```
provider "aws" {

  region = "us-east-1"

}


resource "aws_vpc" "test_vpc" {

  cidr_block           = "10.0.0.0/16"

  enable_dns_support   = true

  enable_dns_hostnames = true

}


resource "aws_subnet" "test_subnet" {

  count                   = 2

  vpc_id                  = aws_vpc.test_vpc.id

  cidr_block              = "10.0.1.${count.index * 64}/26"

  map_public_ip_on_launch = true

  availability_zone       = element(["us-east-1a", "us-east-1b"], count.index)

}


resource "aws_internet_gateway" "test_igw" {

  vpc_id = aws_vpc.test_vpc.id

}


resource "aws_route_table" "test_route_table" {

  vpc_id = aws_vpc.test_vpc.id


  route {
    cidr_block = "0.0.0.0/0"

    gateway_id = aws_internet_gateway.test_igw.id

  }

}


resource "aws_route_table_association" "test_rta" {

  count          = length(aws_subnet.test_subnet.*.id)

  subnet_id      = element(aws_subnet.test_subnet.*.id, count.index)

  route_table_id = aws_route_table.test_route_table.id

}


resource "aws_security_group" "test_sg" {

  name        = "test-sg"

  description = "Security group for testing with all ports open"

  vpc_id      = aws_vpc.test_vpc.id


  ingress {

    from_port   = 0

    to_port     = 0

    protocol    = "-1"

    cidr_blocks = ["0.0.0.0/0"]

  }


  egress {

    from_port   = 0

    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}


resource "aws_ecs_cluster" "test_cluster" {
  name = "test-cluster"
}


resource "aws_ecs_service" "test_service" {
  name            = "test-service"
  cluster         = aws_ecs_cluster.test_cluster.id
  task_definition = aws_ecs_task_definition.app_task.arn
  desired_count   = 1
  launch_type     = "FARGATE"

  network_configuration {
    assign_public_ip = true
    subnets         = aws_subnet.test_subnet.*.id
    security_groups = [aws_security_group.test_sg.id]
  }
}



# Add your ECS Task Definition and Service here, using the aws_ecs_task_definition and
```

aws_ecs_service resources.

```
resource "aws_iam_role" "ecs_task_execution_role" {
  name = "ecs_task_execution_role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
      {
        Action = "sts:AssumeRole",
        Effect = "Allow",
        Principal = {
          Service = "ecs-tasks.amazonaws.com",
        },
        Sid = "",
      },
    ],
  })
}
resource "aws_iam_policy" "ecr_read_policy" {
  name        = "ecr_read_policy"
  path        = "/"
  description = "IAM policy for reading from ECR"

  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
      {
        Action = [
          "ecr:GetDownloadUrlForLayer",
          "ecr:BatchGetImage",
          "ecr:BatchCheckLayerAvailability",
        ],
        Effect   = "Allow",
        Resource = "*",
      },
    ],
  })
}
resource "aws_iam_policy" "ecr_policy" {
  name        = "ECRPolicy"
  path        = "/"
  description = "Allow ECS tasks to pull images from ECR"

  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
      {
        Effect = "Allow",
        Action = "ecr:GetAuthorizationToken",
        Resource = "*"
      },
      {
        Effect = "Allow",
        Action = [
          "ecr:BatchCheckLayerAvailability",
          "ecr:GetDownloadUrlForLayer",
          "ecr:BatchGetImage"
        ],
        Resource = "arn:aws:ecr:REGION-GOES-HERE:USER-ACCOUNT-ID:repository/helloworld"
      }
    ]
  })
}
resource "aws_iam_policy_attachment" "ecr_policy_attach" {
  name       = "ECRPolicyAttachment"
  roles      = [aws_iam_role.ecs_task_execution_role.name]
  policy_arn = aws_iam_policy.ecr_policy.arn
}
resource "aws_iam_role_policy_attachment" "ecs_task_execution_role_policy_attach" {
  role       = aws_iam_role.ecs_task_execution_role.name
  policy_arn = aws_iam_policy.ecr_read_policy.arn
}
# Note: This script sets up the VPC, subnets, and security group. Ensure your ECS Task Definition
and Service configurations align with this setup.
resource "aws_ecs_task_definition" "app_task" {
  family                   = "helloworld"
  network_mode             = "awsvpc"
  requires_compatibilities = ["FARGATE"]
  },
  execution_role_arn       = aws_iam_role.ecs_task_execution_role.arn
  task_role_arn            = aws_iam_role.ecs_task_execution_role.arn
  cpu                      = "4096" # Minimum vCPU for FARGATE
  memory                   = "16384" # Minimum memory for FARGATE

  ephemeral_storage {
    size_in_gib = 70
  }
  container_definitions = jsonencode([
    {
      name      = "helloworld-container"
                                          image                                    =
"USER-ACCOUNT-ID.dkr.ecr.REGION-GOES-HERE.amazonaws.com/helloworld:latest"
      cpu       = 2048
      memory    = 8192
      essential = true
      portMappings = [
        {
          containerPort = 80
          hostPort      = 80
          protocol      = "tcp"
        },
      ]
    },
  ])
}
```