ДОДАТОК А

Лістинг програмного коду Terraform&Terragrunt

EC2 Configuration

```
resource "aws_launch_configuration" "launch_configuration" {
  name           = "${var.developer}-launch_configuration"
  image_id       = var.image_id
  instance_type  = var.instance_type
  spot_price     = var.spot_price
  security_groups = [aws_security_group.ec2_sg.id]
  user_data      = <<EOF
                   #!/bin/bash
                   echo
ECS_CLUSTER=${data.terraform_remote_state.ecs_cluster.outputs.ecs_cluster_name} >>
/etc/ecs/ecs.config
                   EOF
  lifecycle {
    create_before_destroy = true
  }

  depends_on = [
    aws_security_group.ec2_sg,
    data.terraform_remote_state.ecs_cluster,
  ]
}


resource "aws_autoscaling_group" "autoscaling_group" {
  name                      = "${var.developer}-asg"
  max_size                  = var.autoscaling_group_max_size
  min_size                  = var.autoscaling_group_min_size
  health_check_grace_period = var.health_check_grace_period
  health_check_type         = var.health_check_type
  desired_capacity          = var.autoscaling_group_desired_capacity
  force_delete              = true
  launch_configuration      = aws_launch_configuration.launch_configuration.name
  vpc_zone_identifier       = aws_subnet.private_subnet.*.id

  depends_on = [
    aws_launch_configuration.launch_configuration
  ]
}


resource "aws_instance" "simple_server" {
  name          = "${var.developer}-simple-server"
  ami           = var.image_id
  instance_type = var.instance_type
}
```

AWS Cognito

```
resource "aws_cognito_user_pool" "user_pool" {
  name = "${var.developer}-users"

  username_attributes      = ["phone_number"]
  auto_verified_attributes = ["phone_number"]

  password_policy {
    minimum_length                   = 8
    require_symbols                  = true
    require_numbers                  = true
    require_uppercase                = true
    require_lowercase                = true
    temporary_password_validity_days = 1
  }

  username_configuration {
    case_sensitive = false
  }

  account_recovery_setting {
    recovery_mechanism {
      name     = "verified_phone_number"
      priority = 1
    }
  }

  admin_create_user_config {
    allow_admin_create_user_only = false
  }

  device_configuration {
    challenge_required_on_new_device      = true
    device_only_remembered_on_user_prompt = false
  }

  sms_configuration {
    external_id    = "diploma_external_id"
    sns_caller_arn = aws_iam_role.sns_caller.arn
  }

  verification_message_template {
    sms_message = "Hello, {username}! Your verification code is {####}."
  }

  user_pool_add_ons {
    advanced_security_mode = "AUDIT"
  }
}

resource "aws_cognito_user_pool_client" "user_pool_client" {
```

```
  generate_secret               = false
  name                          = "${var.developer}-user-pool-client"
  user_pool_id                  = aws_cognito_user_pool.user_pool.id
  access_token_validity         = var.tokens_validity
  refresh_token_validity        = 365
  id_token_validity             = var.tokens_validity
  prevent_user_existence_errors = "LEGACY"
  enable_token_revocation       = true
  callback_urls                 = [var.default_url]
  default_redirect_uri          = var.default_url
  explicit_auth_flows           = [
    "ALLOW_ADMIN_USER_PASSWORD_AUTH",
    "ALLOW_CUSTOM_AUTH",
    "ALLOW_REFRESH_TOKEN_AUTH",
    "ALLOW_USER_PASSWORD_AUTH",
    "ALLOW_USER_SRP_AUTH"
  ]

  token_validity_units {
    access_token  = "minutes"
    id_token      = "minutes"
    refresh_token = "minutes"
  }
}
```

AWS IAM

```
resource "aws_iam_role" "sns_caller" {
  name = "${var.developer}-sns-caller-role"

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

resource "aws_iam_role_policy" "sns_caller" {
  name = "${var.project_name}-sns-caller-policy"
  role = aws_iam_role.sns_caller.id
```

```
  policy = <<EOF
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": ["sns:publish"],
         "Resource": ["*"]
      }
   ]
}
EOF
}

resource "aws_iam_role" "codepipeline_role" {
  name = "${var.developer}-codepipeline-role"

  assume_role_policy = jsonencode({
    "Version" : "2012-10-17",
    "Statement" : [
      {
        "Effect" : "Allow",
        "Action" : "sts:AssumeRole",
        "Principal" : {
          "Service" : "codepipeline.amazonaws.com"
        },
      },
      {
        "Effect" : "Allow",
        "Action" : "s3:*",
        "Resource" : aws_s3_bucket.codepipeline_bucket.arn
      },
      {
        "Effect" : "Allow",
        "Action" : "codebuild:*",
        "Resource" : aws_codebuild_project.codebuild.arn # ?
      },
      {
        "Effect" : "Allow",
        "Action" : "codedeploy:*",
        "Resource" : "*",
      },
      {
        "Action" : [
          "ecr:*",
          "ecs:*"
        ],
        "Resource" : [
          aws_ecr_repository.ecr_repository.arn,
          data.terraform_remote_state.ecs_cluster.outputs.ecs_cluster_arn
        ],
```

```
      "Effect" : "Allow"
    },
   ]
  })
}

resource "aws_iam_role" "codebuild" {
  name = "${var.developer}-codebuild"

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

resource "aws_iam_role" "codedeploy" {
  name = "${var.developer}-codedeploy"

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


resource "aws_iam_role_policy_attachment" "AWSCodeDeployRole" {
  policy_arn = "arn:aws:iam::aws:policy/service-role/AWSCodeDeployRole"
  role       = aws_iam_role.codedeploy.name
}


resource "aws_iam_role" "ecs_task" {
  name = "${var.developer}-ecs-task"

  assume_role_policy = jsonencode({
    "Version" : "2012-10-17",
```

```
    "Statement" : [
      {
        "Action" : "sts:AssumeRole",
        "Principal" : {
          "Service" : "ecs-tasks.amazonaws.com"
        },
        "Effect" : "Allow"
      },
      {
        "Effect" : "Allow",
        "Action" : [
          "ecr:*",
          "logs:*"
        ],
        "Resource" : "*"
      },
      {
        "Effect" : "Allow",
        "Action" : [
          "secretsmanager:GetSecretValue"
        ],
        "Resource" : [
          data.aws_secretsmanager_secret.database_name.arn,
          data.aws_secretsmanager_secret.database_host.arn,
          data.aws_secretsmanager_secret.database_port.arn,
          data.aws_secretsmanager_secret.database_password.arn,
          data.aws_secretsmanager_secret.database_username.arn,

          data.aws_secretsmanager_secret.cognito_user_pool.arn,
          data.aws_secretsmanager_secret.cognito_client_ids.arn,

          data.aws_secretsmanager_secret.app_secret_key.arn

        ]
      }
    ]
  })
}

resource "aws_iam_role" "cloudwatch_event_pipeline_iam_role" {
  name = "${var.project_name}-cloudwatch-event-pipeline"

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

```
    ]
  })
}


resource "aws_iam_role_policy" "cloudwatch_event_pipeline_iam_role_policy" {
  name = "${var.project_name}-cw-event-pipeline-policy"
  role = aws_iam_role.cloudwatch_event_pipeline_iam_role.name

  policy = jsonencode({
    "Version" : "2012-10-17",
    "Statement" : [
      {
        "Effect" : "Allow",
        "Action" : [
          "codepipeline:StartPipelineExecution"
        ],
        "Resource" : [
          aws_codepipeline.deploy_codepipeline.arn
        ]
      }
    ]
  })
}

resource "aws_iam_role" "ecs_instance_role" {
  name               = "${var.developer}-ecs-instance"
  path               = "/"
  assume_role_policy = data.aws_iam_policy_document.ecs_instance_policy.json
}


data "aws_iam_policy_document" "ecs_instance_policy" {
  statement {
    actions = ["sts:AssumeRole"]

    principals {
      type        = "Service"
      identifiers = ["ec2.amazonaws.com"]
    }
  }
}

resource "aws_iam_role_policy_attachment" "ecs_instance_role_attachment" {
  role       = aws_iam_role.ecs_instance_role.name
  policy_arn = "arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role"

  depends_on = [
    aws_iam_role.ecs_instance_role
  ]
}
```

AWS Load Balancer

```
resource "aws_lb" "load_balancer" {
  name            = "${var.developer}-lb"
  security_groups = [aws_security_group.load_balancer_sg.id]
  subnets         = aws_subnet.public_subnet.*.id
  depends_on      = [aws_security_group.load_balancer_sg]

  tags = {
    Environment = "${var.developer}-development"
  }
}

resource "aws_lb_listener" "http_listener" {
  load_balancer_arn = aws_lb.load_balancer.arn
  port              = "80"
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.instance_lb_tg.arn
  }
}

resource "aws_lb_listener" "https_listener" {
  load_balancer_arn = aws_lb.load_balancer.arn
  port              = "443"
  protocol          = "HTTPS"
  certificate_arn   = "data.terraform_remote_state.certificates.outputs.certificate_arn"
  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.instance_lb_tg.arn
 }
}

resource "aws_lb_target_group" "instance_lb_tg" {
  name     = "${var.developer}-lb-tg"
  port     = 5000
  protocol = "HTTP"
  vpc_id   = aws_vpc.main.id

  health_check {
    healthy_threshold   = "5"
    unhealthy_threshold = "2"
    interval            = "30"
    matcher             = "200"
    path                = var.alb_health_check_url
    port                = "traffic-port"
    protocol            = "HTTP"
    timeout             = "5"
  }
}
```

```
resource "aws_lb_target_group" "http_target_group" {
  name     = "${var.developer}-lb-tg"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.main.id

  health_check {
    healthy_threshold   = 5
    unhealthy_threshold = 2
    interval            = 30
    matcher             = 200
    path                = var.alb_health_check_url
    port                = "traffic-port"
    protocol            = "HTTP"
    timeout             = 5
  }
}
```

AWS Secret Manager

```
resource "aws_secretsmanager_secret" "database_host" {
  name                    = "${var.developer}-db-host"
  description             = "Database host"
  recovery_window_in_days = 0
}

resource "aws_secretsmanager_secret" "database_port" {
  name                    = "${var.developer}-db-port"
  description             = "Database port"
  recovery_window_in_days = 0
}

resource "aws_secretsmanager_secret" "database_username" {
  name                    = "${var.developer}-db-username"
  description             = "Database username"
  recovery_window_in_days = 0
}

resource "aws_secretsmanager_secret" "database_password" {
  name                    = "${var.developer}-db-password"
  description             = "Database password"
  recovery_window_in_days = 0
}

resource "aws_secretsmanager_secret" "database_name" {
  name                    = "${var.developer}-db-name"
  description             = "Database name"
  recovery_window_in_days = 0
}

resource "aws_secretsmanager_secret" "cognito_user_pool" {
```

```
  name                    = "${var.developer}-cognito-user-pool"
  description             = "Cognito user pool"
  recovery_window_in_days = 0
}

resource "aws_secretsmanager_secret" "cognito_client_ids" {
  name                    = "${var.developer}-cognito-client-ids"
  description             = "Client IDS for available applications"
  recovery_window_in_days = 0
}

resource "aws_secretsmanager_secret" "app_secret_key" {
  name                    = "${var.developer}-application-secret-key"
  description             = "Secret key for Django app"
  recovery_window_in_days = 0
}
```

AWS VPC & NAT

```
resource "aws_vpc" "main" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = "true"
  tags                 = {
    Name = "Default vpc. Developer - ${var.developer}."
  }
}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "${var.developer}-igw"
  }
}

resource "aws_eip" "nat_eip" {
  vpc        = true
  depends_on = [aws_internet_gateway.igw]
}

resource "aws_nat_gateway" "nat" {
  count         = length(aws_subnet.public_subnet.*.id)
  allocation_id = aws_eip.nat_eip.id
  subnet_id     = element(aws_subnet.public_subnet.*.id, count.index)
  depends_on    = [aws_internet_gateway.igw]

  tags = {
    Name = "nat"
  }
}
```

```
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id

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

resource "aws_main_route_table_association" "public" {
  vpc_id         = aws_vpc.main.id
  route_table_id = aws_route_table.public.id
}

resource "aws_route_table" "private" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block     = "0.0.0.0/0"
    nat_gateway_id = aws_nat_gateway.nat.*.id[count.index]
  }

  count = length(aws_subnet.private_subnet)
}

resource "aws_route_table_association" "private" {
  subnet_id      = aws_subnet.private_subnet.*.id[count.index]
  route_table_id = aws_route_table.private.*.id[count.index]
  count          = length(aws_subnet.private_subnet)
}
```