```
provider "aws" {

  region = "us-east-1"

}


terraform {

  required_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 3.0"

    }

    kubernetes = {

      source  = "hashicorp/kubernetes"

      version = "~> 2.0"

    }

  }

}


resource "aws_vpc" "eks_vpc" {

  cidr_block = "10.0.0.0/16"

  enable_dns_support = true

  enable_dns_hostnames = true

}


resource "aws_subnet" "eks_subnet1" {

  vpc_id            = aws_vpc.eks_vpc.id

  cidr_block        = "10.0.1.0/24"
  availability_zone = "us-east-1a"

  map_public_ip_on_launch = true

}



resource "aws_subnet" "eks_subnet2" {

  vpc_id            = aws_vpc.eks_vpc.id

  cidr_block        = "10.0.2.0/24"

  availability_zone = "us-east-1b"

  map_public_ip_on_launch = true

}



resource "aws_eks_cluster" "eks_cluster" {

  name     = "my-eks-cluster"

  role_arn = aws_iam_role.eks_cluster_role.arn



  vpc_config {

    subnet_ids = [aws_subnet.eks_subnet1.id, aws_subnet.eks_subnet2.id]

  }



  depends_on = [

    aws_iam_role_policy_attachment.eks_AmazonEKSClusterPolicy,

    aws_iam_role_policy_attachment.eks_AmazonEKSServicePolicy,

  ]

}



resource "aws_iam_role" "eks_cluster_role" {
  name = "eks-cluster-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Principal = {
          Service = "eks.amazonaws.com"
        }
      },
    ]
  })
}

resource "aws_iam_role_policy_attachment" "eks_AmazonEKSClusterPolicy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
  role       = aws_iam_role.eks_cluster_role.name
}

resource "aws_iam_role_policy_attachment" "eks_AmazonEKSServicePolicy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSServicePolicy"
  role       = aws_iam_role.eks_cluster_role.name
}
```

```
# Node Group
resource "aws_eks_node_group" "eks_node_group" {
  cluster_name    = aws_eks_cluster.eks_cluster.name
  node_group_name = "eks-node-group"
  node_role_arn   = aws_iam_role.eks_node_role.arn
  subnet_ids      = [aws_subnet.eks_subnet1.id, aws_subnet.eks_subnet2.id]

  scaling_config {
    desired_size = 1
    max_size     = 2
    min_size     = 1
  }
}

resource "aws_iam_role" "eks_node_role" {
  name = "eks-node-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Principal = {
          Service = "ec2.amazonaws.com"
        }
      },
    ]
  })
}

resource "aws_iam_role_policy_attachment" "eks_worker_node_policy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
  role       = aws_iam_role.eks_node_role.name
}

resource "aws_iam_role_policy_attachment" "eks_cni_policy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"
  role       = aws_iam_role.eks_node_role.name
}

resource "aws_iam_role_policy_attachment" "eks_ecr_policy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
  role       = aws_iam_role.eks_node_role.name
}

output "cluster_endpoint" {
  value = aws_eks_cluster.eks_cluster.endpoint
}

output "cluster_ca_certificate" {
  value = aws_eks_cluster.eks_cluster.certificate_authority[0].data
```

```
}
```