**Steps to Connect to a Private Endpoint EKS Cluster:-**

**1) Quick overview — what “private endpoint” means:**

* When EKS is configured with a private endpoint, the Kubernetes API server endpoint resolves to private IP addresses inside the VPC (not public IPs).
* So your client must have network-level access into that VPC (or into a VPC that can route to it) and DNS must resolve the cluster endpoint name to those private IPs.

**2) High-level options to reach the private API**

Pick one depending on your environment / security rules:

* **Run kubectl from inside the same VPC** (EC2 instance, bastion/jump host, or a management pod).
* **VPN / Direct Connect / Client VPN** — connect your laptop/network into the VPC.
* **VPC peering / Transit Gateway** — from another VPC that can route to the cluster VPC (plus DNS).
* **AWS Systems Manager (SSM)** — run kubectl on an EC2 via SSM Session Manager (no inbound SSH required).
* **SSH tunnel / port-forward via a bastion host** — tunnel 6443 through a host inside the VPC.

**3) Prerequisites (things you must confirm first):-**

* **Network connectivity**: You have a route from your machine (or the host where you’ll run kubectl) to the cluster VPC and to the API server’s private IPs (port **443**).
* **DNS resolution**: The cluster DNS name must resolve to private IPs for your client (if not, you may need Route53 Resolver rules or manual hosts entries).
* **AWS credentials & permissions**: You have an IAM user/role allowed to call eks:DescribeCluster and to generate an auth token (AWS CLI v2 recommended).
* **Kubernetes RBAC**: The IAM identity you use must be mapped to Kubernetes users/groups (via the aws-auth ConfigMap or appropriate role bindings) so you have permissions inside the cluster.
* **Tools installed**: aws CLI v2, kubectl (compatible version), optionally eksctl.

**4) Typical concrete approaches — step-by-step:-**

**A) From an EC2 bastion / management host inside the same VPC:-**

(Recommended for operations when you can’t/shouldn’t open the endpoint to internet.)

1. Launch an EC2 in same VPC/subnet (or that has routing to the cluster subnets). Prefer SSM-managed instance (no public IP required).
2. On the host, install AWS CLI v2 and kubectl.
3. Configure AWS credentials (IAM role on the EC2 instance or aws configure for user keys).
4. Generate kubeconfig for the cluster:

aws eks update-kubeconfig --region <region> --name <cluster-name> \

--alias <context-name>

If you need to assume a role first:

aws eks update-kubeconfig --region <region> --name <cluster-name> \

--role-arn arn:aws:iam::<account-id>:role/<RoleToAssume>

1. Test:

kubectl get ns

kubectl get nodes

**B) From your laptop via AWS Client VPN or corporate VPN:-**

* Create/configure an AWS Client VPN endpoint attached to the VPC/subnets that contain route to the EKS control plane.
* Connect your laptop to the Client VPN.
* Confirm you can reach the endpoint (e.g., curl -k https://<cluster-endpoint>/version or nc -vz <endpoint> 443).
* Run aws eks update-kubeconfig locally, then kubectl get pods -A.

**C) From another VPC (cross-account) via VPC peering / Transit Gateway:-**

* Ensure routing between VPCs and that security groups allow control-plane traffic.
* DNS: either use Route53 Resolver rules or manually configure DNS so the cluster endpoint name resolves to private IPs reachable in the peered VPC.
* Use aws eks update-kubeconfig from a host in the peered VPC.

**D) Use AWS Systems Manager (SSM) (no public IP / no open SSH):-**

* Run a maintenance EC2 in the cluster VPC with SSM agent enabled (or attach SSM managed instance role).
* Start an SSM session (aws ssm start-session) to the instance and run aws eks update-kubeconfig there. Run kubectl commands during the session.
* Alternatively use SSM port forwarding if you want to forward 6443 back to your laptop.

**5) Important commands & checks:-**

* Check cluster endpoint & access mode:

aws eks describe-cluster --name <cluster-name> --region <region> \

--query "cluster.{endpoint:endpoint,private:resourcesVpcConfig.endpointPrivateAccess,public:resourcesVpcConfig.endpointPublicAccess}" --output table

* Generate/update kubeconfig:

aws eks update-kubeconfig --region <region> --name <cluster-name>

* Inspect kubeconfig:

kubectl config view --minify

* Test network reachability (from host inside VPC):

curl -vk https://<cluster-endpoint>/version

# or

nc -vz <cluster-endpoint> 443

**6) kubeconfig + IAM authentication notes:-**

* aws eks update-kubeconfig creates a kubeconfig that uses an **exec** plugin calling aws eks get-token — your AWS credentials are used to create a short-lived token for the Kubernetes API. No static kube token is needed.
* Make sure the IAM principal (user/role) that obtains tokens is **allowed by Kubernetes RBAC**. EKS maps IAM → Kubernetes via the aws-auth ConfigMap (mapRoles / mapUsers) and then you apply RBAC roles/rolebindings in the cluster.
* To view/add mappings (must be done by an account with admin access):

kubectl -n kube-system edit configmap/aws-auth

* Example mapping entry to give an IAM role admin privileges:

mapRoles: |

- rolearn: arn:aws:iam::<ACCOUNT\_ID>:role/<AdminRole>

username: admin

groups:

- system:masters

**7) DNS gotchas:-**

* The cluster endpoint hostname resolves to **private IPs** only **from VPCs** that have DNS resolution to them. If you connect via peering/VPN, you may need to configure **Route53 Resolver rules** (inbound/outbound or conditional forwarding) so your client can resolve the cluster endpoint to the private addresses.
* If DNS is not working, you can test with dig or nslookup from a host inside the target network.

**8) Security and best-practices:-**

* Prefer SSM or Client VPN over opening SSH ports publicly. SSM avoids incoming SSH and is auditable via CloudTrail.
* Use least-privileged IAM roles and map to fine-grained Kubernetes RBAC (avoid giving everyone system:masters).
* If you must allow limited public access, prefer restricting publicAccessCidrs to known admin IPs (but private-only is more secure).
* Audit aws-auth and RBAC mappings carefully.

**9) Common errors & troubleshooting:-**

* i/o timeout / dial tcp: connect: connection timed out → network route or security group blocking port 443.
* could not resolve host → DNS resolution issue; use Route53 resolver rules or host entries.
* x509: certificate signed by unknown authority → CA mismatch; check kubeconfig .
* Forbidden / Unauthorized → IAM principal not mapped to Kubernetes RBAC; check aws-auth and role bindings.
* kubectl hangs → confirm you can curl the endpoint and that the exec plugin aws eks get-token is returning a token.

**10) Quick checklist before trying kubectl from your laptop:-**

* You have network connectivity to cluster VPC (VPN/peering/SSM/bastion).
* DNS resolves cluster endpoint to private IPs.
* aws CLI v2 installed and credentials set.
* You have permissions (IAM + Kubernetes RBAC) to access the cluster.
* Run aws eks update-kubeconfig --region <r> --name <cluster> and then kubectl get ns.