ДОДАТОК А

Dockerfile для створення Jenkins образу

```
FROM jenkins/jenkins:lts

ARG TF_VERSION='0.14.8'
ARG TG_VERSION='v0.28.24'
ARG LOCAL_BIN='/usr/bin'
ENV LOCAL_BIN=${LOCAL_BIN}
ENV TF_VERSION=${TF_VERSION}
ENV TG_VERSION=${TG_VERSION}
ENV AWS_PAGER=""

USER root
RUN wget -O terraform.zip "https://releases.hashicorp.com/terraform/${TF_VERSION}/terraform_$
{TF_VERSION}_linux_amd64.zip" \
   && unzip terraform.zip \
   && mv terraform /usr/bin/ \
   && rm terraform.zip
RUN wget -O terragrunt https://github.com/gruntwork-io/terragrunt/releases/download/$
{TG_VERSION}/terragrunt_linux_arm64 \
   && chmod +x terragrunt \
   && mv terragrunt /usr/bin/
# install pre-commit
RUN apt update && export DEBIAN_FRONTEND=noninteractive && apt -y upgrade \
   && apt -y install --no-install-recommends python3-pip gawk \
   && python3 -m pip install -U setuptools \
   && curl https://pre-commit.com/install-local.py | python3 - \
   && wget -O terraform-docs https://github.com/terraform-docs/terraform-
docs/releases/download/v0.12.1/terraform-docs-v0.12.1-linux-amd64 \
      && chmod +x terraform-docs \
      &&  mv terraform-docs /usr/bin/ \
   && wget -O tfsec https://github.com/tfsec/tfsec/releases/download/v0.39.21/tfsec-linux-amd64 \
      && chmod +x tfsec \
      && mv tfsec /usr/bin/ \
   && wget -O tflint.zip https://github.com/terraform-
linters/tflint/releases/download/v0.26.0/tflint_linux_amd64.zip \
      && unzip tflint.zip \
```

```
      && mv tflint /usr/bin/ \
      && rm tflint.zip \
   && python3 -m pip install -U checkov
RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip" \
   && unzip awscliv2.zip \
   && bash -e ./aws/install -i /opt/aws-cli -b ${LOCAL_BIN} \
   && chown -R jenkins:jenkins /opt/aws-cli \
   && chown jenkins:jenkins ${LOCAL_BIN}/aws \
   && echo "complete -C '${LOCAL_BIN}/aws_completer' aws" >> /etc/profile \
   && rm awscliv2.zip \
   && rm -rf ./aws
RUN jenkins-plugin-cli --plugins pipeline-model-definition scm-api git credentials ansicolor
USER jenkins
```

ДОДАТОК Б

Код Jenkins конвеєра для перевірки завдання з доступу до віртуальної машини

```
pipeline {
  agent none

  options {
    buildDiscarder(logRotator(numToKeepStr: '100'))
    // timeout(time: 300, unit: 'SECONDS')
    ansiColor('xterm')
    timestamps()
  }
  parameters {
    choice(name:"dir",description:" ", choices: ["connect_to_ec2",] )
  }

  environment {
    stashName="root_stash"
    tfWorkingDir = "tf_modules/${params.dir}/infra"
    tfValidationDir = "tf_modules/${params.dir}/verification"
    AWS_DEFAULT_PROFILE = "bv_03_tf_admin"
    AWS_PROFILE = "${env.AWS_DEFAULT_PROFILE}"
    AWS_DEFAULT_REGION = 'us-east-1'
    AWS_REGION = "${env.AWS_DEFAULT_REGION}"
    TF_IN_AUTOMATION = 'true'
    tfbackendBucket = "cloud-mentor-bv-tf-states"
    tfbackendkey    = "bastion/terraform.tfstate"
    tfbackendCompeletedkey = "bastion/completed_task.tfstate"
    tfbackendUncompeletedkey = "bastion/uncompleted_task.tfstate"
    taskRessourcesDeployed = ""
  }

  stages {
    stage("plan task ressources") {

      agent {label "master"}
```

```
        steps {
            script {
                dir(tfWorkingDir){
                    sh "terraform init"
                    env.TF_VAR_deploy_task_resources = "true"
                    sh "terraform plan -out tfplan"
                }
            }
            stash name: stashName, allowEmpty: true
        }
    }

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

        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://${env.tfbackendBucket}/${env.tfbackendkey} s3://$
{env.tfbackendBucket}/${env.tfbackendCompeletedkey}"

                    // remove task solution ressources
                    env.TF_VAR_deploy_task_resources = "false"
                    sh "terraform apply --auto-approve >/dev/null"
```

```
                sh "aws s3 cp s3://${env.tfbackendBucket}/${env.tfbackendkey} s3://${env.tfbackendBucket}/${env.tfbackendUncompeletedkey}"
                sh "aws s3 cp s3://${env.tfbackendBucket}/${env.tfbackendCompeletedkey} s3://${env.tfbackendBucket}/${env.tfbackendkey}"
            }
            taskRessourcesDeployed = true
            stash name: stashName, allowEmpty: true
        }
    }
}

stage("validate task"){
    agent none

    options {
        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(tfWorkingDir){
                        env.TF_VAR_deploy_task_resources = "true"
                        tfPlanCode = sh (
```

```
                        script: "terraform plan -detailed-exitcode >/dev/null",
                        returnStatus: true
                    )
                    echo "tfPlanCode=${tfPlanCode}"
                }

                if (tfPlanCode != 0 ){
                    error "Task check failed -> The task has not been solved!"
                }

                stash name: stashName, allowEmpty: true
            }
        }
    }

  }
}
post {
    always {
        node('master') {
            script {
                unstash name: stashName
                echo "Going to clean up task ressources."
                dir(tfWorkingDir) {
                    try{
                        sh "aws s3 cp s3://${env.tfbackendBucket}/${env.tfbackendCompeletedkey}
s3://${env.tfbackendBucket}/${env.tfbackendkey}"
                        env.TF_VAR_deploy_task_resources = "true"
                        sh "terraform destroy --auto-approve"
                    }
                    catch (Exception){
                        echo "No task ressource to clean up"
                    }

                }
                cleanWs()
            }
        }
```

```
        }
    }
}
```

ДОДАТОК В

Terraform код з описом інфраструктури для задання з доступу до віртуальної машини

```
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 put task
ressources in state"
  default = false
}

data "aws_ami" "amazon_linux2" {
  most_recent = true

  owners = ["amazon"]

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

resource "aws_iam_role" "ec2_ssm_role" {
```

```
  name = "ec2_ssm_role"

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

  tags = merge(local.default_tags, {
    Name = "private_instance"
  })
}

resource "aws_iam_role_policy_attachment" "ec2_ssm_role_policy" {
  count = var.deploy_task_resources ? 1 : 0
  role       = aws_iam_role.ec2_ssm_role.name
  policy_arn =
"arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
}

resource "aws_iam_instance_profile" "ec2_ssm_role" {
  name = "ec2_ssm_role"
  role = aws_iam_role.ec2_ssm_role.name
}

resource "aws_network_interface" "private" {
  subnet_id = module.vpc.private_subnets[0]
  tags = {
    Name = "ec2 Public"
  }
}

resource "aws_instance" "private" {
```

```
  ami           = data.aws_ami.amazon_linux2.id
  instance_type = "t2.micro"

  network_interface {
    network_interface_id = aws_network_interface.private.id
    device_index         = 0
  }
  iam_instance_profile = var.deploy_task_resources ?
aws_iam_instance_profile.ec2_ssm_role.id : null

  tags = merge(local.default_tags, {
    Name = "private"
  })
}

resource "aws_network_interface" "public" {
  subnet_id = module.vpc.public_subnets[0]
  tags = {
    Name = "ec2 Public"
  }
}

resource "aws_instance" "bastion" {
  ami           = data.aws_ami.amazon_linux2.id
  instance_type = "t2.micro"

  network_interface {
    network_interface_id = aws_network_interface.public.id
    device_index         = 0
  }
  iam_instance_profile = aws_iam_instance_profile.ec2_ssm_role.id

  tags = merge(local.default_tags, {
    Name = "bastion"
  })
}

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"
  }
}

data "aws_availability_zones" "available" {
  state = "available"
}

module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.77.0"

  name = "tf_bastion_vpc"

  cidr = "10.0.0.0/16"

  azs             = data.aws_availability_zones.available.names
  private_subnets = ["10.0.1.0/24"]
  public_subnets  = ["10.0.101.0/24"]

  enable_ipv6 = true

  enable_nat_gateway = false
  single_nat_gateway = true

  public_subnet_tags = {
    Name = "tf_bastion_vpc_public"
  }

  private_subnet_tags = {
    Name = "tf_bastion_vpc_private"
  }

  tags = {
    Owner       = "E-mentor"
  }

  vpc_tags = {
    Name = "tf_bastion_vpc"
```

```
  }
}

output "vpc_nat_ids" {
  value = module.vpc.natgw_ids
}

terraform {

  backend "s3" {
    bucket = "cloud-mentor-bv-tf-states"
    key    = "bastion/terraform.tfstate"
    region = "us-east-1"
  }

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


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
  }
}
```

ДОДАТОК Г

Слайди презентації

Харківський національний
університет радіоелектроніки

Кафедра інформаційно-мережної інженерії

Кваліфікаційна робота магістра на тему:

"Дослідження шляхів автоматизації перевірки цілісності хмарної інфраструктури"

Виконав: студент групи ІМІзм-19-2

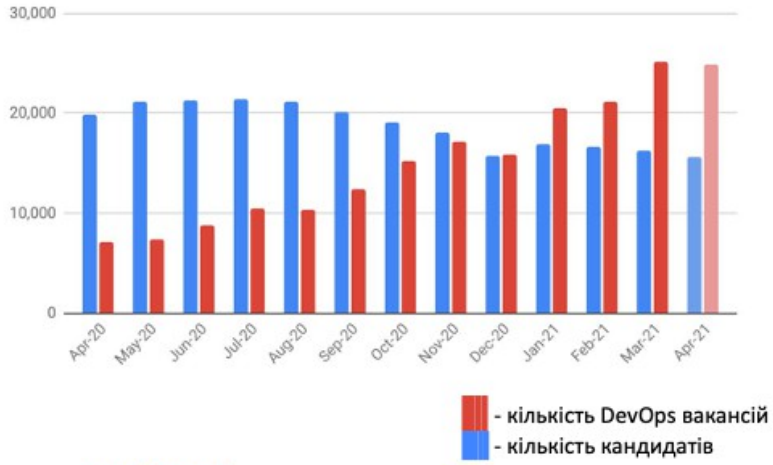
Батов Вадим Володимирович

Науковий керівник: доц. Костромицький А. І.

Мета роботи

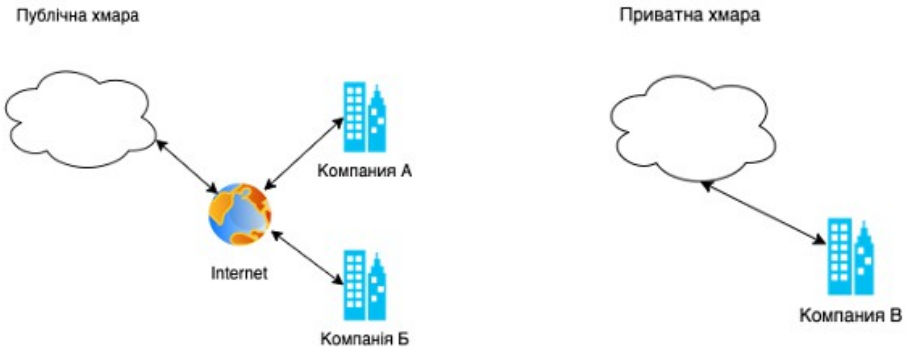
- Розробити алгоритм перевірки цілісності хмарної інфраструктури.
- Проаналізувати можливості використання подібної методики для автоматизації перевірки завдань, навчання та підготовки спеціалістів в області хмарних технологій

Аналіз ринку України за спеціальністю DevOps



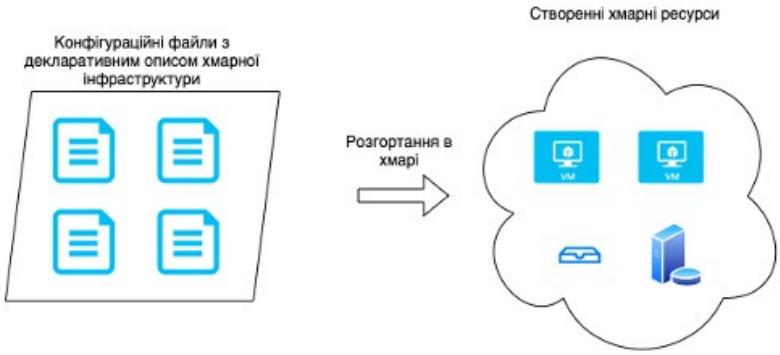
За даними ресурсу https://djinni.co/

Огляд видів хмар



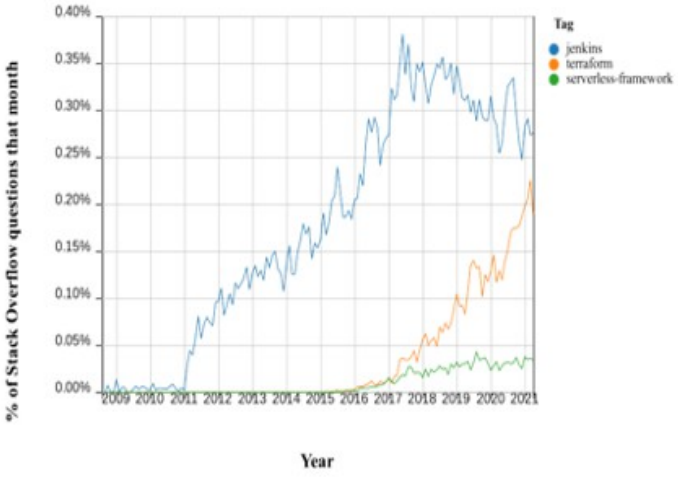
Огляд хмарних провайдерів

Поняття інфраструктура як код

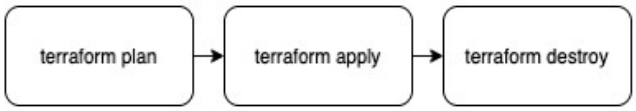


Безперервна доставка

Огляд популярності інструментів



Робочий процес з Terraform



Структура прототипу для автоматизації навчання

Висновки

- В рамках роботи було розглянуто хмарні технології, їх структуру та деталі реалізації.
- Проаналізовано дослідження компанії Garthner, щодо лідерів на ринку, на основі чого було обрано хмарного провайдера AWS для виконання практичної частини.
- Дано визначення поняттям гнучких методологій та безперервної доставки. Проведено огляд інструментів для виконання роботи та вибрано такі інструменти як Jenkins та Terraform для реалізації бізнес логіки системи.
- Розроблено алгоритм для перевірки цілісності інфраструктури. На основі якого створено прототип системи для автоматизації навчання та підготовки спеціалістів в області хмарних технологій з використанням продуктів з відкритим програмним кодом Docker, Jenkins та Terraform.

Позначення	Найменування	Дод. відомості
	Текстові документи	
	Пояснювальна записка	55 с.
	Графічні документи	
	Слайди презентації	11 шт.

Змін.	Арк.	Номер докум.	Підпис	Дата	Дослідження шляхів автоматизації перевірки цілісності хмарної інфраструктури	Літ.	Аркуш	Аркушів
Розроб.		Батов				У	1	1
Перевір.		Костромицький						
Н. Контр.		Золотарьов			Відомість кваліфікаційної роботи	ХНУРЕ кафедра ІМІ		
Затв.		Безрук						