**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

**Інститут комп’ютерних технологій, автоматики та метрології**

**Кафедра "Комп'ютеризовані системи автоматики"**



**ЗВІТ**

про виконання **лабораторної роботи №4**

з навчальної дисципліни**: «Хмарні технології»**

**Виконав:**

студент групи ІР-31

Борецький Б. І.

**Прийняв:**

к.т.н., асистент

Рибак В. І.

Львів — 2023

**Завдання**

* описати інфраструктуру і логіку 2 лабораторної за допомогою terraform.
* створити CI/CD який при пуші нового terraform коду в репозиторій оновить інфраструктуру.

**Виконання**

Описав інфраструктуру лабораторної роботи 2 використавши terraform

main.tf

terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 4.16"

    }

  }

  backend "s3" {

    bucket         = "terraform-state-lab4-iot"

    key            = "tfstate"

    region         = "us-east-1"

    dynamodb\_table = "terraform-state-lock"

  }

  required\_version = ">= 1.2.0"

}

provider "aws" {

  alias  = "us\_east\_1"

  region = "us-east-1"

}

resource "aws\_secretsmanager\_secret" "db-creds" {

  name = "prod/lab4/db-credentials"

}

resource "aws\_secretsmanager\_secret\_version" "secret\_credentials" {

  secret\_id = aws\_secretsmanager\_secret.db-creds.id

  secret\_string = jsonencode({

    "db\_username" = var.db\_username

    "db\_password" = var.db\_password

  })

}

resource "aws\_default\_vpc" "default" {

  tags = {

    Name = "Default VPC"

  }

}

resource "aws\_internet\_gateway" "gw" {

  vpc\_id = aws\_default\_vpc.default.id

}

resource "aws\_subnet" "us\_east\_1a" {

  vpc\_id     = aws\_default\_vpc.default.id

  cidr\_block = "172.31.16.0/20"

  map\_public\_ip\_on\_launch = true

  availability\_zone       = "us-east-1a"

}

resource "aws\_subnet" "us\_east\_1b" {

  vpc\_id     = aws\_default\_vpc.default.id

  cidr\_block = "172.31.32.0/20"

  map\_public\_ip\_on\_launch = true

  availability\_zone       = "us-east-1b"

}

resource "aws\_db\_instance" "lab4\_db" {

  allocated\_storage                   = 20

  availability\_zone                   = aws\_subnet.us\_east\_1a.availability\_zone

  backup\_retention\_period             = 1

  ca\_cert\_identifier                  = "rds-ca-2019"

  db\_name                             = "lab4\_db"

  engine                              = "mysql"

  engine\_version                      = "8.0.33"

  max\_allocated\_storage               = 0

  identifier                          = "database-lab4-iot"

  iops                                = 0

  tags                                = {}

  port                                = 3306

  publicly\_accessible                 = true

  instance\_class                      = "db.t3.micro"

  username                            = jsondecode(aws\_secretsmanager\_secret\_version.secret\_credentials.secret\_string)["db\_username"]

  password                            = jsondecode(aws\_secretsmanager\_secret\_version.secret\_credentials.secret\_string)["db\_password"]

  storage\_encrypted                   = true

  skip\_final\_snapshot                 = true

  customer\_owned\_ip\_enabled           = false

  deletion\_protection                 = false

  enabled\_cloudwatch\_logs\_exports     = []

  iam\_database\_authentication\_enabled = false

  vpc\_security\_group\_ids              = [aws\_security\_group.default.id, aws\_security\_group.rds-ec2-1.id]

}

asg\_for\_ecs.tf

resource "aws\_autoscaling\_group" "lab4\_app\_asg" {

  name                    = "asg-lab4-app"

  capacity\_rebalance      = false

  default\_cooldown        = 300

  default\_instance\_warmup = 0

  enabled\_metrics         = []

  suspended\_processes     = []

  min\_size                = 1

  max\_size                = 3

  target\_group\_arns    = [aws\_lb\_target\_group.lab4\_app\_tg.arn]

  termination\_policies = []

  vpc\_zone\_identifier  = [aws\_subnet.us\_east\_1a.id, aws\_subnet.us\_east\_1b.id]

  tag {

    key                 = "AmazonECSManaged"

    value               = true

    propagate\_at\_launch = true

  }

  launch\_template {

    id      = aws\_launch\_template.ecs\_launch\_template\_lab4\_app.id

    version = "$Default"

  }

  #   terraform import aws\_autoscaling\_group.lab4\_app\_asg asg-lab4-app

}

resource "aws\_autoscaling\_group" "lab4\_stresstest\_asg" {

  name                      = "asg-stresstest-lab4-app"

  capacity\_rebalance        = false

  default\_cooldown          = 300

  default\_instance\_warmup   = 0

  enabled\_metrics           = []

  suspended\_processes       = []

  min\_size                  = 1

  max\_size                  = 1

  wait\_for\_capacity\_timeout = 0

  termination\_policies = []

  vpc\_zone\_identifier  = [aws\_subnet.us\_east\_1a.id, aws\_subnet.us\_east\_1b.id]

  tag {

    key                 = "AmazonECSManaged"

    value               = true

    propagate\_at\_launch = true

  }

  launch\_template {

    id      = aws\_launch\_template.ecs\_launch\_template\_lab4\_stresstest\_app.id

    version = "$Default"

  }

  #   terraform import aws\_autoscaling\_group.lab4\_stresstest\_asg asg-stresstest-lab4-app

}

iam\_roles.tf

data "aws\_iam\_policy" "ssm\_readonly\_access" {

  name = "AmazonSSMReadOnlyAccess"

}

data "aws\_iam\_policy" "container\_service\_for\_ec2\_role" {

  name = "AmazonEC2ContainerServiceforEC2Role"

}

resource "aws\_iam\_role" "ecs\_task\_execution\_role" {

  name = "ecsTaskExecutionRole"

  assume\_role\_policy = jsonencode({

    Version = "2008-10-17"

    Statement = [

      {

        Action = "sts:AssumeRole"

        Effect = "Allow"

        Sid    = ""

        Principal = {

          Service = "ecs-tasks.amazonaws.com"

        }

      },

    ]

  })

}

resource "aws\_iam\_role" "ecs\_task\_role" {

  name        = "ecs-task-role"

  description = "Allows ECS tasks to call AWS services on your behalf."

  assume\_role\_policy = jsonencode({

    Version = "2012-10-17"

    Statement = [

      {

        Action = "sts:AssumeRole"

        Effect = "Allow"

        Sid    = ""

        Principal = {

          Service = "ecs-tasks.amazonaws.com"

        }

      },

    ]

  })

  managed\_policy\_arns = [

    data.aws\_iam\_policy.ssm\_readonly\_access.arn,

  ]

}

resource "aws\_iam\_role" "ecs\_instance\_role" {

  name = "ecsInstanceRole"

  assume\_role\_policy = jsonencode({

    Version = "2008-10-17"

    Statement = [

      {

        Action = "sts:AssumeRole"

        Effect = "Allow"

        Sid    = ""

        Principal = {

          Service = "ec2.amazonaws.com"

        }

      },

    ]

  })

  managed\_policy\_arns = [

    data.aws\_iam\_policy.container\_service\_for\_ec2\_role.arn,

  ]

}

resource "aws\_iam\_instance\_profile" "profile\_ecs\_instance\_role" {

  role = aws\_iam\_role.ecs\_instance\_role.name

}

data "aws\_iam\_policy" "ecs\_service\_role\_policy" {

  name = "AmazonECSServiceRolePolicy"

}

resource "aws\_iam\_role" "service\_role\_for\_ecs" {

  name        = "AWSServiceRoleForECS"

  description = "Role to enable Amazon ECS to manage your cluster."

  path        = "/aws-service-role/ecs.amazonaws.com/"

  assume\_role\_policy = jsonencode({

    Version = "2012-10-17"

    Statement = [

      {

        Action = "sts:AssumeRole"

        Effect = "Allow"

        Principal = {

          Service = "ecs.amazonaws.com"

        }

      },

    ]

  })

  managed\_policy\_arns = [

    data.aws\_iam\_policy.ecs\_service\_role\_policy.arn,

  ]

}

# data "aws\_iam\_policy" "ecs\_service\_role\_policy" {

#   name = "AmazonECSServiceRolePolicy"

# }

CICD описав також за допомогою terraform

resource "aws\_codecommit\_repository" "terraform\_iaac\_repo" {

  repository\_name = "terraform-iaac-repo"

  description     = "Terraform resources"

}

output "repository\_clone\_url\_ssh" {

  value = aws\_codecommit\_repository.terraform\_iaac\_repo.clone\_url\_http

}

resource "aws\_s3\_bucket" "codepipeline\_terrafrom\_s3\_bucket" {

  bucket = "codepipeline-terraform-s3-bucket-419801"

}

resource "aws\_s3\_bucket\_acl" "codepipeline\_terraform\_bucket\_acl" {

  bucket = aws\_s3\_bucket.codepipeline\_terrafrom\_s3\_bucket.id

  acl    = "private"

}

resource "aws\_cloudwatch\_log\_group" "codebuild\_terraform\_pipeline" {

  name = "codebuild/terraform\_pipeline"

}

resource "aws\_iam\_role" "codebuild\_terraform\_role" {

  name               = "codebuild-terrafrom-service-role"

  assume\_role\_policy = <<EOF

{

  "Version": "2012-10-17",

  "Statement": [

    {

      "Effect": "Allow",

      "Principal": {

        "Service": "codebuild.amazonaws.com"

      },

      "Action": "sts:AssumeRole"

    }

  ]

}

EOF

}

resource "aws\_iam\_role\_policy" "codebuild\_terraform\_policy" {

  name   = "codebuild\_terraform\_policy"

  role   = aws\_iam\_role.codebuild\_terraform\_role.id

  policy = <<EOF

{

    "Statement": [

        {

            "Action": "\*",

            "Resource": "\*",

            "Effect": "Allow"

        }

    ],

    "Version": "2012-10-17"

  }

EOF

}

resource "aws\_iam\_role" "codepipeline\_terraform\_role" {

  name               = "codepipeline-terraform-service-role"

  assume\_role\_policy = <<EOF

  {

    "Version": "2012-10-17",

    "Statement": [

      {

        "Effect": "Allow",

        "Principal": {

          "Service": "codepipeline.amazonaws.com"

        },

        "Action": "sts:AssumeRole"

      }

    ]

  }

EOF

}

resource "aws\_iam\_role\_policy" "codepipeline\_terraform\_policy" {

  name   = "codepipeline\_terraform\_policy"

  role   = aws\_iam\_role.codepipeline\_terraform\_role.id

  policy = <<EOF

{

  "Statement": [

      {

          "Action": "\*",

          "Resource": "\*",

          "Effect": "Allow"

      }

  ],

  "Version": "2012-10-17"

}

EOF

}

resource "aws\_codebuild\_project" "terraform" {

  name          = "terraform\_resources"

  description   = "Apply terrafrom resource"

  build\_timeout = "5"

  service\_role  = aws\_iam\_role.codebuild\_terraform\_role.arn

  artifacts {

    type = "CODEPIPELINE"

  }

  environment {

    compute\_type                = "BUILD\_GENERAL1\_SMALL"

    image                       = "aws/codebuild/standard:6.0"

    type                        = "LINUX\_CONTAINER"

    image\_pull\_credentials\_type = "CODEBUILD"

    privileged\_mode             = true

    environment\_variable {

      name  = "TF\_COMMAND"

      value = "apply"

    }

    environment\_variable {

      name  = "TF\_VERSION"

      value = "1.6.3"

    }

    environment\_variable {

      name  = "db\_username"

      value = var.db\_username

    }

    environment\_variable {

      name  = "db\_password"

      value = var.db\_password

    }

  }

  logs\_config {

    cloudwatch\_logs {

      group\_name  = "codepipeline"

      stream\_name = "terraform"

    }

  }

  source {

    type                = "CODEPIPELINE"

    git\_clone\_depth     = 0

    insecure\_ssl        = false

    report\_build\_status = false

  }

}

resource "aws\_codepipeline" "terrafrom\_codepipeline" {

  name     = "terrafrom-pipeline"

  role\_arn = aws\_iam\_role.codepipeline\_terraform\_role.arn

  artifact\_store {

    location = aws\_s3\_bucket.codepipeline\_terrafrom\_s3\_bucket.bucket

    type     = "S3"

  }

  stage {

    name = "Source"

    action {

      name             = "Source"

      category         = "Source"

      owner            = "AWS"

      provider         = "CodeCommit"

      version          = "1"

      output\_artifacts = ["SourceArtifact"]

      configuration = {

        RepositoryName = aws\_codecommit\_repository.terraform\_iaac\_repo.repository\_name

        BranchName     = "main"

      }

    }

  }

  stage {

    name = "Build"

    action {

      name             = "Build"

      category         = "Build"

      owner            = "AWS"

      region           = var.region

      provider         = "CodeBuild"

      input\_artifacts  = ["SourceArtifact"]

      output\_artifacts = ["BuildArtifact"]

      version          = "1"

      configuration = {

        ProjectName = aws\_codebuild\_project.terraform.name

      }

    }

  }

}

**Висновок**

Виконавши цю лабораторну роботу, я навчився працювати з terraform.