a7e6 \
  --zones ap-south-1a,ap-south-1b,ap-south-1c
```

NOTE: Cluster creation takes 5-10 minutes. Be patient! Once complete, verify the cluster by running:

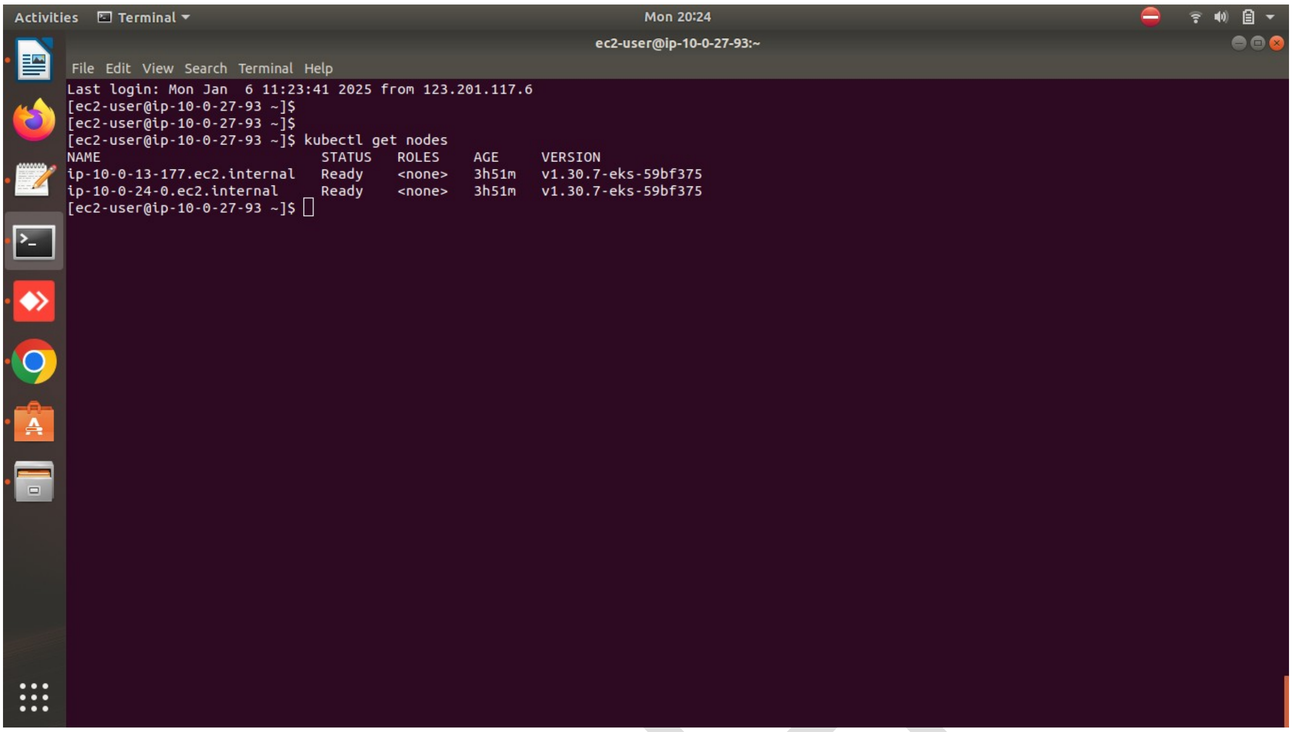
#kubectl get nodes



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:instanceState=running

Instances (3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
eks-cluster-eks-worker-node1	i-01300eef5300d9e53	Running	t3.medium	3/3 checks passec	View alarms +	us-east-1a	ec2-3-92-229-164.c
eks-cluster	i-0f6e53c2224018ead	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-54-86-164-123
eks-cluster-eks-worker-node2	i-0a6fc779019c9bce5	Running	t3.medium	3/3 checks passec	View alarms +	us-east-1b	ec2-54-174-136-17



```
Last login: Mon Jan  6 11:23:41 2025 from 123.201.117.6
[ec2-user@ip-10-0-27-93 ~]$
[ec2-user@ip-10-0-27-93 ~]$
[ec2-user@ip-10-0-27-93 ~]$ kubectl get nodes
NAME                           STATUS   ROLES    AGE     VERSION
ip-10-0-13-177.ec2.internal   Ready    <none>   3h51m   v1.30.7-eks-59bf375
ip-10-0-24-0.ec2.internal     Ready    <none>   3h51m   v1.30.7-eks-59bf375
[ec2-user@ip-10-0-27-93 ~]$ 
```

Congratulations! Your EKS cluster is now ready to use. Start deploying your Kubernetes applications and enjoy the power of scalable container orchestration!

Deploying a Login Page on EKS: Step-by-Step Walkthrough

In this guide, I demonstrate how I deployed a responsive HTML-based login page within an Amazon EKS cluster. The deployment is crafted for scalability and accessibility, using Kubernetes ConfigMap, Pod, and Service YAML configurations to ensure a seamless workflow.

This approach showcases best practices, leveraging Kubernetes capabilities to host and expose a static web application through a Load Balancer for external access.

Step 1: HTML Content Stored in ConfigMap

The login page's HTML content is stored in a Kubernetes ConfigMap for centralized management. This approach simplifies updates and decouples static content from the application logic.

Here’s the `nginx-login-configmap.yaml`:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: login-page-configmap
data:
  index.html: |
    <!DOCTYPE html>
    <html lang="en">
    <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Login Page</title>
      <style>
        /* CSS styles for responsive design and visual appeal */
        body {
          font-family: 'Arial', sans-serif;
          margin: 0;
          padding: 0;
          background: linear-gradient(135deg, #6a11cb 0%, #2575fc 100%);
          display: flex;
          justify-content: center;
          align-items: center;
          height: 100vh;
          color: #333;
        }
        .login-container {
          background: #fff;
          padding: 40px;
          border-radius: 10px;
          box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);
          width: 100%;
          max-width: 400px;
        }
        /* Additional styling omitted for brevity */
      </style>
    </head>
    <body>
      <div class="login-container">
        <h2>Login</h2>
        <form>
          <div class="form-group">
            <label for="username">Username</label>
            <input type="text" id="username" name="username" placeholder="Enter
your username" required>
          </div>
          <div class="form-group">
            <label for="password">Password</label>
            <input type="password" id="password" name="password"
placeholder="Enter your password" required>
          </div>
          <button type="submit" class="login-button">Login</button>
        </form>
        <div class="login-footer">
          <p>Don't have an account? <a href="#">Sign Up</a></p>
          <p><a href="#">Forgot Password?</a></p>
        </div>
      </div>
    </body>
    </html>
```

Command to Apply ConfigMap:

```
kubectl apply -f nginx-login-configmap.yaml
```

Step 2: Deploy the Pod

The application is hosted on an Nginx web server, with the HTML file served from the ConfigMap as a mounted volume. This ensures that any changes to the ConfigMap are reflected dynamically.

Here’s the `nginx-login.yaml`:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-login-pod
  labels:
    app: nginx-login
spec:
  containers:
  - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
    volumeMounts:
    - name: login-page-volume
      mountPath: /usr/share/nginx/html
  volumes:
  - name: login-page-volume
    configMap:
      name: login-page-configmap
```

Command to Deploy the Pod:

```
kubectl apply -f nginx-login.yaml
```

Step 3: Expose the Application via a Load Balancer

To make the application accessible outside the cluster, I used a Kubernetes Service of type `LoadBalancer`.

Here’s the `nginx-login-svc.yaml`:

```
yaml
Copy code
apiVersion: v1
kind: Service
metadata:
  name: nginx-login-service
spec:
  selector:
    app: nginx-login
  ports:
  - protocol: TCP
```

```
    port: 80
    targetPort: 80
  type: LoadBalancer
```

Command to Expose the Pod:

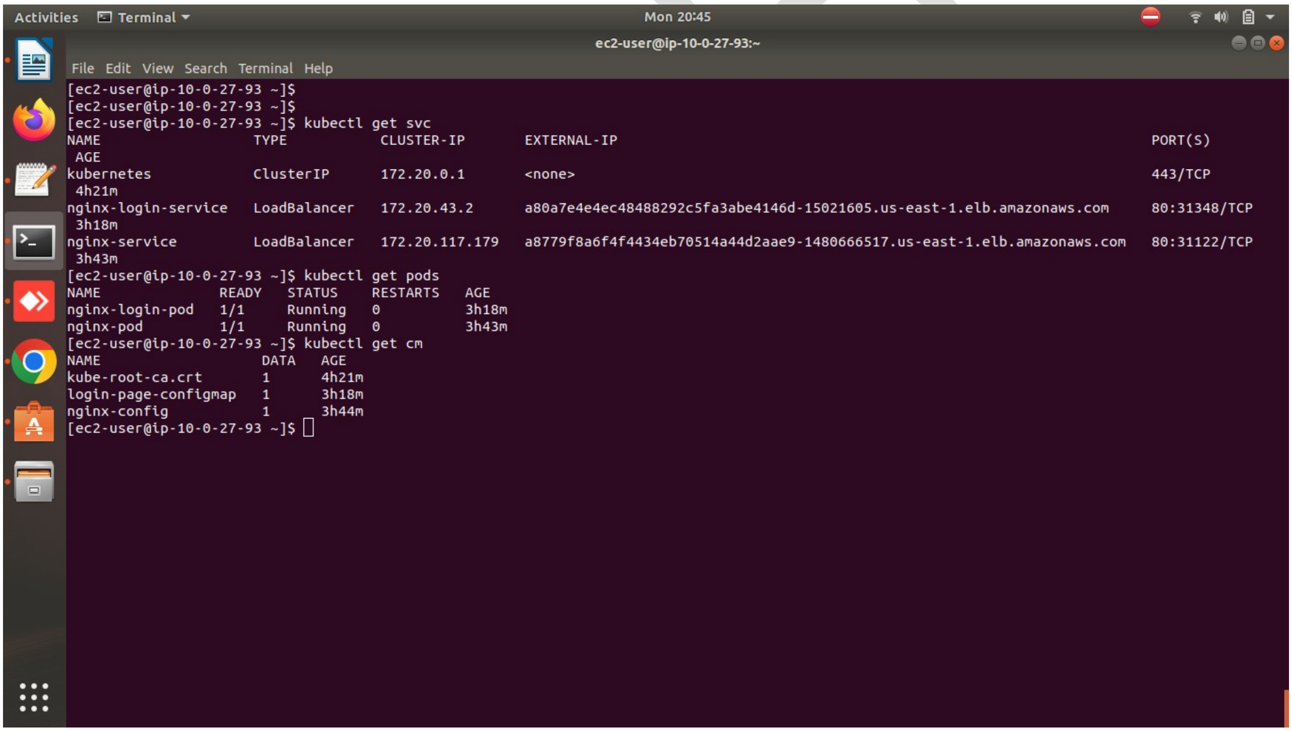
```
kubectl apply -f nginx-login-svc.yaml
```

Step 4: Access the Application

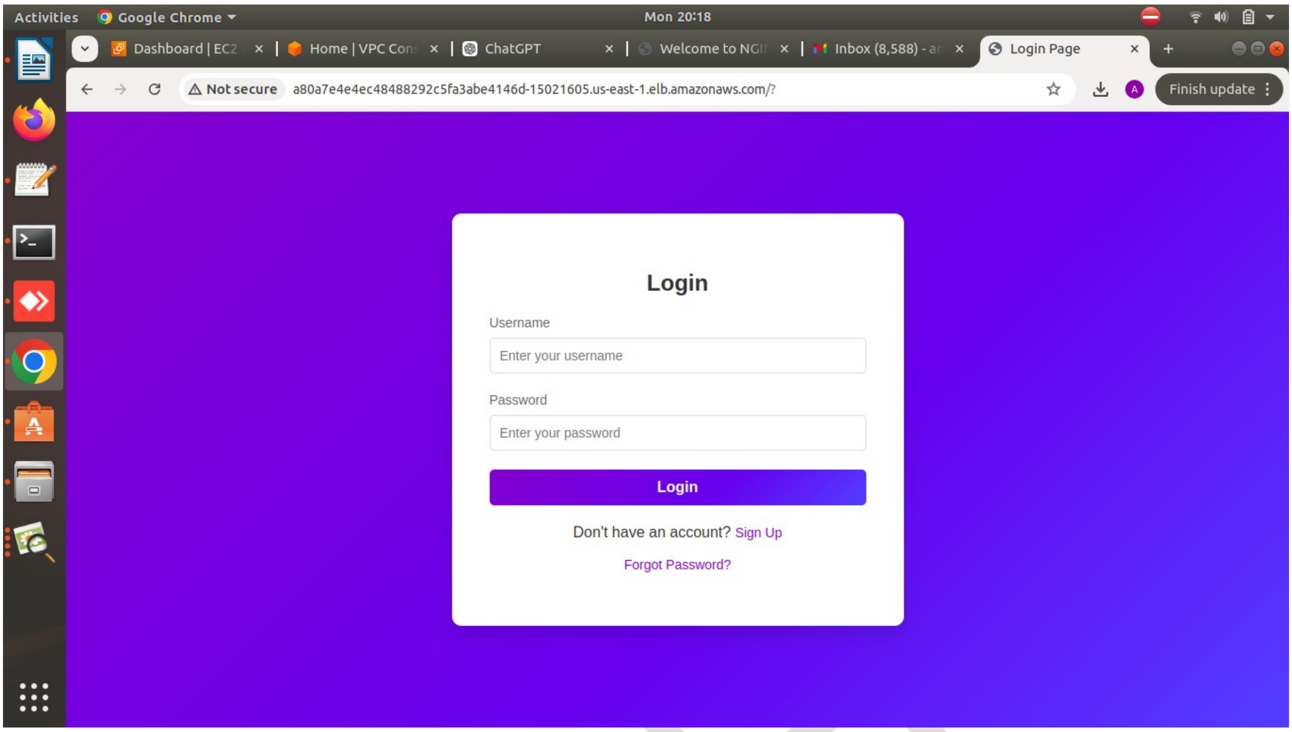
Once the Service is created, retrieve the external IP address of the Load Balancer:

```
kubectl get svc nginx-login-service
```

Open the external IP in your browser, and you'll see the professional and responsive login page live!



```
[ec2-user@ip-10-0-27-93 ~]$
[ec2-user@ip-10-0-27-93 ~]$
[ec2-user@ip-10-0-27-93 ~]$ kubectl get svc
NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)
 AGE
kubernetes            ClusterIP      172.20.0.1      <none>                                                                     443/TCP
 4h21m
nginx-login-service   LoadBalancer   172.20.43.2     a80a7e4e4ec48488292c5fa3abe4146d-15021605.us-east-1.elb.amazonaws.com     80:31348/TCP
 3h18m
nginx-service         LoadBalancer   172.20.117.179  a8779f8a6f4f4434eb70514a44d2aae9-1480666517.us-east-1.elb.amazonaws.com   80:31122/TCP
 3h43m
[ec2-user@ip-10-0-27-93 ~]$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
nginx-login-pod   1/1     Running   0          3h18m
nginx-pod         1/1     Running   0          3h43m
[ec2-user@ip-10-0-27-93 ~]$ kubectl get cm
NAME                   DATA   AGE
kube-root-ca.crt       1      4h21m
login-page-configmap   1      3h18m
nginx-config           1      3h44m
[ec2-user@ip-10-0-27-93 ~]$ 
```



Summary

- **ConfigMap**: Stores the HTML content for centralized, dynamic updates.
- **Pod**: Hosts the Nginx server, serving the login page via ConfigMap as a mounted volume.
- **Service**: Exposes the application using a Load Balancer for external access.

This setup demonstrates how Kubernetes can streamline static content hosting while maintaining flexibility and scalability.

Outcome: A fully functional and responsive login page deployed on Amazon EKS, accessible via a public endpoint.

✨ *This deployment is a great example of leveraging Kubernetes best practices to deliver scalable and professional-grade web applications.*