**Kubeconfig**

* A file that is used to configure access to clusters is called a *kubeconfig file*.
* A kubeconfig file is a configuration file used by Kubernetes to let the kubectl command-line tool (and other clients) know how to connect to a Kubernetes cluster.
* It acts like a "credentials + connection settings" file for Kubernetes.

**📂 What it Contains**

The kubeconfig file usually stores:

1. **Clusters** → Information about Kubernetes clusters (like API server URL and certificates).
2. **Users (Credentials)** → Authentication details (like tokens, certificates, usernames/passwords).
3. **Contexts** → A mapping that ties a user to a cluster in a specific namespace.
   * Context = **Cluster + User + Namespace**
4. **Current-context** → The default context that kubectl uses if you don’t specify one.

**Connect Private end point EKS cluster**

When your **EKS (Elastic Kubernetes Service) cluster** is created with a **private endpoint**, its API server is **not accessible from the public internet**. That means you cannot just run kubectl from your laptop and expect it to work. You need to connect from a **private network** that can reach the EKS API server.

Here’s what you need to do step by step:

**🔑 Steps to Connect to a Private Endpoint EKS Cluster**

**1. Network Access**

* Make sure you are in the same **VPC** or a **connected network** (via VPN, Direct Connect, or VPC Peering).
* Options:
  + **From an EC2 instance** in the same VPC/subnet (bastion/jump host).
  + **Via AWS VPN Client** or Direct Connect from your local machine.
  + **AWS Workspace / CloudShell** (they can access private endpoints if configured).

**2. IAM Authentication**

EKS uses IAM for authentication. You need:

* awscli installed and configured with credentials (aws configure).
* Permissions: eks:DescribeCluster and eks:ListClusters.

**3. Update Kubeconfig**

Use AWS CLI to generate your kubeconfig entry:

aws eks update-kubeconfig \

--region <your-region> \

--name <cluster-name> \

--role-arn <optional-IAM-role>

This will add the cluster entry to your ~/.kube/config.

**4. Use kubectl**

Now you can run:

kubectl get nodes

✅ But this only works if your machine is inside the private network that can reach the cluster endpoint.

**5. If You’re Outside the VPC**

You need a way in:

* **SSH Bastion Host** → SSH into an EC2 in the VPC, then run kubectl there.
* **VPN** → Connect your laptop to the VPC’s private network.
* **kubectl proxy / port-forward via SSM** → Use AWS SSM Session Manager to tunnel commands through an EC2 instance.

**📝 Example Setup**

Let’s say:

* Your cluster is in us-east-1.
* You have no VPN, but you can create a small EC2 instance in the cluster’s VPC.  
  Steps:

1. Launch EC2 in same VPC as EKS.
2. Install awscli and kubectl on that EC2.
3. Run:

* aws eks update-kubeconfig --region us-east-1 --name my-cluster
* kubectl get pods -A

1. (Optional) Use SSH port forwarding to connect kubectl on your laptop via the EC2.

🔒 In short:  
For **private endpoint EKS**, you need **network access to the VPC** (VPN/EC2/DirectConnect) + **IAM auth + kubeconfig**.