Steps To connect to a remote Amazon EKS (Elastic Kubernetes Service) cluster .

1. Install Required Tools

Ensure you have the following installed on your local machine:

- **AWS CLI** (latest version) → Install Guide
- **kubectl** (compatible with your EKS version) → Install Guide
- **aws-iam-authenticator** (optional, if your AWS CLI version is outdated) → Install Guide
- Helm Installed: Install Helm if not already installed

To create an **Amazon EKS** cluster, install `kubectl`, and configure it to communicate with the cluster, follow these **step-by-step** instructions.

Step 1: Install AWS CLI, eksctl & kubectl

Before creating an EKS cluster, you need to install the required tools.

1. Install AWS CLI

If you haven't installed the AWS CLI, install it using the following command:

Linux & macOS

```
curl "https://awscli.amazonaws.com/AWSCLIV2-$(uname -s | tr '[:upper:]'
'[:lower:]')-$(uname -m).zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Windows

Download and install from:
https://awscli.amazonaws.com/AWSCLIV2.msi

Verify Installation

```
aws --version
```

2. Configure AWS CLI with IAM User or Role

Run:

```
aws configure
```

Enter your AWS **Access Key ID**, **Secret Access Key**, **Region**, and **Output format**.

To verify:

```
aws sts get-caller-identity
```

3. Install eksctl

Linux & macOS

```
curl -sSL
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(una
me -s)_amd64.tar.gz" | tar xz -C /usr/local/bin
```

Windows

```
choco install eksctl
```

or download from https://github.com/weaveworks/eksctl/releases.

Verify Installation

```
eksctl version
```

4. Install kubectl

EKS requires a specific version of `kubectl`. To install the latest compatible version:

Linux

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

chmod +x kubectl

sudo mv kubectl /usr/local/bin/
```

macOS

```
brew install kubectl
```

Windows

```
choco install kubernetes-cli
```

Verify Installation

```
kubectl version --client
```

Step 2: Create an EKS Cluster

Now, create an **EKS cluster** using `eksctl`.

1. Create an EKS Cluster with eksctl

```
eksctl create cluster \
  --name my-eks-cluster \
  --region us-east-1 \
  --version 1.27 \
  --nodegroup-name my-node-group \
  --node-type t3.medium \
  --nodes 2 \
  --nodes-min 1 \
  --nodes-max 3 \
  --managed
```

- `--name my-eks-cluster` → Name of the cluster.
- `--region us-east-1` → AWS region (Change if needed).
- `--version 1.27` → Kubernetes version.
- `--nodegroup-name my-node-group` → Name of the worker node group.
- `--node-type t3.medium` → EC2 instance type for nodes.
- `--nodes 2` → Initial node count.
- `--nodes-min 1` and `--nodes-max 3` → Auto-scaling settings.
- `--managed` → Managed node group.

This process takes about **10–15 minutes**.

2. Verify the Cluster Creation

After the cluster is created, check its status:

```
eksctl get cluster --name my-eks-cluster --region us-east-1
```

Step 3: Configure kubectl to Connect to EKS

Once your EKS cluster is ready, configure `kubectl` to communicate with it.

1. Update the Kubeconfig

Run:

```
aws eks --region <region-name> update-kubeconfig --name <cluster-name>
ex: aws eks --region us-east-1 update-kubeconfig --name my-eks-cluster
```

This updates the Kubernetes configuration file so `kubectl` can connect to the EKS cluster.

2. Verify the Connection

```
kubectl get nodes
```

You should see your worker nodes in the **Ready** state.

Final Steps

Now, your **EKS cluster is running**, and `kubectl` is configured! 🎉

From here, you can:

- Deploy applications on **EKS**.
- Install **Prometheus and Grafana** (as per the previous guide).
- Implement **Ingress and LoadBalancer** to expose services.

To install helm

Helm Installed: Install Helm if not already installed

```
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

chmod 700 get_helm.sh
./get_helm.sh
```