Main.tf

provider "aws" {

    region = var.region

}

resource "aws\_vpc" "uda\_test" {

    cidr\_block = "80.0.0.0/16"

    tags = {

        Name = "uda\_test"

    }

}

resource "aws\_subnet" "uda\_subnet\_pub\_test"{

    vpc\_id = "${aws\_vpc.uda\_test.id}"

    cidr\_block = "80.0.1.0/24"

    tags = {

        Name = "Uda\_subnet"

    }

}

resource "aws\_security\_group" "uda\_secgrp" {

  name        = "allow\_tls"

  description = "Allow TLS inbound traffic"

  vpc\_id      = "${aws\_vpc.uda\_test.id}"

  ingress {

    description = "TLS from VPC"

    from\_port   = 443

    to\_port     = 443

    protocol    = "tcp"

    cidr\_blocks = [aws\_vpc.uda\_test.cidr\_block]

  }

  egress {

    from\_port   = 0

    to\_port     = 0

    protocol    = "-1"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  tags = {

    Name = "uda\_test\_sec\_grp"

  }

}

resource "aws\_iam\_role" "iam\_for\_lambda" {

  name = "iam\_for\_lambda"

  assume\_role\_policy = <<EOF

{

  "Version": "2012-10-17",

  "Statement": [

    {

      "Action": "sts:AssumeRole",

      "Principal": {

        "Service": "lambda.amazonaws.com"

      },

      "Effect": "Allow",

      "Sid": ""

    }

  ]

}

EOF

  tags = {

    Name = "IAM for Lambda"

  }

}

resource "aws\_iam\_policy" "policy" {

  name        = "Lambda-VPC-policy"

  description = "Lambda policy"

  policy = <<EOF

{

  "Version": "2012-10-17",

  "Statement": [

    {

             "Effect": "Allow",

             "Action": [

                 "logs:CreateLogGroup",

                 "logs:CreateLogStream",

                 "logs:PutLogEvents",

                 "ec2:CreateNetworkInterface",

                 "ec2:DescribeNetworkInterfaces",

                 "ec2:DeleteNetworkInterface"

             ],

             "Resource": "\*"

         }

  ]

}

EOF

}

resource "aws\_iam\_role\_policy\_attachment" "test-attach" {

  role       = "${aws\_iam\_role.iam\_for\_lambda.name}"

  policy\_arn = "${aws\_iam\_policy.policy.arn}"

}

 data "archive\_file" "lambda\_zip" {

    type          = "zip"

    source\_file   = "greet\_lambda.py"

    output\_path   = "lambda\_function.zip"

}

resource "aws\_lambda\_function" "test\_lambda" {

  filename      = "lambda\_function.zip"

  function\_name = "Udacity\_lambda"

  role          = "${aws\_iam\_role.iam\_for\_lambda.arn}"

  handler       = "greet\_lambda.lambda\_handler"

  runtime = "python3.7"

  vpc\_config {

    # Every subnet should be able to reach an EFS mount target in the same Availability Zone. Cross-AZ mounts are not permitted.

    subnet\_ids         = ["${aws\_subnet.uda\_subnet\_pub\_test.id}"]

    security\_group\_ids = ["${aws\_security\_group.uda\_secgrp.id}"]

  }

  environment {

    variables = {

      foo = "bar"

    }

  }

}

Output.tf

# TODO: Define the output variable for the lambda function.

output "function\_arn" {

  description = "The ARN of the Lambda function"

  value       = aws\_lambda\_function.test\_lambda.arn

}

output "function\_invoke\_arn" {

  description = "The Invoke ARN of the Lambda function"

  value       = aws\_lambda\_function.test\_lambda.invoke\_arn

}

output "function\_name" {

  description = "The name of the Lambda function"

  value       = aws\_lambda\_function.test\_lambda.function\_name

}

output "function\_qualified\_arn" {

  description = "The qualified ARN of the Lambda function"

  value       = aws\_lambda\_function.test\_lambda.qualified\_arn

}

Variables.tf

# TODO: Define the variable for aws\_region

variable "region" {

    default = "us-east-1"

}

Greet\_Lambda.py

import os

def lambda\_handler(event, context):

    return "Hello from Lambda: Udacity"