S3

Hello! And welcome back to the Terraform and Ansible for AWS course on Linux Academy! My name is derek, and in this lesson, we’re going to create the resources required for the S3 storage portion of our infrastructure.

Before we get started on our resources, we need to add a variable for our domain name. This will allow us to name the S3 resources and avoid conflicts with other resources since S3 resources must be globally unique. Here is my variables.tf file which has domain\_name added, and here is my terraform.tfvars file in which I have defined the variable as linuxsuperhero. I didn’t add the .com because there are other areas in this script where we will use this variable, so it’s easiest just to leave it off and not have to define too many variables.

S3 requires more resources than other components of our infrastructure. S3 will require the buckets and the ability for EC2 instances to access these buckets. This will require a profile, a role policy, and a role located in the IAM section of our script. It will also require an “endpoint” to ensure the private instances can access the bucket, this will be located in the VPC section of our script. Finally, we will need to create the bucket itself, which we will locate in the compute area of our script.

Let’s start with the proper IAM role profile, role policy, and role required for our EC2 instances to access.

resource "aws\_iam\_instance\_profile" "s3\_access" {

name = "s3\_access"

roles = ["${aws\_iam\_role.s3\_access.name}"]

}

resource "aws\_iam\_role\_policy" "s3\_access\_policy" {

name = "s3\_access\_policy"

role = "${aws\_iam\_role.s3\_access.id}"

policy = <<EOF

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "s3:\*",

"Resource": "\*"

}

]

}

EOF

}

resource "aws\_iam\_role" "s3\_access" {

name = "s3\_access"

assume\_role\_policy = <<EOF

{

"Version": "2012-10-17",

"Statement": [

{

"Action": "sts:AssumeRole",

"Principal": {

"Service": "ec2.amazonaws.com"

},

"Effect": "Allow",

"Sid": ""

}

]

}

EOF

}

Due to the complexity of this, I have just copied and pasted the required information. I suggest you do the same and edit accordingly. Amazon can be very picky about spacing, so it’s easiest just to copy/paste when you can. Sample policies can be find in the AWS docs as well as the Terraform docs, I have included links to those to assist with creating S3 policies if you wish to customize the ones provided here.

As you can see, I have created the iam instance profile first, the iam role policy second, and the iam role third. You may not remember having this many steps when creating a role policy in the console. This is because the instance profile is actually created automatically from within AWS when you create a new instance role. I have attached an AWS document to this course which explains this further.

<http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2_instance-profiles.html>

Now we will scroll down to the VPC section of our script to create the S3 VPC endpoint. This will allow routing to and from S3 to the private instances without having to have an S3 IP addresses added. This code is as follows:

#create S3 VPC endpoint

resource "aws\_vpc\_endpoint" "private-s3" {

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

service\_name = "com.amazonaws.${var.aws\_region}.s3"

route\_table\_ids = ["${aws\_vpc.vpc.main\_route\_table\_id}", "${aws\_route\_table.public.id}"]

policy = <<POLICY

{

"Statement": [

{

"Action": "\*",

"Effect": "Allow",

"Resource": "\*",

"Principal": "\*"

}

]

}

POLICY

}

The only part here that may seem a little strange is the service\_name. In an S3 endpoint, the service\_name is the S3 and region backwards. This effectively is s3.us-east-1.amazonaws.com. As you can see, we used our region variable to populate this. I have then allowed all resources with permission to access the S3 buckets.

Now we can finally create our S3 bucket. In this example, we are only going to create a code bucket, but if you want to extend this infrastructure and add cloudfront, you will also want to create a media bucket as well.

This is the example of the bucket:

Resource “aws\_s3\_bucket” “code” {

bucket = “${var.domain\_name}\_code1115”

acl = “private”

force\_destroy = true

tags {

Name = “code bucket”

}

}

We have created the bucket, I used the variable domain\_name that we created earlier, to name the bucket. I then added a number to the end just to be absolutely sure it’s not duplicated, but if your domain name is unique enough, you shouldn’t have an issue. I then gave it a private ACL to ensure it’s not publicly accessible, set force\_destroy to true to allow the ability to destroy it if it has items in it (this is up to you, I keep it this way for testing, but for production, you may want to set that to false.), and tagged it with a name of media bucket.

Great! We have created our S3 bucket and required resources to work with our ec2 instances. Join me in the next lesson as we continue our infrastructure!