ДОДАТОК А

Блок-схема алгоритму перевірки інфрастуктури завдань у хмарі

ДОДАТОК Б

Вихідний код Jenkins пайплайну для перевірки інфрастуктури завдань у хмарі

```
pipeline {
  agent none

  options {
    buildDiscarder(logRotator(numToKeepStr: '100'))
    ansiColor('xterm')
    timestamps()
  }
  parameters {
        string(name:"pipeline_branch", description:"Branch to run pipeline from",
        defaultValue:  "main")
    choice(name:"task",description:" ", choices: [
      "ec2_session_manager_setup",
      "iam_managed_permissions",
      "s3_static_site",
      "vpc_security_groups"
    ])
    string(name:"student_id", description:"student s3 key", defaultValue:  "user1")
      string(name:"student_aws_account_id", description:"student aws account id",
        defaultValue:  "")
  }
```

```
environment {
    stashName="root_stash"
    tfWorkingDir = "tf_modules/${params.task}/infra"
    tfValidationDir = "tf_modules/${params.task}/verification"

    // AWS_DEFAULT_PROFILE = ""
    // AWS_PROFILE = "${env.AWS_DEFAULT_PROFILE}"
    AWS_DEFAULT_REGION = 'eu-central-1'

    AWS_REGION = "${env.AWS_DEFAULT_REGION}"
    TF_IN_AUTOMATION = 'true'

    // tf access variales
    TF_VAR_aws_region = "${AWS_DEFAULT_REGION}"
                                    TF_VAR_aws_role_arn      =      "arn:aws:iam::$
        {params.student_aws_account_id}:role/cloud-mentor-connector-role"

    // tfbackendBucket = "cloud-mentor-tf-state"
    // tfbackendRegion = 'eu-central-1'
     tfbackendkey    = "${params.student_id}/${params.student_aws_account_id}/$
        {params.task}/infra/terraform.tfstate"
                            tfbackendCompeletedkey     =     "${params.student_id}/$
        {params.student_aws_account_id}/$
        {params.task}/infra/completed_task.tfstate"
                        tfbackendUncompeletedkey     =     "${params.student_id}/$
        {params.student_aws_account_id}/$
        {params.task}/infra/uncompleted_task.tfstate"
                    tfVerificationbackendkey              =     "${params.student_id}/$
        {params.student_aws_account_id}/$
        {params.task}/verification/terraform.tfstate"
    taskResourcesDeployed = ""
}

stages {
```

```
stage("plan task resources") {

   agent {label "master"}

   steps {
      script {

         dir(tfWorkingDir){
                                    sh "terraform init -backend-config='bucket=$
     {env.CLOUD_MENTOR_TF_BUCKET}'          -backend-config='key=$
     {env.tfbackendkey}'                    -backend-config='region=$
     {env.CLOUD_MENTOR_BACKEND_REGION}'"
            env.TF_VAR_deploy_task_resources = "true"
            sh "terraform plan -out tfplan"
         }
      }
      stash name: stashName, allowEmpty: true
   }
}

stage("approve task resources deploy") {
   agent none
   steps {
      script {
         input(
            id: 'userDeployInput', message: "Approve deploy task infrastructure?",
     ok: "Approve"
         )
      }
   }
}
stage("deploy task resources") {

   agent {label "master"}
```

```
    steps {
        script{
            unstash name: stashName
            dir(tfWorkingDir){
                env.TF_VAR_deploy_task_resources = "true"
                sh "terraform apply --auto-approve tfplan"
                        sh "aws s3 cp s3://${env.CLOUD_MENTOR_TF_BUCKET}/$
      {env.tfbackendkey}          s3://${env.CLOUD_MENTOR_TF_BUCKET}/$
      {env.tfbackendCompeletedkey}"

                // remove task solution resources
                env.TF_VAR_deploy_task_resources = "false"
                sh "terraform apply --auto-approve >/dev/null"
                        sh "aws s3 cp s3://${env.CLOUD_MENTOR_TF_BUCKET}/$
      {env.tfbackendkey}          s3://${env.CLOUD_MENTOR_TF_BUCKET}/$
      {env.tfbackendUncompeletedkey}"
                        sh "aws s3 cp s3://${env.CLOUD_MENTOR_TF_BUCKET}/$
      {env.tfbackendCompeletedkey}                                    s3://$
      {env.CLOUD_MENTOR_TF_BUCKET}/${env.tfbackendkey}"
            }
            taskResourcesDeployed = true
            stash name: stashName, allowEmpty: true
        }
    }
}

stage("validate task"){
    agent none

    options {
        timeout(time: 6000, unit: 'SECONDS')
        retry(99999)
    }

    stages {
```

```
 stage("check task input") {
    agent none
    steps {
       script {
          input(
                  id: 'userInput', message: "Is the task completed and ready to
check?", ok: "Check",
          )
       }
    }
 }

 stage("check task") {
    agent {label "master"}
    steps {
       script {
          unstash name: stashName

          dir(env.tfValidationDir){
             // deploy validation module
             echo "Validation is in progress, it might take a few minutes."
                      env.TF_VAR_infra_backend_config="{ bucket = \"$
{env.CLOUD_MENTOR_TF_BUCKET}\", key = \"$
{env.tfbackendkey}\", region = \"$
{env.CLOUD_MENTOR_BACKEND_REGION}\" }"
                              sh "terraform init -backend-config='bucket=$
{env.CLOUD_MENTOR_TF_BUCKET}' -backend-config='key=$
{env.tfVerificationbackendkey}' -backend-config='region=$
{env.CLOUD_MENTOR_BACKEND_REGION}'"
             sh "terraform apply --auto-approve"
               validationOutput = readJSON text: sh (script: "terraform output
-json", returnStdout: true)
          }

          if (validationOutput.result.value != "success" ){
```

```
                    dir(env.tfWorkingDir){
                        env.TF_VAR_deploy_task_resources = "true"
                        sh (script: "terraform plan -out=tfplan >/dev/null")

                        sh (script: "terraform show -json tfplan > tfplan.json")
                        changed_resources = []
                        tfPlanProcessed = readJSON file: "tfplan.json"
                        // println(tfPlanProcessed)
                    }
                    tfPlanProcessed.resource_changes.each{
                        actions = it.change.actions
                        // println(actions)
                        if (actions.contains("update") || actions.contains("create")) {
                            changed_resources.add(it)
                        }
                    }

                                                echo "Steps left to finish the task -> '$
        {changed_resources.size()}'."
                    error "Task check failed -> The task has not been solved!"

                }

                stash name: stashName, allowEmpty: true
            }
        }
    }

  }
}
post {
  always {
    node('master') {
```

```
            script {
                echo "Going to clean up verification resources"
                try{
                    unstash name: stashName
                    dir(env.tfValidationDir){
                        // destroy validation module
                        sh "terraform destroy --auto-approve"
                        env.TF_VAR_infra_backend_config=""
                    }
                }
                catch (Exception){
                    echo "No verification resource to clean up"
                }

                echo "Going to clean up task resources."
                dir(tfWorkingDir) {
                    try{
                            sh "aws s3 cp s3://${env.CLOUD_MENTOR_TF_BUCKET}/$
            {env.tfbackendCompeletedkey}                                    s3://$
            {env.CLOUD_MENTOR_TF_BUCKET}/${env.tfbackendkey}"
                        env.TF_VAR_deploy_task_resources = "true"
                        sh "terraform destroy --auto-approve"
                    }
                    catch (Exception){
                        echo "No task resource to clean up"
                    }

                }

                cleanWs()
            }
        }
    }
}
```

ДОДАТОК В

Terraform код з описом інфраструктури задання для створення маршрутизаціїї між публічною та приватною підмережею

```
// ec2.tf
data "aws_ami" "amazon_linux2" {
  most_recent = true

  owners = ["amazon"]

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*-x86_64-ebs"]
  }
}

resource "aws_instance" "private" {
  ami           = data.aws_ami.amazon_linux2.id
  instance_type = "t2.micro"
  key_name      = aws_key_pair.default.id
  subnet_id = aws_subnet.private_subnet.id
  vpc_security_group_ids = [aws_security_group.server_sg.id]

tags = {
    Name  = "Private server"
    Owner = "E-mentor"
  }
}

resource "aws_instance" "bastion" {
  ami           = data.aws_ami.amazon_linux2.id
  instance_type = "t2.micro"
  key_name      = aws_key_pair.default.id
  subnet_id = aws_subnet.public_subnet.id
  vpc_security_group_ids = [aws_security_group.bastion_sg.id]
  associate_public_ip_address = true

  tags = {
    Name  = "Public bastion"
    Owner = "E-mentor"
  }
}

resource "aws_key_pair" "default" {
  key_name   = "cloud-mentor-keypair"
```

```
  public_key = tls_private_key.ssh.public_key_openssh
}

resource "tls_private_key" "ssh" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "aws_ssm_parameter" "secret" {
  name        = "cloud-mentor-ssh-key"
  description = "SSH private key for instance in private and public zone"
  type        = "SecureString"
  value       = tls_private_key.ssh.private_key_pem

  tags = {
    Owner = "E-mentor"
  }
}


// locals.tf
locals {
  aws_profile  = var.aws_profile == "" ? null : var.aws_profile
  aws_role_arn = var.aws_role_arn == "" ? null : var.aws_role_arn
  default_tags = {
    Owner = "E-mentor"
  }

  server_sg_ingress = [
    {
      description = "ICMP"
      from_port   = -1
      to_port     = -1
      protocol    = "icmp"
      cidr_blocks = [var.public_subnet_cidr]
      prefix_list_ids = null
      security_groups = null
      ipv6_cidr_blocks = null
      self = null
    },
    {
      description = "SSH"
      from_port   = 22
      to_port     = 22
      protocol    = "tcp"
      cidr_blocks = [var.public_subnet_cidr]
      prefix_list_ids = null
      security_groups = null
      ipv6_cidr_blocks = null
      self = null
    }
```

```
]

server_sg_egress = [
  {
    description = "All Egress"
    from_port   = -1
    to_port     = -1
    protocol    = "All"
    cidr_blocks = ["0.0.0.0/0"]
    prefix_list_ids = null
    security_groups = null
    ipv6_cidr_blocks = null
    self = null
  }
]

bastion_sg_ingress = [
  {
    description = "ICMP"
    from_port   = -1
    to_port     = -1
    protocol    = "icmp"
    cidr_blocks = ["0.0.0.0/0"]
    prefix_list_ids = null
    security_groups = null
    ipv6_cidr_blocks = null
    self = null
  },
  {
    description = "SSH"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
    prefix_list_ids = null
    security_groups = null
    ipv6_cidr_blocks = null
    self = null
  }
]

bastion_sg_egress = [
  {
    description = "All Egress"
    from_port   = -1
    to_port     = -1
    protocol    = "All"
    cidr_blocks = ["0.0.0.0/0"]
    prefix_list_ids = null
    security_groups = null
    ipv6_cidr_blocks = null
```

```
      self = null
    }
  ]
}

//output.tf
output "private_instance_ip" {
  value = aws_instance.private.private_ip
}

output "public_instance_ip" {
  value = aws_instance.bastion.public_ip
}

output "ssh_pem_key" {
  sensitive = true
  value = tls_private_key.ssh.private_key_pem
}

//provider.tf
terraform {

  backend "s3" {
    bucket = ""
    key    = ""
    region = ""
  }

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 3.0"
    }

    local = {
      source  = "hashicorp/local"
      version = "2.1.0"
    }

    null = {
      source  = "hashicorp/null"
      version = "2.1.2"
    }

    template = {
      source  = "hashicorp/template"
      version = "2.1.2"
    }
  }
}
```

```
provider "aws" {
  profile = var.aws_profile != "" ? var.aws_profile : null
  region  = var.aws_region != "" ? var.aws_region : null
  assume_role {
    role_arn = var.aws_role_arn != "" ? var.aws_role_arn : null
  }

  default_tags {
    tags = {
      Owner = "cloud-mentor"
    }
  }
}

// variables.tf
variable "aws_profile" {
  description = "The AWS profile to create infra with"
  default     = ""
}

variable "aws_role_arn" {
  description = "The AWS profile to create infra with"
  default     = ""
}

variable "aws_region" {
  description = "The AWS region where to create infra"
  default     = ""
}

variable "deploy_task_resources" {
  description = "Special variable to deploy and remove task related resources"
  default     = true
}

variable "vpc_cidr" {
  description = "Initial VPC for network task"
  default     = "10.0.0.0/16"
}

variable "public_subnet_cidr" {
  description = "CIDR for the public subnet"
  default     = "10.0.1.0/24"
}

variable "private_subnet_cidr" {
  description = "CIDR for the private subnet"
  default     = "10.0.2.0/24"
}
```

```
// vpc_resources.tf

# Define our VPC
resource "aws_vpc" "cloud_mentor_vpc" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = true

  tags = {
    Name  = "Cloud Mentor VPC"
    Owner = "E-mentor"
  }
}

# Define the public subnet
resource "aws_subnet" "public_subnet" {
  vpc_id            = aws_vpc.cloud_mentor_vpc.id
  cidr_block        = var.public_subnet_cidr
  availability_zone = "eu-central-1a"

  tags = {
    Name  = "Cloud Mentor Public Subnet"
    Owner = "E-mentor"
  }
}

# Define the private subnet
resource "aws_subnet" "private_subnet" {
  vpc_id            = aws_vpc.cloud_mentor_vpc.id
  cidr_block        = var.private_subnet_cidr
  availability_zone = "eu-central-1b"

  tags = {
    Name  = "Cloud Mentor Private Subnet"
    Owner = "E-mentor"
  }
}

# Define the internet gateway
resource "aws_internet_gateway" "gw" {
//  count  = var.deploy_task_resources ? 1 : 0
  vpc_id = aws_vpc.cloud_mentor_vpc.id

  tags = {
    Name  = "Cloud Mentor IGW"
    Owner = "E-mentor"
  }
}

# Define the route table
resource "aws_route_table" "public_rt" {
```

```
  count  = var.deploy_task_resources ? 1 : 0
  vpc_id = aws_vpc.cloud_mentor_vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gw.id
  }

  tags = {
    Name  = "Public Subnet RT"
    Owner = "E-mentor"
  }
}

# Assign the route table to the public Subnet
resource "aws_route_table_association" "public-rt" {
  count          = var.deploy_task_resources ? 1 : 0
  subnet_id      = aws_subnet.public_subnet.id
  route_table_id = aws_route_table.public_rt[count.index].id
}

# Define the security group for public subnet
resource "aws_security_group" "bastion_sg" {
  name        = "bastion_sg"
  description = "Allow SSH access / ICMP"
  vpc_id      = aws_vpc.cloud_mentor_vpc.id

  ingress = var.deploy_task_resources ? local.bastion_sg_ingress : null
  egress  = var.deploy_task_resources ? local.server_sg_egress : null

  tags = {
    Name  = "Bastion SG"
    Owner = "E-mentor"
  }
}

# Define the security group for private subnet
resource "aws_security_group" "server_sg" {
  name        = "server_sg"
  description = "Allow traffic from public subnet"
  vpc_id      = aws_vpc.cloud_mentor_vpc.id

  ingress = var.deploy_task_resources ? local.server_sg_ingress : null
  egress  = var.deploy_task_resources ? local.server_sg_egress : null

  tags = {
    Name  = "Private instance SG"
    Owner = "E-mentor"
  }
}
```

```
// tests.tf

locals {
  tests_result = module.check_ssh_connection.exitstatus == 0 ? "success" : "fail
ed"
}

data "terraform_remote_state" "infra" {
  backend = "s3"
  config  = var.infra_backend_config
}

data "template_file" "ssh_config" {
  template = file("templates/ssh_config")

  vars = {
    private_instance_ip = data.terraform_remote_state.infra.outputs.private_inst
ance_ip
    public_instance_ip  = data.terraform_remote_state.infra.outputs.public_insta
nce_ip
  }
}

resource "null_resource" "provision_ssh_key" {
  triggers = {
    always_run = timestamp()
  }

  provisioner "local-exec" {
    command     = "mkdir -p ~/.ssh; cat > ~/.ssh/cloud-mentor.pem <<EOL\n$
{data.terraform_remote_state.infra.outputs.ssh_pem_key}\nEOL; chmod 400 ~/.ssh/c
loud-mentor.pem"
    interpreter = ["/bin/bash", "-c"]
  }
}

resource "null_resource" "generate_ssh_config" {
  triggers = {
    always_run = timestamp()
  }

  provisioner "local-exec" {
    command = "cat >> ~/.ssh/config <<EOL\n$
{data.template_file.ssh_config.rendered}\nEOL"
  }

  depends_on = [
    null_resource.provision_ssh_key
  ]
}
```

```
module "check_ssh_connection" {
  source = "matti/resource/shell"

  depends = [
    null_resource.generate_ssh_config
  ]

  environment = {
    PRIVATE_INSTANCE_IP = data.terraform_remote_state.infra.outputs.private_inst
ance_ip
  }

  command = "eval `ssh-agent`; ssh -o ConnectTimeout=10 -i ~/.ssh/cloud-
mentor.pem -o StrictHostKeyChecking=no ec2-user@$PRIVATE_INSTANCE_IP;"
  trigger = timestamp()
}

output "result" {
  value = local.tests_result
}

// ssh_config

Host ${public_instance_ip}
    ForwardAgent yes
    User ec2-user
    IdentityFile ~/.ssh/cloud-mentor.pem
    Hostname ${public_instance_ip}
    ServerAliveInterval 120

Host ${private_instance_ip}
    ProxyCommand ssh -W %h:%p ${public_instance_ip}
    IdentityFile ~/.ssh/cloud-mentor.pem
    User ec2-user
```

ДОДАТОК Г

Слайди презентації

ДОДАТОК Д

Наукова публікація на тему кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	82 с.
	Графічні документи	
	Слайди презентації	14

Змін	Арк.	Номер докум.	Підпис	Дата			
Розроб.		Калиняк				Літ.	Аркушів

Перевір.	Костромицький			Автоматизація перевірки інфраструктурного рішення в хмарі Відомість атестаційної роботи			1	1
Н. контр.	Костромицький				ХНУРЕ кафедра ІМІ			
Затв.	Безрук							