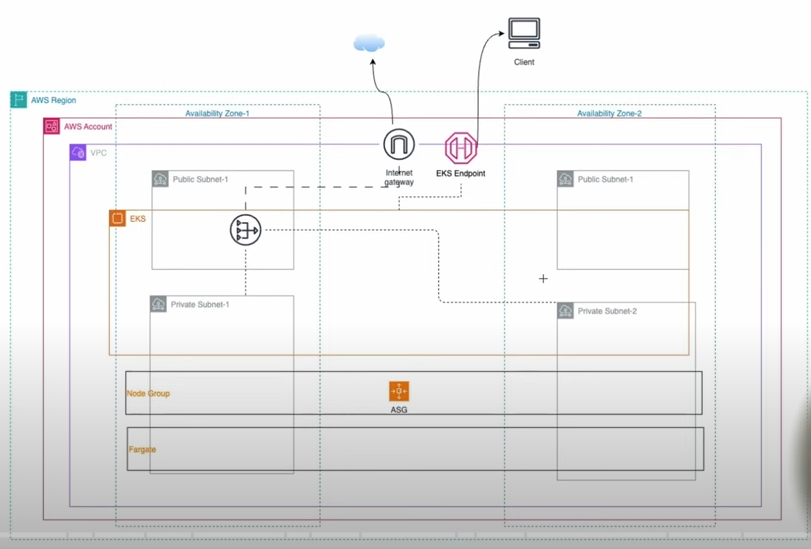
Create an EKS cluster With Terraform | Modules vs. No Modules



Tot codul este aici https://github.com/gurlal-1/devops-avenue/tree/main/yt-videos/eks-terraform

Node Group si Fargate profile trebuie definite in private subnets

1.without\_modules directory:

0-provider.tf:

provider "aws" {

  region = var.region

}

1-variables.tf:

variable "region" {

    type = string

    default = "us-east-1"

    description = "AWS region"

}

variable "cidr\_block" {

    type = string

    default = "10.10.0.0/16"

    description = "cidr block for VPC"

}

variable "tags" {

    type = map(string)

    default = {

        terraform = true

        kubernetes = "demo\_eks\_cluster"

        }

    description = "tag to apply to all resources"

    }

Terraform init -> download the plugins and the provider aws si verifica daca credentialele mele sunt configurate pt acest terminal

Variablila tags – se va aplica tag la toate resursele din vpc

2-vpc.tf

In primul rand vom define data source pt availability zones (sunt resurse care sunt deja create in AWS nu au fost create prin code terraform)

Enable\_dns\_hostnames = true // ne ajuta la asocierea dintre worker nodes si eks cluster

In continuare vom crea cele 4 subnets:

Cidrsubnet - functia permite să se împartă un interval IP mai mare(ex: 10.10.0.0/16) în subintervale (subneturi) mai mici(ex:10.10.10.0/24), prin extinderea prefixului.

In continuare vom crea IGW si NGW. Pt NGW vom avea nevoie de un elastic IP

Elastic IP si NGW trebuie sa fie create dup ace e creat IGW (depends\_on)

In continuare vom create route table (public si private). Pt public rt vom crea ruta catre internet(0.0.0.0/0) prin IGW si pt private rt vom crea ruta catre internet prin nat gw

In continuare vom crea adsocierile intre public subnets si public rt si intre private subnets si private rt

2-vpc.tf:

data "aws\_availability\_zones" "available" {

  state = "available"

}

resource "aws\_vpc" "demo\_eks\_cluster\_vpc" {

  cidr\_block           = var.cidr\_block

  enable\_dns\_hostnames = true

  tags                 = var.tags

}

resource "aws\_subnet" "public-subnet-1" {

  vpc\_id            = aws\_vpc.demo\_eks\_cluster\_vpc.id

  cidr\_block        = cidrsubnet(var.cidr\_block, 8, 10) # 10.10.10.0/24

  availability\_zone = data.aws\_availability\_zones.available.names[0]

  tags              = var.tags

}

resource "aws\_subnet" "public-subnet-2" {

  vpc\_id            = aws\_vpc.demo\_eks\_cluster\_vpc.id

  cidr\_block        = cidrsubnet(var.cidr\_block, 8, 20) # 10.10.20.0/24

  availability\_zone = data.aws\_availability\_zones.available.names[1]

  tags              = var.tags

}

resource "aws\_subnet" "private-subnet-1" {

  vpc\_id            = aws\_vpc.demo\_eks\_cluster\_vpc.id

  cidr\_block        = cidrsubnet(var.cidr\_block, 8, 110) #10.10.110.0/24

  availability\_zone = data.aws\_availability\_zones.available.names[0]

  tags              = var.tags

}

resource "aws\_subnet" "private-subnet-2" {

  vpc\_id            = aws\_vpc.demo\_eks\_cluster\_vpc.id

  cidr\_block        = cidrsubnet(var.cidr\_block, 8, 120) #10.10.120.0/24

  availability\_zone = data.aws\_availability\_zones.available.names[1]

  tags              = var.tags

}

resource "aws\_internet\_gateway" "eks-igw" {

  vpc\_id = aws\_vpc.demo\_eks\_cluster\_vpc.id

  tags   = var.tags

}

resource "aws\_eip" "eks-ngw-eip" {

  domain     = "vpc"

  depends\_on = [aws\_internet\_gateway.eks-igw]

  tags       = var.tags

}

resource "aws\_nat\_gateway" "eks-ngw" {

  allocation\_id = aws\_eip.eks-ngw-eip.id

  subnet\_id     = aws\_subnet.public-subnet-1.id

  # To ensure proper ordering, it is recommended to add an explicit dependency

  # on the Internet Gateway for the VPC.

  depends\_on = [aws\_internet\_gateway.eks-igw]

  tags       = var.tags

}

resource "aws\_route\_table" "public-rt" {

  vpc\_id = aws\_vpc.demo\_eks\_cluster\_vpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.eks-igw.id

  }

  tags = var.tags

}

resource "aws\_route\_table" "private-rt" {

  vpc\_id = aws\_vpc.demo\_eks\_cluster\_vpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_nat\_gateway.eks-ngw.id

  }

  tags = var.tags

}

resource "aws\_route\_table\_association" "public-rt-association-1" {

  subnet\_id      = aws\_subnet.public-subnet-1.id

  route\_table\_id = aws\_route\_table.public-rt.id

}

resource "aws\_route\_table\_association" "public-rt-association-2" {

  subnet\_id      = aws\_subnet.public-subnet-2.id

  route\_table\_id = aws\_route\_table.public-rt.id

}

resource "aws\_route\_table\_association" "private-rt-association-1" {

  subnet\_id      = aws\_subnet.private-subnet-1.id

  route\_table\_id = aws\_route\_table.private-rt.id

}

resource "aws\_route\_table\_association" "private-rt-association-2" {

  subnet\_id      = aws\_subnet.private-subnet-2.id

  route\_table\_id = aws\_route\_table.private-rt.id

}

Terraform plan si apply

Cream un fisier 3-eks.tf

Primul lucru trebuie sa cream un Role! Cu ajutorul acestui Role , EKS poate face modificari in numele meu in eks cluster

Apoi trebuie sa cream policy attachment la acest Role: AmazonEKSClusterPolicy – ii da drepturi lui Role sa faca modificari de resurse in numele meu.

Apoi cream variable pt eks cluster name si eks version

In 1-variables.tf adaug:

variable "eks\_version" {

    type = string

    default = "1.31"

    description = "eks\_version"

}

Si

variable "cluster\_name" {

    type = string

    default = "demo-eks-cluster"

    description = "eks cluster name"

}

In continuare vom crea EKS resource:

endpoint\_private\_access = true =. Worker nodes se vor putea asocial la eks cluster

[bootstrap\_cluster\_creator\_admin\_permissions](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/eks_cluster" \l "bootstrap_cluster_creator_admin_permissions-1) – vom putea accesa clusterul de indata ce resursele sunt create. Daca pun pe false va trebuie sa access config cu cine vrem noi sa aiba acces la cluster.

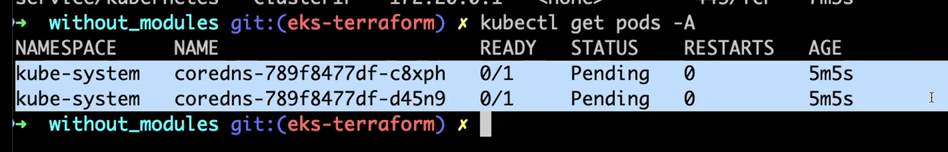
bootstrap\_self\_managed\_addons = true => sunt 3 addon-uri by default care se creaza pt ca Pods sa poate comunica intre ele

terraform plan si apply

Apoi trebuie sa configurez terminalul meu sa poata accesa acest eks cluster:

Aws eks update-kubeconfig –region us-east-1 –-name demo\_eks\_cluster

Kubectl get pods -A -> vad toate Pods



Pods sunt in Pending pt ca nu au worker nodes unde sa fie mapate

Cream profilul de fargate => 4-fargate.tf:

Avem nevoie de un Role la care sa adaugam un policy de Pod execution

resource "aws\_iam\_role" "demo-eks-fargate-profile-role" {

  name = "demo-eks-fargate-profile-role"

  assume\_role\_policy = jsonencode({

    Statement = [{

      Action = "sts:AssumeRole"

      Effect = "Allow"

      Principal = {

        Service = "eks-fargate-pods.amazonaws.com"

      }

    }]

    Version = "2012-10-17"

  })

}

resource "aws\_iam\_role\_policy\_attachment" "example-AmazonEKSFargatePodExecutionRolePolicy" {

  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy"

  role       = aws\_iam\_role.demo-eks-fargate-profile-role.name

}

Apoi trebuie sa cream Fargate profile:

La resource "aws\_eks\_fargate\_profile" folosim selector ca sa specificam in ce namespace sa programam maparea Pods

La subnets\_id -> punem ambele private subnets unde vrem sa avem nodurile pe care vor rule podurile

#these subnets must be labeled with kubernetes.io/cluster/{cluster-name} = owned –> a.i sanu avem probleme cand se creeaza nodurile unde vor rula Pods

subnet\_ids = [

aws\_subnet.private-subnet-1.id,

aws\_subnet.private-subnet-2.id

]

depends\_on = [ aws\_iam\_role\_policy\_attachment.fargate-execution-policy ]

}

* In “2-vpc.tf” vom crea o variabila locala:
* locals {
* additional\_tags = {
* "kubernetes.io/cluster/${var.cluster-name}" = "owned"
* }
* }

Vom updata si tags de la private subnets:

resource "aws\_subnet" "private-subnet-1" {

  vpc\_id            = aws\_vpc.demo\_eks\_cluster\_vpc.id

  cidr\_block        = cidrsubnet(var.cidr\_block, 8, 110) #10.10.110.0/24

  availability\_zone = data.aws\_availability\_zones.available.names[0]

  tags              = merge(var.tags, local.additional\_tags)

}

resource "aws\_subnet" "private-subnet-2" {

  vpc\_id            = aws\_vpc.demo\_eks\_cluster\_vpc.id

  cidr\_block        = cidrsubnet(var.cidr\_block, 8, 120) #10.10.120.0/24

  availability\_zone = data.aws\_availability\_zones.available.names[1]

  tags              = merge(var.tags, local.additional\_tags)

}

Terraform plan + apply

Daca as crea acum un Pod => fargate ar crea un node unde sa fie mapat Podul meu

Kubectl run nginx –image=nginx

Mai departe vrem sa cream si un node group => 5-nodegroup.tf

Pt node group trebuie sa cream un Role la care sa atasam 3 Policy:

resource "aws\_iam\_role" "demo-eks-ng-role" {

  name = "demo-eks-node-group-role"

  assume\_role\_policy = jsonencode({

    Statement = [{

      Action = "sts:AssumeRole"

      Effect = "Allow"

      Principal = {

        Service = "ec2.amazonaws.com"

      }

    }]

    Version = "2012-10-17"

  })

}

resource "aws\_iam\_role\_policy\_attachment" "eks-demo-ng-WorkerNodePolicy" {

  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"

  role       = aws\_iam\_role.demo-eks-ng-role.name

}

resource "aws\_iam\_role\_policy\_attachment" "eks-demo-ng-AmazonEKS\_CNI\_Policy" {

  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy"

  role       = aws\_iam\_role.demo-eks-ng-role.name

}

resource "aws\_iam\_role\_policy\_attachment" "eks-demo-ng-ContainerRegistryReadOnly" {

  policy\_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"

  role       = aws\_iam\_role.demo-eks-ng-role.name

}

Primul policy e pt ca worker nodes sa poata accesa images/containerele

Al doilea policy e pt a active networking intre Pods

Al treilea policy e pr ca node group sa poata downloada imagini din Container Registry(ECR)

Mai departe definim resursa node group :

resource "aws\_iam\_role" "demo-eks-ng-role" {

  name = "demo-eks-node-group-role"

  assume\_role\_policy = jsonencode({

    Statement = [{

      Action = "sts:AssumeRole"

      Effect = "Allow"

      Principal = {

        Service = "ec2.amazonaws.com"

      }

    }]

    Version = "2012-10-17"

  })

}

resource "aws\_iam\_role\_policy\_attachment" "eks-demo-ng-WorkerNodePolicy" {

  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"

  role       = aws\_iam\_role.demo-eks-ng-role.name

}

resource "aws\_iam\_role\_policy\_attachment" "eks-demo-ng-AmazonEKS\_CNI\_Policy" {

  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy"

  role       = aws\_iam\_role.demo-eks-ng-role.name

}

resource "aws\_iam\_role\_policy\_attachment" "eks-demo-ng-ContainerRegistryReadOnly" {

  policy\_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"

  role       = aws\_iam\_role.demo-eks-ng-role.name

}

resource "aws\_eks\_node\_group" "eks-demo-node-group" {

  cluster\_name    = aws\_eks\_cluster.demo-eks-cluster.name

  node\_group\_name = "demo-eks-node-group"

  node\_role\_arn   = aws\_iam\_role.demo-eks-ng-role.arn

 subnet\_ids             = [

    aws\_subnet.private-subnet-1.id,

    aws\_subnet.private-subnet-2.id

    ]

  scaling\_config {

    desired\_size = 2

    max\_size     = 4

    min\_size     = 1

  }

  update\_config {

    max\_unavailable = 1

  }

  # Ensure that IAM Role permissions are created before and deleted after EKS Node Group handling.

  # Otherwise, EKS will not be able to properly delete EC2 Instances and Elastic Network Interfaces.

  depends\_on = [

    aws\_iam\_role\_policy\_attachment.eks-demo-ng-WorkerNodePolicy,

    aws\_iam\_role\_policy\_attachment.eks-demo-ng-AmazonEKS\_CNI\_Policy,

    aws\_iam\_role\_policy\_attachment.eks-demo-ng-ContainerRegistryReadOnly,

  ]

}

Terraform validate,plan,apply

Vad ca se creaza cele 2 instante din node group si kubectl get pods -A =>

Cele 2 Pods cu coredns sunt create in kube-system namespace

2.In continuare vom crea un EKS cluster cu module:

0-provider.tf si 1-variables.tf si copiem de la fara module:

provider "aws" {

  region = var.region

}

variable "region" {

  type        = string

  default     = "us-east-2"

  description = "AWS region"

}

variable "cidr\_block" {

  type        = string

  default     = "10.10.0.0/16"

  description = "cidr block for VPC"

}

variable "vpc\_name" {

    type = string

    default = "demo-eks-vpc"

}

variable "tags" {

  type = map(string)

  default = {

    terraform  = true

    kubernetes = "demo\_eks\_cluster"

  }

  description = "tag to apply to all resources"

}

Cream 2-vpc.tf:

Definim data source for availability zone:

data "aws\_availability\_zones" "available" {

  state = "available"

}

Apoi vom folosi vpc module:

data "aws\_availability\_zones" "available" {

  state = "available"

}

module "eks\_vpc" {

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

  name = var.vpc\_name

  cidr = var.cidr\_block

  azs             = [data.aws\_availability\_zones.available.names[0], data.aws\_availability\_zones.available.names[1]]

  private\_subnets = [cidrsubnet(var.cidr\_block, 8, 110), cidrsubnet(var.cidr\_block, 8, 120)]

  public\_subnets  = [cidrsubnet(var.cidr\_block, 8, 10), cidrsubnet(var.cidr\_block, 8, 20)]

  create\_igw = true   # by default is true

  enable\_dns\_hostnames = true   # by default is true, daca nu e e true Worker Nodes s-ar putea sa aiba probleme sa se ataseze la cluster

  enable\_nat\_gateway = true

  single\_nat\_gateway = true

  one\_nat\_gateway\_per\_az = false   # din rationamente financiare folosim 1 singur Nat GW

  create\_private\_nat\_gateway\_route = true   # by default is true

  tags = var.tags

}

Terraform init,validate,plan

3-eks.tf

La variable adaug cluster\_name si eks\_version

variable "eks\_version" {

    type = string

    default = "1.31"

    description = "eks\_version"

}

variable "cluster\_name" {

    type = string

    default = "demo-eks-cluster"

    description = "eks cluster name"

}

attach\_cluster\_encryption\_policy = false # default is true - Ce face cand e true? creaza un encryption policy si il ataseaza la Role

cluster\_endpoint\_private\_access = true => worker nodes se pot atasa la cluster in VPC

cluster\_endpoint\_public\_access = true => ma pot conecta la cluster din afara VPC

control\_plane\_subnet\_ids = concat(module.eks-vpc.private\_subnets, module.eks-vpc.public\_subnets) => combina mai multe liste intr-o noua lista

create\_cluster\_security\_group = true # default is true

bootstrap\_self\_managed\_addons = true # clusterul de EKS isi creaza singur cele 3 default addons

enable\_cluster\_creator\_admin\_permissions = true # cine creaza clusterul are automat si admin permission

dataplane\_wait\_duration = "40s" => cat asteapta clusterul pentru crearea unei resurse si afisarea pe ecran a mesajului “resursa a fost create pentru tine”

“

module "eks" {

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

  cluster\_name    = var.cluster\_name

  cluster\_version = var.eks\_version

  vpc\_id     = module.eks\_vpc.vpc\_id

  create\_iam\_role = true   # default is true

  attach\_cluster\_encryption\_policy = false   # default is true

  cluster\_endpoint\_private\_access = true

  cluster\_endpoint\_public\_access = true

  control\_plane\_subnet\_ids = concat(module.eks\_vpc.private\_subnets, module.eks\_vpc.public\_subnets)

  create\_cluster\_security\_group = true   # default is true

  cluster\_security\_group\_description = "eks cluster SG"

  bootstrap\_self\_managed\_addons = true

  enable\_cluster\_creator\_admin\_permissions = true

  dataplane\_wait\_duration = "40s"

  # Some defaults

  enable\_security\_groups\_for\_pods = true   # true by dafault

  # Override defaults

  create\_cloudwatch\_log\_group = false

  create\_kms\_key = false

  enable\_kms\_key\_rotation = false

  kms\_key\_enable\_default\_policy = false

  enable\_irsa = false   # enable your Pods to access your aws resources

  cluster\_encryption\_config = {}

  enable\_auto\_mode\_custom\_tags = false

  # EKS managed Node Groups

  create\_node\_security\_group = false

  node\_security\_group\_enable\_recommended\_rules = false

}

”

Terraform init,validate,plan

Aws eks update-kubeconfig –-region us-east-2 -–name demo-eks-cluster // ca sa ne putem conecta la cluster ca si admin

Kubectl get pods -A // vad coredns Pods in pending – nu au unde sa fie mapate

In continuare trebuie sa cream Node Group

create\_node\_security\_group = true

node\_security\_group\_enable\_recommended\_rules = true // pt ca worker nodes sa poata comunica cu Cluster nodes

# EKS managed Node Groups

  create\_node\_security\_group = true

  node\_security\_group\_enable\_recommended\_rules = true

  node\_security\_group\_description = "SG used by worker nodes to communicate with Cluster API server"

  node\_security\_group\_use\_name\_prefix = true

  subnet\_ids = module.eks\_vpc.private\_subnets

  eks\_managed\_node\_groups = {

    group1 = {

      # Starting on 1.30, AL2023 is the default AMI type for EKS managed node groups

      name = "demo-eks-node-group"

      ami\_type       = "AL2023\_x86\_64\_STANDARD"

      instance\_types = ["t3.medium"]

      capacity\_type = "SPOT"

      min\_size     = 2

      max\_size     = 4

      desired\_size = 2

    }

  }

}

Terraform init,validate,plan,apply

Dup ace se creaza cele 2 EC2 din node group1 , cele 2 Pods coredns vor fi mapate aici

In continuare vom crea fargate profile:

  fargate\_profiles = {

    profile1 = {

        selectors = [

            {

                namespace = "kube-system"

            }

        ]

    }

  }

}

Fargate profiles va adauga Iam Role cu policy AmazonEKSFargatePodExecutionRolePolicy si AmazonEKS\_CNI\_Policy.

Acum noile Pods care vor fi programate in kube-system namespace vor ateriza pe nodurile managiuite de Fargate