Creating a Multi-Region Network with VPC Peering Using SGs, IGW, and RTs

00:37:55 Exit Lab ✓ Complete Lab

45 minutes duration Professional Rate this lab

Videos Guide

Creating a Multi-Region Network with VPC Peering Using SGs, IGW, and RTs

Introduction

It can get cumbersome trying to track all the different routing components of a network, especially in the fast-moving, dynamic IT operations world today. By maintaining your AWS resources such as VPC, SGs, and IGWs using Terraform, you can track all of the changes as code.

In this lab, students will go through creating a network setup complete with VPCs, subnets, security groups, internet gateways, and VPC peering in AWS using Terraform. Students are expected to have working knowledge of VPC resources and basic network components within AWS to be able to take full advantage of this lab.

Solution

Log in to the Terraform Controller Node EC2 Instance

1. Find the details for logging in to the Terraform Controller node provided by the hands-on lab interface and log in to the node using SSH:

```
ssh cloud_user@<IP-OF-TERRAFORM-CONTROLLER>
```

> **Note**: This instance already has an EC2 instance profile (role) attached to it and has all necessary AWS API permissions required for this lab. It also has the AWS CLI set up and configured with the AWS account attached to this lab, for which the console login credentials are also provided in the lab interface page once the lab spins up.

2. After logging in, verify the version of Terraform installed (should be 12.29). Execute the following command to check:

```
terraform version
```

Clone the GitHub Repo for Terraform Code

Use the `git` command to clone the GitHub repo which has the Terraform code for deploying the solution of this lab. GitHub repo URL.

1. Execute the following command:

```
git clone https://github.com/linuxacademy/content-deploying-to-aws-ansible-terraform.git
```

2. Change to the directory for lab Terraform code:

```
cd content-deploying-to-aws-ansible-terraform/lab_network_vpc_peering
```

3. Examine the contents of the directory you're in:

```
ls
```

Deploy the Terraform Code

1. Initialize the Terraform directory you changed into to download the required provider

```
terraform init
```

2. Ensure Terraform code is formatted properly:

```
terraform fmt
```

3. Ensure code has proper syntax and no errors:

```
terraform validate
```

4. See the execution plan and note the number of resources that will be created:

```
terraform plan
```

Enter `yes` when prompted.

5. Deploy resources:

   ```
   terraform apply
   ```

   Enter `yes` when prompted.

   After `terraform apply` has run successfully, you can either use the AWS CLI on the Controller node to list and describe created resources or you can log in to the AWS Console to verify and investigate created resources.

6. Finally, on the Terraform Controller node CLI, delete all resources which were created and ensure that it runs through successfully.

   ```
   terraform destroy
   ```

Conclusion

Congratulations — you've completed this hands-on lab!

Tools

Lab Diagram Instant Terminal

Credentials

How do I connect?

AWS Account

Username

cloud_user

Password

G$MV[c2wTL1b]0*niMz2

Open Link in Incognito Window

Cloud Server PublicIP

Username

cloud_user

Password

tPr(J8qG

PublicIP

34.229.18.73

Launch Instant Terminal

Additional Resources

You work for Pinehead Terraformers, a company which specializes in deploying networking layouts on AWS using Terraform. You're a Solutions Architect on part of a project for a start-up whereby you need to deploy a multi-region network for the start-up. The requirement is that the customer (start-up) wants to deploy EC2 instances in more than one region and wants EC2 instances in different VPCs to be able to communicate securely. The customer has no idea how to do this via AWS and so have left it up to you to show them a sample layout.

You already have a plan in mind. You're thinking along the following lines:

1. Deploy a VPC each in `us-east-1` and `us-west-2`. (This is a requirement too)
2. The `us-east-1` VPC will have CIDR range of `10.0.0.0/16`.
3. The `us-west-2` VPC will have CIDR range of `192.168.0.0/16`.
4. There will be two subnets in the `us-east-1` VPC with the following CIDR ranges: `10.0.1.0/24` and `10.0.2.0/24`.
5. There will be one subnet in the `us-west-2` VPC with the following CIDR range: `192.168.1.0/24`.
6. You'll deploy a VPC peering connection between both VPCs. (You'll initiate the peering connection from `us-east-1`).

7. You'll be creating/modifying VPC route tables accordingly so that traffic can flow between the two VPCs.
8. Create and attach internet gateways to both VPCs.
9. Create security groups which allow incoming traffic on TCP ports 22 and 8080. You'll also need to ensure security groups in each region allow traffic on all ports from the IP range of the peered VPC or at least its subnets.

Good luck!

Learning Objectives

0 of 3 completed

- [ ] Log in to the Terraform Controller Node EC2 Instance
- [ ] Clone the GitHub Repo for Terraform Code
- [ ] Deploy the Terraform Code