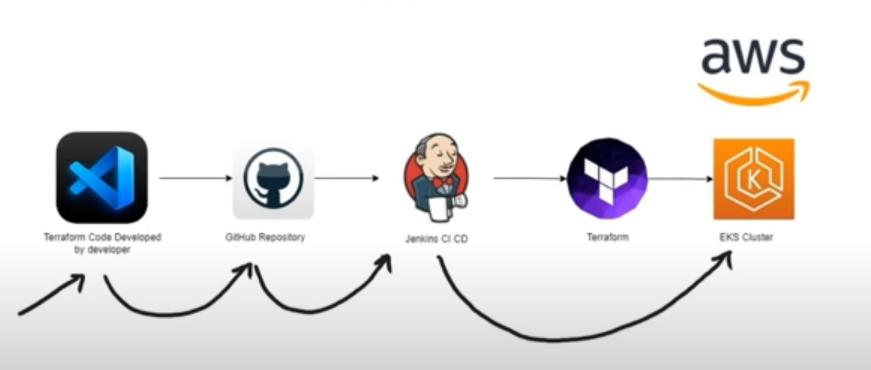
Deploy EKS cluster using Terraform modules and Jenkins:



1.Create an EC2 instance + Jenkins

2.Write terraform code for EKS

3.Push the code on GitHub

4.Create a Jenkins pipeline which is going to deploy an EKS cluster

5.Deploy the changes to AWS

6.Implement a deployment file with kubectl -> create nginx Pod

<https://github.com/gauri17-pro/terraform-jenkins-eks>

1. Trebuie sa am awscli instalat cu user vscode si access key si secret access key:

Creez provider.tf:

“provider "aws" {

  region = "us-east-1"

}

”

Creez S3 “cicd-terraform-jenkins-1985” pt backend.tf unde va fi depozitat terraform.tfstate

Backend.tf:

“terraform {

  backend "s3" {

    bucket = "cicd-terraform-eks-1985"

    key    = "jenkins/terraform.tfstate"

    region = "us-east-1"

  }

}

”

Apoi configuram datasources data.tf pt a prelua AMI din AWS:

Caut pe google “aws ami daasource terraform”

“data "aws\_ami" "example" {

  most\_recent      = true

  owners           = ["amazon"]

  filter {

    name   = "name"

    values = ["amzn2-ami-kernel-\*-hvm-\*-x86\_64-gp2"]

  }

  filter {

    name   = "root-device-type"

    values = ["ebs"]

  }

  filter {

    name   = "virtualization-type"

    values = ["hvm"]

  }

}

”

Apoi mai trebuie un datasource pt availability zone:

“data "aws\_ami" "example" {

  most\_recent      = true

  owners           = ["amazon"]

  filter {

    name   = "name"

    values = ["amzn2-ami-kernel-\*-hvm-\*-x86\_64-gp2"]

  }

  filter {

    name   = "root-device-type"

    values = ["ebs"]

  }

  filter {

    name   = "virtualization-type"

    values = ["hvm"]

  }

}

data "aws\_availability\_zones" "azs" {}

”

Mergem apoi la main.tf

Caut dupa “terraform vpc modules”

Pt avz avem nevoie de parametru names

Pt subnets nu avem nevoie de private subnets doar de 1 public subnet in cae vom avea EC2-ul nostru ce tine Jenkins. O dam ca variabila

Nu am nevoie de:

enable\_nat\_gateway = true

enable\_vpn\_gateway = true

Dar am nevoie de enable\_dns\_hostnames = true

IGW,route tables,route tables association totul e facut de acest modul din AWS!!!

#VPC

module "vpc" {

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

  name = "jenkins-vpc"

  cidr = var.vpc\_cidr

  azs             = data.aws\_availability\_zones.azs.names

  public\_subnets  = var.public\_subnets

  enable\_dns\_hostnames = true

  tags = {

    Name = "jenkins-vpc"

    Terraform = "true"

    Environment = "dev"

  }

  public\_subnets\_tags = {

    Name = "jenkins-subnet"

  }

}

#SG

#EC2

In variables.tf:

variable "vpc\_cidr"  {

    description = "VPC CIDR"

    type = string

}

variable "public\_subnets" {

    description = "Public subnet for EC2"

    type = list(string)

}

In terraform.tfvars:

vpc\_cidr = "10.0.0.0/16"

public\_subnets = ["10.0.1.0/24"]

In continuare trebuie sa ma ocup de SG

Caut pe google terraform aws module security group

Pt SG am nevoie si de egress - > ma duc la Inputs si caut egress with cidr blocks

#SG

module "sg" {

  source = "terraform-aws-modules/security-group/aws"

  name        = "jenkins-sg"

  description = "Security group for jenkins server"

  vpc\_id      = module.vpc.vpc\_id

  ingress\_with\_cidr\_blocks = [

    {

      from\_port   = 8080

      to\_port     = 8080

      protocol    = "tcp"

      description = "HTTP for jenkins"

      cidr\_blocks = "0.0.0.0/0"

    },

    {

      from\_port   = 22

      to\_port     = 22

      protocol    = "tcp"

      description = "SSH"

      cidr\_blocks = "0.0.0.0/0"

    }

  ]

  egress\_with\_cidr\_blocks = [

     {

      from\_port   = 0

      to\_port     = 0

      protocol    = "-1"

      description = "All traffic - all protocols"

      cidr\_blocks = "0.0.0.0/0"

    }

  ]

  tags = {

    Name = "jenkins-sg"

  }

}

Pt EC2 – caut pe google “terraform EC2 module”

Subnet id vine ca output de la module de VPC(public\_subnets) si SG id vine ca output de la module SG(security\_group\_id)

Vreau ca pt EC2 meu cu Jenkins sa-I ascociez public ip => la module de EC2 la inputs =>

associate\_public\_ip\_address = true

apoi vreau sa folosesc user\_data ca sa instalez Jenkins,Git,terraform si kubectl => creez fisierul Jenkins-install.sh

Ne mai trebuie si availability zones -> availability\_zone = data.aws\_availability\_zones.azs.names[0]

#EC2

module "ec2\_instance" {

  source = "terraform-aws-modules/ec2-instance/aws"

  name = "Jenkins-Server"

  instance\_type               = var.instance\_type

  key\_name                    = "demo\_key"

  monitoring                  = true

  vpc\_security\_group\_ids      = [module.sg.security\_group\_id]

  subnet\_id                   = module.vpc.public\_subnets[0]

  associate\_public\_ip\_address = true

  user\_data                   = file("jenkins-install.sh")

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

  tags = {

    Name        = "Jenkins-Server"

    Terraform   = "true"

    Environment = "dev"

  }

}

Terraform init,plan si apply

Pt instalare Jenkins pe EC2 cu Amazon:

sudo yum update -y

sudo yum install java-17-amazon-corretto -y // install java

java -version // verific Java version

sudo wget -O /etc/yum.repos.d/jenkins.repo \  
 [https://pkg.jenkins.io/redhat-stable/jenkins.repo //](https://pkg.jenkins.io/redhat-stable/jenkins.repo%20//) Add Jenkins repository

sudo rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key> // Import a key file from Jenkins-CI to enable installation from the package:

sudo yum install jenkins -y // install Jenkins

sudo systemctl enable jenkins

sudo systemctl start Jenkins

Ca sa accesam Jenkins ne trebuie parola initiala =>

ec2-user@ip-10-0-1-197 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Install Sugested plugins

arse / parolamea -> Save and Continue

In continuare vom crea EKS cluster code:

Cream si fisierul “.gitignore”: specificam fisierele care nu vor ajunge in github repository

“.terraform/\*

.terraform.lock.hcl

”

Punem toate fiserele in directorul JenkinsServer si cream EKS\_Cluster(si aici cream main.tf, data.tf,etc..)

Provider.tf:

“provider "aws" {

    region = "us-east-1"

}

”

Backend.tf:

“terraform {

    backend "s3" {

        bucket = "cicd-terraform-jenkins-123"

        key = "eks/terraform.tfstate"

        region = "us-east-1"

    }

}

”

In main.tf definim un alt VPC in care deployem un eks cluster

Copie modulul de vpc si il modificam:

Variabilele cidr si public\_subnet le luam din fisierul variables.tf la fel procedam si cu terraform.tfvars copiem cidr si public\_subnet

Terraform.tfvars:

“vpc\_cidr       = "192.168.0.0/16"

private\_subnets = ["192.168.1.0/24","192.168.2.0/24","192.168.3.0/24"]

public\_subnets = ["192.168.4.0/24","192.168.5.0/24","192.168.6.0/24"]

”

Vom deploya eks cluster in a private subnet.

In modulul de VPC sterg “map\_public\_ip\_on\_launch = true”

Avem nevoie de single nat gw

Main.tf:

#VPC

module "vpc" {

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

  name = "jenkins-vpc"

  cidr = var.vpc\_cidr

  azs = data.aws\_availability\_zones.azs.names

  private\_subnets = var.private\_subnets

  public\_subnets = var.public\_subnets

  enable\_dns\_hostnames = true

  enable\_nat\_gateway = true

  single\_nat\_gateway = true

  tags = {

    "kubernetes.io/cluster/my-eks-cluster" = "shared"

  }

  public\_subnet\_tags = {

    "kubernetes.io/cluster/my-eks-cluster" = "shared"

    "kubernetes.io/role/elb"               = 1

  }

  private\_subnet\_tags = {

    "kubernetes.io/cluster/my-eks-cluster" = "shared"

    "kubernetes.io/role/internal-elb"      = 1

  }

}

In continuare trebuie sa caut pe google “terraform eks module”si dau copy / paste la cea cu cluster addonss min\_size

Tot in main.tf adaug si

“#EKS

module "eks" {

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

  version = "~> 20.0"

  cluster\_name    = "my-eks-cluster"

  cluster\_version = "1.24"

  vpc\_id                   = module.vpc.vpc\_id

  subnet\_ids               = module.vpc.private\_subnets

  control\_plane\_subnet\_ids = ["subnet-xyzde987", "subnet-slkjf456", "subnet-qeiru789"]

  eks\_managed\_node\_groups = {

    node = {

      min\_size     = 1

      max\_size     = 3

      desired\_size = 2

      instance\_type = ["t2.small"]

    }

  }

  tags = {

    Environment = "dev"

    Terraform   = "true"

  }

}

”

Apoi cd EKS\_cluster

Terraform init,fmt,validate,plan dar nu si apply (vrem sa deployem EKS din Jenkins)

Gitbash:

Git init

Git status

git commit -m "Adding files to Repo"

Intai creez repo in github, apoi:

**…or create a new repository on the command line**

echo "# TerraformProject4" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/arse1985/TerraformProject4.git

git push -u origin main

**…or push an existing repository from the command line**

git remote add origin https://github.com/arse1985/TerraformProject4.git

git branch -M main

git push -u origin main

Apoi cream Jenkins Pipeline:

Jenkins -> New item -> “terraform-eks-cicd” -> Pipeline -> ok

Trebuie sa adaug Access key si Secret access key pt ca Jenkins va face modificari in contul meu de AWS -> Jenkins -> Manage Jenkins -> Credentials -> Global -> Add credentials -> Kind -> Secret text -> AWS\_ACCESS\_KEY\_ID si Add Credentials -> AWS\_SECRET\_ACCESS\_KEY.

Apoi Dashboard -> terraform-eks-cicd -> Configure -> Pipeline -> Pipeline script ->

pipeline {

agent any

environment {

AWS\_ACCESS\_KEY\_ID = credentials('AWS\_ACCESS\_KEY\_ID')

AWS\_SECRET\_ACCESS\_KEY = credentials('AWS\_SECRET\_ACCESS\_KEY')

AWS\_DEFAULT\_REGION = "us-east-1"

}

stages {

stage('Checkout SCM'){

steps{

script{

checkout scmGit(branches: [[name: '\*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/arse1985/TerraformProject4.git']])

}

}

}

stage('Initializing Terraform'){

steps{

script{

dir('EKS'){

sh 'terraform init'

}

}

}

}

stage('Formatting Terraform Code'){

steps{

script{

dir('EKS'){

sh 'terraform fmt'

}

}

}

}

stage('Validating Terraform'){

steps{

script{

dir('EKS'){

sh 'terraform validate'

}

}

}

}

stage('Previewing the Infra using Terraform'){

steps{

script{

dir('EKS'){

sh 'terraform plan'

}

input(message: "Are you sure to proceed?", ok: "Proceed")

}

}

}

stage('Creating/Destroying an EKS Cluster'){

steps{

script{

dir('EKS') {

sh 'terraform $action --auto-approve'

}

}

}

}

stage('Deploying Nginx Application') {

steps{

script{

dir('EKS/ConfigurationFiles') {

sh 'aws eks update-kubeconfig --name my-eks-cluster'

sh 'kubectl apply -f deployment.yaml'

sh 'kubectl apply -f service.yaml'

}

}

}

}

}

}

Pt script deschidem Pipeline syntax in fereastra noua -> Checkout from version control -> Repo URL -> <https://github.com/arse1985/TerraformProject4.git> -> in loc de master scriu “main” -> Generate pipeline script

Save -> Build now ->