Sure! Here's a detailed explanation of the entire Terraform setup you requested to:

* Build a **VPC** with one **public subnet**, one **private subnet**
* Add an **Internet Gateway** for public access
* Add a **NAT Gateway** so private subnet instances can access the internet securely
* Deploy a **private EKS cluster** (no public endpoint)
* Include **worker nodes** (managed node group) in the private subnet
* Allow listing of nodes and pods from within the cluster

**🔧 1. Provider and Terraform Configuration (provider.tf)**

hcl

CopyEdit

provider "aws" {

region = "us-east-1"

}

This sets the AWS region and tells Terraform to use the AWS provider.

hcl

CopyEdit

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.0"

}

kubernetes = {

source = "hashicorp/kubernetes"

version = "~> 2.0"

}

}

required\_version = ">= 1.3.0"

}

This block ensures:

* You’re using compatible AWS and Kubernetes providers.
* Terraform version is at least 1.3.

**🎚️ 2. Variables (variables.tf)**

We declare values like:

* CIDRs for VPC and subnets
* AWS region
* EKS cluster name

This allows us to reuse or change them easily later.

**🌐 3. VPC, Subnets, Gateways (main.tf)**

**VPC**

hcl

CopyEdit

resource "aws\_vpc" "main" {

cidr\_block = var.vpc\_cidr

enable\_dns\_hostnames = true

enable\_dns\_support = true

...

}

* Creates a VPC with CIDR 10.0.0.0/16.
* DNS options are enabled (necessary for EKS).

**Public & Private Subnets**

hcl

CopyEdit

resource "aws\_subnet" "public" { ... }

resource "aws\_subnet" "private" { ... }

* map\_public\_ip\_on\_launch = true is set only for the public subnet.
* Each subnet is placed in a specific availability zone (us-east-1a).

**Internet Gateway (IGW) + Route Table**

hcl

CopyEdit

resource "aws\_internet\_gateway" "igw" { ... }

* Public subnet needs an IGW to allow direct internet access.
* Public route table sends traffic 0.0.0.0/0 to the IGW.

**NAT Gateway + Route Table for Private Subnet**

hcl

CopyEdit

resource "aws\_nat\_gateway" "nat" { ... }

resource "aws\_eip" "nat" { ... }

* A NAT Gateway allows **private subnet** instances (e.g., EKS nodes) to access the internet **outbound** (e.g., pull Docker images), without being reachable from outside.
* EIP is required to assign a static IP to the NAT Gateway.

**☸️ 4. EKS Cluster (Private Access Only)**

hcl

CopyEdit

module "eks" {

source = "terraform-aws-modules/eks/aws"

version = "20.8.4"

cluster\_name = var.cluster\_name

...

cluster\_endpoint\_private\_access = true

cluster\_endpoint\_public\_access = false

...

}

* Uses the [official AWS EKS Terraform module](https://github.com/terraform-aws-modules/terraform-aws-eks), which abstracts a lot of complexity.
* EKS Control Plane endpoint is set to **private only**, i.e., **no internet exposure**.
* subnet\_ids is set to the **private subnet** for both cluster control plane and worker nodes.

**Managed Node Group**

hcl

CopyEdit

eks\_managed\_node\_groups = {

node\_group = {

desired\_capacity = 2

instance\_types = ["t3.medium"]

...

}

}

* Uses **EKS-managed node groups** (best practice).
* Nodes will automatically scale between 1 and 3 as needed.
* All worker nodes will live in the **private subnet**.

**📤 5. Outputs (outputs.tf)**

You get:

* Cluster name
* Endpoint
* Kubeconfig filename (useful if needed programmatically)

**✅ Post Apply: Accessing the Private EKS Cluster**

Since the EKS endpoint is private, it’s **not reachable from your local machine directly**. You must connect from:

1. An EC2 instance in the **private subnet** (bastion or jumpbox).
2. Or set up a **VPN or AWS SSM** to securely access the private VPC.

Once inside a node in that VPC, you can run:

bash

CopyEdit

aws eks update-kubeconfig --name my-private-eks-cluster --region us-east-1

kubectl get nodes

kubectl get pods -A

**🛡️ Industry Best Practices Followed**

| **Feature** | **Practice** |
| --- | --- |
| **Network Segregation** | Public & Private Subnets |
| **Security** | Private EKS Endpoint |
| **Scalability** | Managed Node Groups |
| **Cost Control** | Single NAT Gateway |
| **Modularity** | Using Terraform Modules |
| **Tagging** | Resources are tagged for visibility |
| **Version Pinning** | Ensures predictable builds |

Would you like me to help you add:

* A **bastion host** to access the private cluster?
* **Remote backend** (like S3 + DynamoDB) for Terraform state?
* Or setup **IAM roles** and aws-auth config for RBAC?

**TERRAFORM PLAN OUTPUT**

Thanks for sharing your Terraform plan output file tfout.tf. I’ve reviewed it and here’s a **clear, plain-English explanation** of what this Terraform plan is doing and what resources it will create.

**🔍 Summary of the Plan**

Your Terraform plan is preparing to create a **secure and scalable AWS infrastructure** that includes:

1. **A new VPC**
2. **Public and Private subnets**
3. **Routing tables and internet/NAT gateways**
4. **EKS cluster (Elastic Kubernetes Service)** with managed node group
5. **Security groups**
6. **IAM roles and policies**

**📘 Breakdown of Resources Being Created**

**1. VPC and Networking**

* A **VPC** with a specific CIDR block (e.g., 10.0.0.0/16)
* **2 Subnets**:
  + **1 Public Subnet** (for NAT Gateway, bastion hosts, etc.)
  + **1 Private Subnet** (for EKS worker nodes and internal workloads)
* **Internet Gateway (IGW)** for allowing internet access to public subnets.
* **NAT Gateway** to allow private subnets to access the internet securely without being exposed.
* **Elastic IP (EIP)** attached to the NAT Gateway.
* **Route Tables** for each subnet, with correct routes to the IGW or NAT Gateway.

**2. EKS Cluster**

* A **private EKS cluster**, meaning:
  + The Kubernetes control plane is accessible only within the VPC.
* **IAM roles and policies** for the EKS control plane and worker nodes.
* **Managed Node Group** with EC2 worker nodes that will join the cluster.

**3. Security Groups**

* Security group rules that:
  + Allow worker nodes to talk to the control plane
  + Allow nodes to talk to each other
  + Allow outbound internet (from nodes)

**4. Outputs**

* The plan also prepares to output key data such as:
  + Cluster name
  + Endpoint
  + Certificate authority data
  + Subnet IDs

These will be useful for:

* Generating kubeconfig manually
* Connecting to the cluster using kubectl

**🧠 In Plain English**

This Terraform plan is going to set up a secure AWS environment with both public and private networking. It creates a private EKS Kubernetes cluster using AWS-managed nodes inside private subnets. Internet access for these nodes is handled securely using a NAT Gateway. All the network and security resources like subnets, gateways, and IAM roles are created automatically. Once complete, you can manage your Kubernetes cluster and workloads from your local machine (or through bastion/VPN if needed).

If you'd like, I can extract and explain each individual resource from the file or help you clean it up for best practices — would you like that?