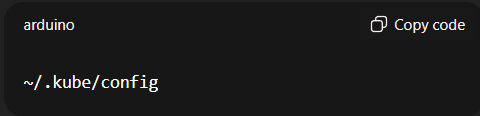
1. What is meant by kubeconfig in Kubernetes?

**Definition:**  
kubeconfig is a configuration file used by Kubernetes command-line tools (like kubectl, helm, k9s) to authenticate and communicate with a Kubernetes cluster.

**Location:**

By default, it is stored at:



**What it contains?**

A kubeconfig file typically has 3 key sections:

* **Clusters** → Information about API server endpoints and certificates of different clusters.
* **Users (credentials)** → Authentication details (token, IAM roles, certificates).
* **Contexts** → A combination of cluster + user + namespace (so you can quickly switch between clusters and namespaces).

**What is the need of kubeconfig file?**

* Kubeconfig allows switching between clusters and namespaces easily by changing the "context"—grouping cluster user and namespace.
* It is essential not only for end-users but also for cluster components (controllers, kubelets) to interact securely with the Kubernetes API server.
* Administrators or cloud providers give kubeconfig to users that need cluster access. It is crucial to protect kubeconfig files since they contain sensitive tokens and certificates.

**Example of kubeconfig:**

**apiVersion: v1**

**kind: Config**

**clusters:**

**- name: my-cluster**

**cluster:**

**server: https://1234ABCD.gr7.us-east-1.eks.amazonaws.com**

**certificate-authority-data: LS0tLS1CRUdJTi...**

**users:**

**- name: my-user**

**user:**

**exec:**

**apiVersion: client.authentication.k8s.io/v1beta1**

**command: aws**

**args:**

**- "eks"**

**- "get-token"**

**- "--cluster-name"**

**- "my-cluster"**

**contexts:**

**- name: my-context**

**context:**

**cluster: my-cluster**

**user: my-user**

**namespace: default**

**current-context: my-context**

1. **Connecting to a Private Endpoint EKS Cluster**

When you create an Amazon EKS cluster, you can configure the API server endpoint as:

* Public → Accessible over the internet.
* Private → Only accessible from within the VPC.
* Both → Accessible from the internet (restricted with CIDRs) + inside VPC.

If your EKS cluster has a private-only endpoint, then:

Steps to Connect:

Step 1: Ensure Network Access

* + Since the cluster endpoint is inside the VPC, your machine needs network access into that VPC.
  + You have 3 main options:

1. From an EC2 instance inside the VPC
   * SSH into a bastion/jump host or directly into an EC2 instance in the same VPC/subnet as your EKS cluster.
   * Use kubectl from that EC2 machine.
2. From your local machine via VPN/Direct Connect
   * If your company has VPN access or AWS Direct Connect to the VPC, you can run kubectl locally, since your laptop will be on the same network.
3. AWS PrivateLink + VPC Endpoint
   * Create a VPC endpoint for the EKS API server in your VPC.
   * Route your requests through that endpoint.

Step 2: Update kubeconfig

* Run the AWS CLI command to generate kubeconfig for your cluster:

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

Step 3: IAM Authentication

* EKS uses IAM authentication, so you must:
  + Have IAM permissions (eks:DescribeCluster, eks:GetToken).
  + Be mapped in the aws-auth ConfigMap of the cluster for Kubernetes RBAC.

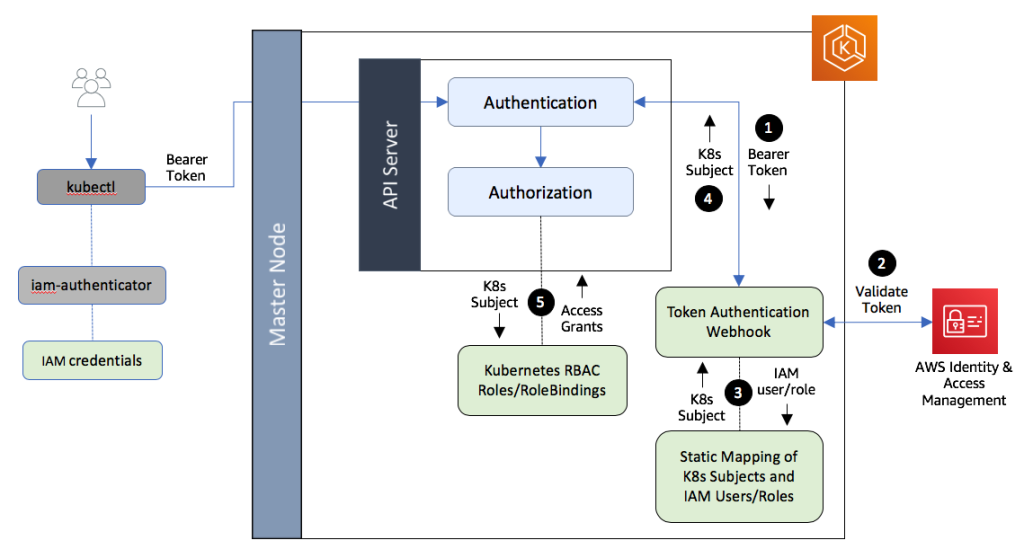
Step 4: Test Connection

Once network + kubeconfig + IAM are set:

**kubectl get nodes**

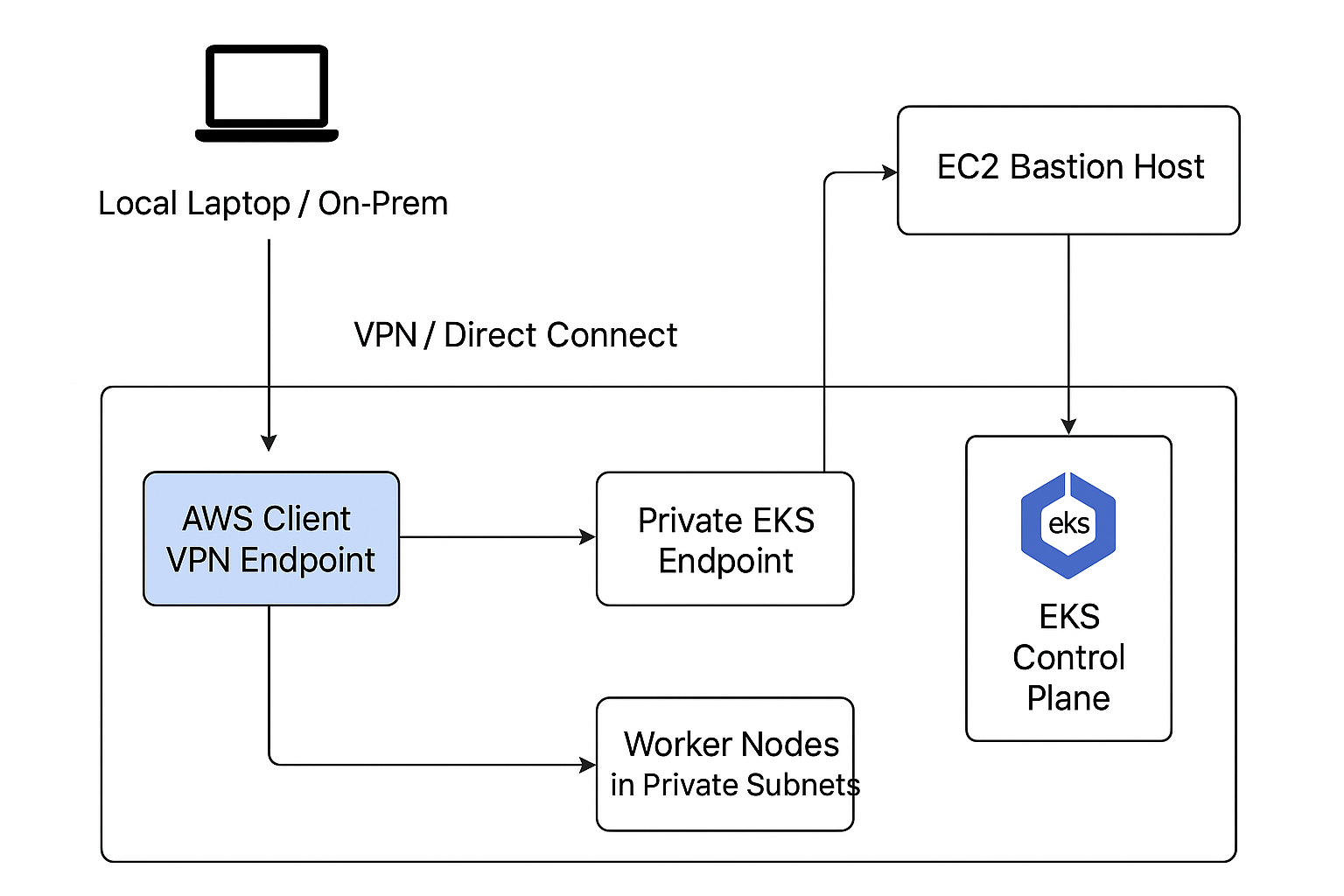
Workflow & Architecture

**(Cluster Networking & Components)**



* This diagram shows the infrastructure and network-level view of an EKS cluster.
* It explains that:
  + The control plane (API server, etcd, scheduler, controller manager) runs inside the AWS-managed VPC.
  + The worker nodes (EC2 or Fargate) run inside the customer’s VPC.
  + Communication between control plane and worker nodes happens through ENIs (Elastic Network Interfaces).
  + A kubectl user connects to the EKS API endpoint (public or private).

Amazon EKS Private Cluster Access Architecture Diagram



* This diagram shows the Amazon EKS Private Cluster Access Architecture Diagram.
* Here’s what each part means:
* **Local Laptop / On-Prem** → Your machine where you run kubectl.
* **VPN / Direct Connect** → The secure network tunnel from your machine into the AWS VPC.
* **AWS Client VPN Endpoint** → An AWS-managed VPN service that lets your laptop join the VPC privately.
* **EC2 Bastion Host** → A jump server inside the VPC that you can SSH/SSM into and then run kubectl.
* **Private EKS Endpoint** → The Kubernetes API server endpoint, only reachable from inside the VPC.
* **EKS Control Plane** → The managed Kubernetes master components (API server, etcd, controller manager).
* **Worker Nodes in Private Subnets** → EC2 nodes (or Fargate pods) where workloads actually run, inside private subnets.

In short: This shows the **network flow and access methods (VPN or Bastion)** needed to reach a **private EKS cluster endpoint** securely.