VPC and subnets

1. In this lesson, we are going to build our VPC, the subnets, and the security groups required for our infrastructure to run.
2. First up, we need to create the VPC

Resource “aws\_vpc” “vpc” {

Cidr\_block = “10.1.0.0/16”

}

* 1. This will create a subnet with a slash 16, which means the last 2 “octets” 10 and 1 are static and the last 2 0s will change. Our subnets will further divide this range into /24 subnets. This means we will assign a different third octet for each subnet, leaving a /24, or about 252 addresses, for our resources to use. If you foresee using more than 252 addresses within any subnet, such as by having a max ASG instance count of 300 ASG hosts in an availability zone, you may want to redefine this scheme.

1. Next, we create the internet gateway. IF you have watched the LinuxAcademy AWS courses, you should understand internet gateways very well.

Resource “aws\_internet\_gateway” “internet\_gateway” {

Vpc\_id = “${aws\_vpc.vpc.id}”

}

1. So what did I just do there? That doesn’t look like a VPC ID! Well, since we don’t know the VPC ID as it hasn’t been created yet, we reference it using what’s called “interpolation syntax”. Interpolation syntax is how we define variables based on several criteria, including their name or ID. In this case, we are referencing the aws\_vpc resource with an “ID” of “vpc”. The “ID” of any resource is the name of the resource specified in quotations after the resource type. We are going to use this extensively, so be sure you understand this concept!
2. Now we are going to create the Route tables.
   1. First we will add the public route table:

resource “aws\_route\_table” “public” {

vpc\_id = “${aws\_vpc.vpc.id}”

route {

cidr\_block = “0.0.0.0/0”

gateway\_id = “${aws\_internet\_gateway.internet-\_gateway.id}

}

tags {

Name = “public”

}

}

* 1. Ok, let’s go through this: first we use the aws\_route\_table resource and we give it an id of “public”. Then we assign this route table to the VPC\_ID as we did with the internet gateway before. We then add a route to the open internet…0.0.0.0/0 using the gateway id referenced above. After that, we tag it with a name of public. This isn’t required, but it can be useful, especially as your environment grows. You may even want to add an environment tag and define production, development, etc.
  2. Now we are going to create our private route table. Due to Terraform being a little picky about requiring internet gateways with route tables, it’s actually easiest just to use the default route table here rather than creating a new one.

resource “aws\_default\_route\_table” “private” {

default\_route\_table\_id = “${aws\_vpc.vpc.default\_route\_table\_id}”

tags {

Name = “private”

}

}

* 1. As you can see, we used an interpolated value using the “default\_route\_table\_id”. This syntax is a little different and it is set by AWS. This is one of those things about terraform that you just kinda have to go with. There are certainly several quirks in Terraform that may require a little searching to solve.