Implementing Cross-Cloud Vpc Peering With Terraform For Inter-Cloud Communication Between Google Cloud And AWS

[1]Mr.V.Gopinath M.E, [2]Kamalesh S, [3]Keerthirajan M

[1]Assistant Professor , [2,3]Students

Department of Computer Science and Engineering, KSR Institute for Engineering and Technology, Tiruchengode

Abstract:In this project it shows how to use Terraform by HashiCorp to create secure, private, site-to-site connections between Google Cloud and Amazon Web Services (AWS) using virtual private networks (VPNs). This is a multi-cloud deployment.In this project, we will deploy virtual machine (VM) instances into custom virtual private cloud (VPC) networks in Google Cloud and AWS then deploy supporting infrastructure to construct a VPN connection with two Internet Protocol security (IPsec) tunnels between the Google Cloud and AWS VPC networks. The environment and tunnel deployment usually completes within four minutes. This project is based off of the Automated Network Deployment.To establish a secure and scalable VPN connection between Google Cloud and AWS using Terraform infrastructure as code. By leveraging the power of both cloud providers, the resulting network architecture will allow for seamless data transfer and enhanced workload distribution, while maintaining robust security measures.

IndexTerms:VPC Peering,Cross cloud,Google cloud,AWS,Terraform.

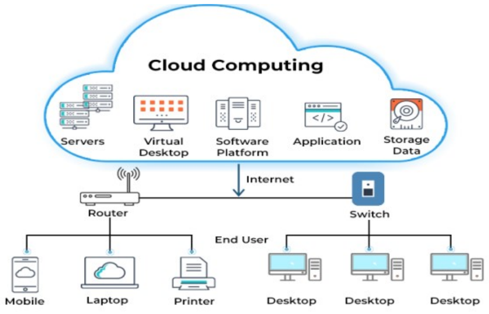
I. INTRODUCTION

This project aims to demonstrate the implementation of a Virtual Private Network (VPN) between two major cloud service providers, Google Cloud and Amazon Web Services (AWS), using Terraform. The project focuses on creating a secure and encrypted connection betweentwo networks to allow for the exchange of data and communication without exposing it to the public internet. The solution leverages the VPN gateway feature of both cloud providers and creates a site-to-site connection between the two networks. The implementation uses Terraform as the infrastructure as code tool to automate the deployment and configuration of the required resources on both cloud platforms. The projectprovides a detailed guide on how to set up the VPN connection and configure the necessary network configurations, as well as the Terraform scripts used for the implementation. Overall, this project serves as a useful reference for anyone looking to establish a VPN connection between Google Cloud and AWS using Terraform.

DOMAIN USED

Cloud computing

cloud computing pall is the on- demand vacuity of computing coffers as services over the internet. It eliminates the want for enterprises to land, configure, or take coffers themselves, and they only pay for what they exercise. pall computing uses a network(most frequently, the internet) to connect druggies to a pall platform where they requisition and pierce leased calculating services. A intermediary garçon handles all the message between customer bias and waiters to grease the trade of data. screen and sequestration features are common or garden factors to keep this information secure and safe.



.Building a VPN between Google Cloud and AWS with Terraform can involve several challenges, including:
Network configuration: Configuring the virtual private network (VPN) connection between two different cloud providers requires understanding the networking requirements and configuration options of both providers. Google Cloud and AWS use different networking technologies, which can complicate the process of creating a VPN connection.
Security: When building a VPN between cloud providers, it's important to ensure that the connection is secure and protected from unauthorized access. This can involve configuring authentication, encryption, and access controls on both ends of the connection.
Compatibility: Terraform is a tool for infrastructure as code (IAC) that allows you to manage cloud resources in a consistent and repeatable way. However, different cloud providers have different APIs and resource types, which can make it challenging to create a unified infrastructure configuration that works seamlessly across multiple cloud providers.
Cost: Building a VPN between Google Cloud and AWS may involve additional costs, such as data transfer fees and network egress charges. It's important to understand the pricing models of both cloud providers and factor in these costs when planning your VPN configuration.
Complexity: Building a VPN between cloud providers is a complex task that requires expertise in networking, security, and cloud infrastructure. It can be challenging to troubleshoot issues that arise during the configuration process, and it may require a significant amount of time and resources to get everything working properly.

OVERVIEW OF THE EXISTING APPROACHES:

There are several modules available on the Terraform registry that can help you set up a VPN connection between Google Cloud and AWS.One such module is the "terraform-aws-vpn" module, which can be used to create a site-to-site VPN connection between AWS and Google Cloud. This module creates an AWS Customer Gateway and a Virtual Private Gateway, and then establishes the VPN connection between the two.
Another module is the "terraform-google-vpn-gateway" module, which can be used to create a VPN gateway in Google Cloud. This module creates a Cloud VPN gateway and a Cloud Router, and then establishes the VPN connection with AWS. To use these modules, you will need to provide some configuration parameters, such as the IP addresses and authentication details for the VPN connection. Once you have configured these parameters, Terraform will automatically create the necessary infrastructure resources and establish the VPN connection between Google Cloud and AWS. Overall, using Terraform modules is a convenient and efficient way to set up a VPN connection between Google Cloud and AWS. These modules have already been tested and proven to work, so you can be confident that your VPN connection will be reliable and secure.

PROPOSED SYSTEM

1. Set up the required infrastructure in Google Cloud and AWS:
- Create a VPC in both Google Cloud and AWS.
- Create subnets in each VPC.
- Create an internet gateway in AWS and a NAT gateway in Google Cloud.
- Create virtual machines (VMs) in each subnet.

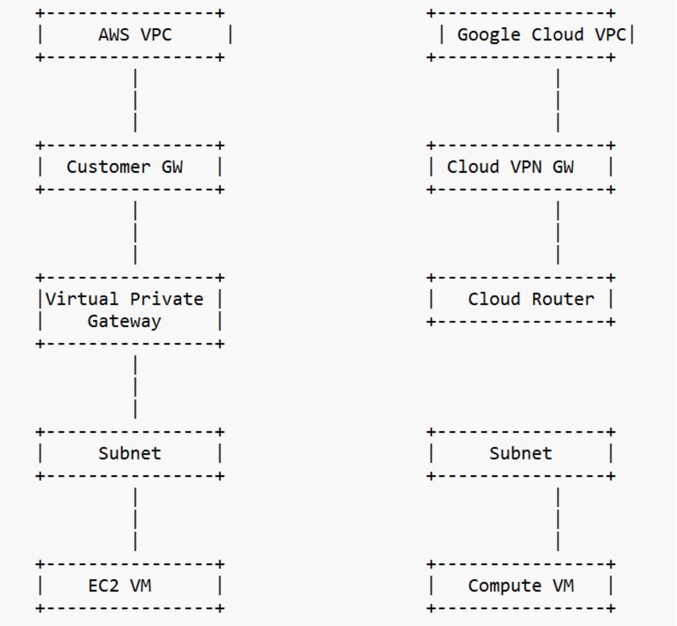
2. Create a virtual private network (VPN) connection in each cloud provider:
- In AWS, create a customer gateway and a virtual private gateway. Then, create a VPN connection that connects the customer gateway to the virtual private gateway.
- In Google Cloud, create a Cloud VPN gateway and configure a Cloud Router to connect the VPN gateway to the VPC network.

3. Configure the VPN connection:
- In AWS, configure the VPN connection with the public IP of the Google Cloud VPN gateway, shared secret, and the routing information.
- In Google Cloud, configure the VPN connection with the public IP of the AWS virtual private gateway, shared secret, and routing information.

4. Use Terraform to automate the infrastructure deployment and configuration:
- Define the required infrastructure resources in Terraform.
- Use Terraform to deploy and configure the infrastructure.

5. Test the VPN connection:
- Verify that the VPN connection is established between the two cloud providers.
- Test the connectivity between the VMs in each cloud provider.

FLOW DIAGRAM

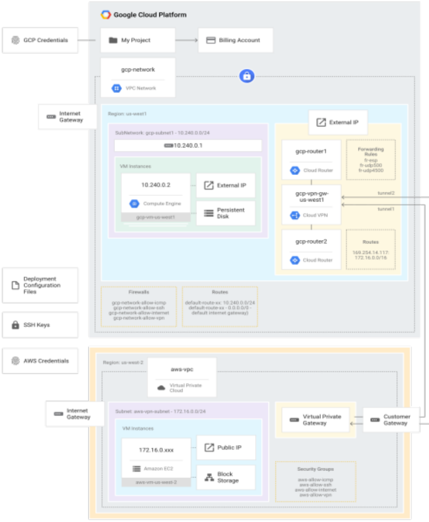


```
+----------------+          +----------------+
|    AWS VPC      |          | Google Cloud VPC|
+----------------+          +----------------+
        |                           |
        |                           |
        |                           |
+----------------+          +----------------+
|  Customer GW   |          | Cloud VPN GW   |
+----------------+          +----------------+
        |                           |
        |                           |
        |                           |
+----------------+          +----------------+
|Virtual Private |          |  Cloud Router  |
|    Gateway     |          +----------------+
+----------------+
        |
        |
        |
+----------------+          +----------------+
|     Subnet     |          |     Subnet     |
+----------------+          +----------------+
        |                           |
        |                           |
        |                           |
+----------------+          +----------------+
|    EC2 VM      |          |  Compute VM    |
+----------------+          +----------------+
```

EXPLAINATION:

In this diagram, there are two VPCs, one in AWS and one in Google Cloud. The VPCs contain subnets where virtual machines (VMs) are deployed. A VPN connection is established between the two VPCs using a customer gateway in AWS and a Cloud VPN gateway in Google Cloud. The VPN connection is configured to use a virtual private gateway in AWS and a Cloud Router in Google Cloud. The VMs in each subnet can then communicate securely over the VPN connection. Terraform is used to define and deploy the infrastructure resources required for the VPN connection, including the VPCs, subnets, VMs, customer gateway, virtual private gateway, Cloud VPN gateway, and Cloud Router. Terraform also configures the VPN connection with the required settings, such as encryption and authentication protocols, routing information, and shared secrets.

DEPLOYMENT ARCHITECTURE



Infrastructure as Code: Using Terraform, the infrastructure can be defined and managed as code, which allows for repeatable and consistent deployment of the infrastructure. Changes can be made to the infrastructure through code, which is then applied through Terraform, ensuring consistency across deployments.

Security: Security is a key consideration when building a VPN between Google Cloud and AWS. This includes configuring the VPN connection to use encryption and authentication protocols, as well as securing the infrastructure resources that are used to create the VPN connection.

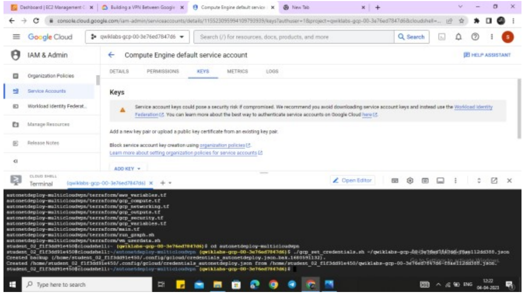
Scalability: The infrastructure should be designed with scalability in mind, so that it can easily grow or shrink as needed. This includes the ability to add or remove resources, as well as the ability to handle increased traffic and load. Resilience: The infrastructure should be designed to be resilient to failures, so that it can continue to operate even if individual components fail. This includes using redundancy and failover mechanisms to ensure high availability of the VPN connection.

Automation: Using Terraform to automate the deployment and configuration of the infrastructure can help to ensure consistency and reduce the risk of human error.

By following these principles, we can build a VPN between Google Cloud and AWS with Terraform that is secure, scalable, resilient, and automated.

RESULTS AND DISCUSSION

Download Compute Engine



Keys

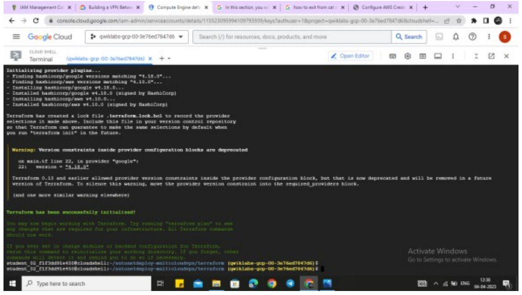
Service account keys could pose a security risk if compromised. We recommend you avoid downloading service account keys and instead use the Workload Identity Federation. You can learn more about the best way to authenticate service accounts on Google Cloud here.

Add a new key pair or upload a public key certificate from an existing key pair.

Block service account key creation using organization policies.
Learn more about setting organization policies for service accounts

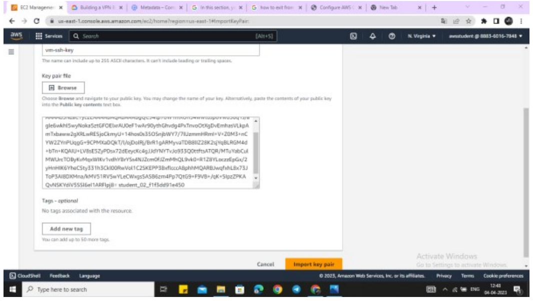
```
autonetdeploy-multicloudvpn/terraform/aws_variables.tf
autonetdeploy-multicloudvpn/terraform/gcp_compute.tf
autonetdeploy-multicloudvpn/terraform/gcp_networking.tf
autonetdeploy-multicloudvpn/terraform/gcp_outputs.tf
autonetdeploy-multicloudvpn/terraform/gcp_security.tf
autonetdeploy-multicloudvpn/terraform/gcp_variables.tf
autonetdeploy-multicloudvpn/terraform/main.tf
autonetdeploy-multicloudvpn/terraform/run_graph.sh
autonetdeploy-multicloudvpn/terraform/vm_userdata.sh
student_02_f1f3dd91e450@cloudshell:~ (qwiklabs-gcp-00-3e76ed7847d6)$ cd autonetdeploy-multicloudvpn
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-00-3e76ed7847d6)$ ./gcp_set_credentials.sh ~/qwiklabs-gcp-00-3e76ed7847d6-f0ae112dd388.json
Created backup (/home/student_02_f1f3dd91e450/.config/gcloud/credentials_autonetdeploy.json.bak.1680591132).
Created /home/student_02_f1f3dd91e450/.config/gcloud/credentials_autonetdeploy.json from /home/student_02_f1f3dd91e450/qwiklabs-gcp-00-3e76ed7847d6-f0ae112dd388.json.
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-00-3e76ed7847d6)$
```

Create AWS access credentials



```
Initializing provider plugins...
- Finding hashicorp/google versions matching "4.18.0"...
- Finding hashicorp/aws versions matching "4.10.0"...
- Installing hashicorp/google v4.18.0...
- Installed hashicorp/google v4.18.0 (signed by HashiCorp)
- Installing hashicorp/aws v4.10.0...
- Installed hashicorp/aws v4.10.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

│ Warning: Version constraints inside provider configuration blocks are deprecated
│
│   on main.tf line 22, in provider "google":
│   22:   version = "4.18.0"
│
│ Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration block, but that is now deprecated and will be removed in a future
│ version of Terraform. To silence this warning, move the provider version constraint into the required_providers block.
│
│ (and one more similar warning elsewhere)

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$
```

Creating key pair in aws



vm-ssh-key

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair file

Choose Browse and navigate to your public key. You may change the name of your key. Alternatively, paste the contents of your public key into the Public key contents text box.

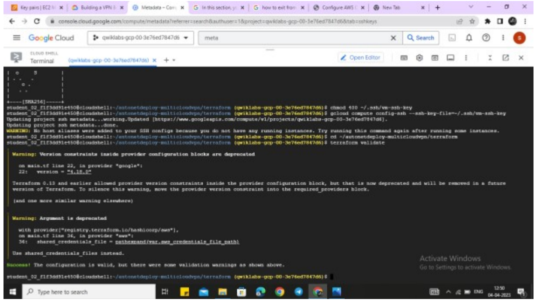
```
gle6wkhI5wyNoka5ztGFOElxrAU0eF1wAr90ythGhvdg4PxTnvoOtXgDvEmhasVLkpA
mTxbaww2gXRLwRESjoCkmyU+14hosOs35O5njbWY7/7IJzmmHRml+V+Z0M3+nC
YW2ZYnPUqgG+9CPMXaDQkT/l/ojDolRj/BrR1gARMyvaTD88IIZ28K2sjYqBLRGM4d
+bTn+KQAIU+LV8sE5ZyPDsx72dEeycKc4gJJdYNYTvJo933Q0ttftsATQR/MTuYabCul
MWUrcTOByKvMqxWIKv1vdhYBrY5s4NJZcm0fJZmMhQL9vk0+R1Z8YLoczzEpGx/2
yHnHIK6YheCSty331h3Ckl00RwVol1C2SKEPP3BxflcccA8phhMQARBJwqfxhL8x73J
ToP3AI8DXMna/kMV51RV5wYLeCWxgsSASB6zm4Pp7QtG9+F9VB+/qK+5IpzZPKA
QvN5KYdIV555I6el1ARFIpjB= student_02_f1f3dd91e450
```

Tags - optional

No tags associated with the resource.

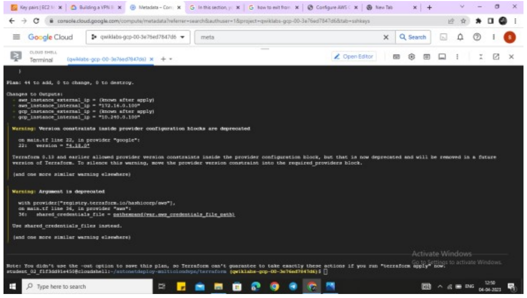
You can add up to 50 more tags.

Terraform validate



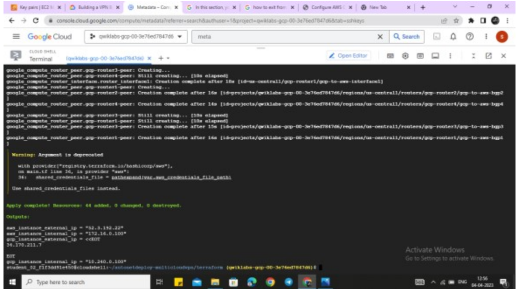
```
| o    S        |
| . .   .       |
|    o .        |
|   . .         |
|               |
+----[SHA256]-----+
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$ chmod 400 ~/.ssh/vm-ssh-key
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$ gcloud compute config-ssh --ssh-key-file=~/.ssh/vm-ssh-key
Updating project ssh metadata...working.Updated [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-00-3e76ed7847d6].
Updating project ssh metadata...done.
WARNING: No host aliases were added to your SSH configs because you do not have any running instances. Try running this command again after running some instances.
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$ cd ~/autonetdeploy-multicloudvpn/terraform
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$ terraform validate

 Warning: Version constraints inside provider configuration blocks are deprecated

   on main.tf line 22, in provider "google":
   22:    version = "4.18.0"

 Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration block, but that is now deprecated and will be removed in a future
 version of Terraform. To silence this warning, move the provider version constraint into the required_providers block.

 (and one more similar warning elsewhere)


 Warning: Argument is deprecated

   with provider["registry.terraform.io/hashicorp/aws"],
   on main.tf line 36, in provider "aws":
   36:    shared_credentials_file = pathexpand(var.aws_credentials_file_path)

 Use shared_credentials_files instead.

Success! The configuration is valid, but there were some validation warnings as shown above.

student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$
```

Terraform plan



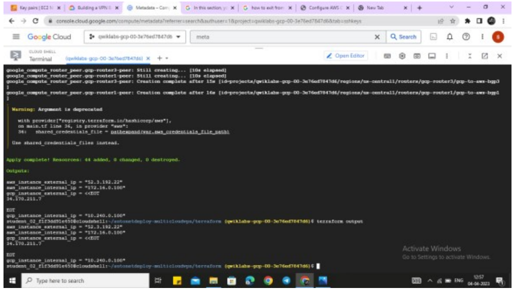
```
    }

Plan: 44 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + aws_instance_external_ip = (known after apply)
  + aws_instance_internal_ip = "172.16.0.100"
  + gcp_instance_external_ip = (known after apply)
  + gcp_instance_internal_ip = "10.240.0.100"

 Warning: Version constraints inside provider configuration blocks are deprecated

   on main.tf line 22, in provider "google":
   22:    version = "4.18.0"

 Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration block, but that is now deprecated and will be removed in a future
 version of Terraform. To silence this warning, move the provider version constraint into the required_providers block.

 (and one more similar warning elsewhere)


 Warning: Argument is deprecated

   with provider["registry.terraform.io/hashicorp/aws"],
   on main.tf line 36, in provider "aws":
   36:    shared_credentials_file = pathexpand(var.aws_credentials_file_path)

 Use shared_credentials_files instead.

 (and one more similar warning elsewhere)


Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$
```

Terraform apply



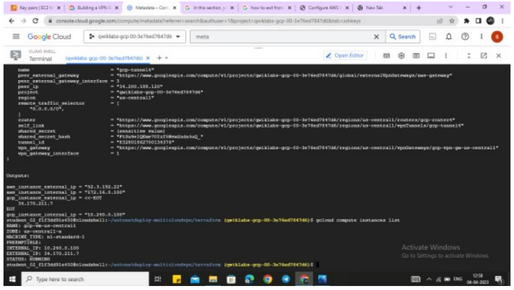
```
google_compute_router_peer.gcp-router3-peer: Creating...
google_compute_router_peer.gcp-router4-peer: Still creating... [10s elapsed]
google_compute_router_interface.router_interface1: Creation complete after 18s [id=us-central1/gcp-router1/gcp-to-aws-interface1]
google_compute_router_peer.gcp-router1-peer: Creating...
google_compute_router_peer.gcp-router2-peer: Creation complete after 16s [id=projects/qwiklabs-gcp-00-3e76ed7847d6/regions/us-central1/routers/gcp-router2/gcp-to-aws-bgp2
]
google_compute_router_peer.gcp-router4-peer: Creation complete after 14s [id=projects/qwiklabs-gcp-00-3e76ed7847d6/regions/us-central1/routers/gcp-router4/gcp-to-aws-bgp4
]
google_compute_router_peer.gcp-router3-peer: Still creating... [10s elapsed]
google_compute_router_peer.gcp-router1-peer: Still creating... [10s elapsed]
google_compute_router_peer.gcp-router3-peer: Creation complete after 15s [id=projects/qwiklabs-gcp-00-3e76ed7847d6/regions/us-central1/routers/gcp-router3/gcp-to-aws-bgp3
]
google_compute_router_peer.gcp-router1-peer: Creation complete after 16s [id=projects/qwiklabs-gcp-00-3e76ed7847d6/regions/us-central1/routers/gcp-router1/gcp-to-aws-bgp1
]

 Warning: Argument is deprecated

   with provider["registry.terraform.io/hashicorp/aws"],
   on main.tf line 36, in provider "aws":
   36:    shared_credentials_file = pathexpand(var.aws_credentials_file_path)

 Use shared_credentials_files instead.


Apply complete! Resources: 44 added, 0 changed, 0 destroyed.

Outputs:

aws_instance_external_ip = "52.3.192.22"
aws_instance_internal_ip = "172.16.0.100"
gcp_instance_external_ip = <<EOT
34.170.211.7

EOT
gcp_instance_internal_ip = "10.240.0.100"
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$
```

Terraform output



```
google_compute_router_peer.gcp-router3-peer: Still creating... [10s elapsed]
google_compute_router_peer.gcp-router1-peer: Still creating... [10s elapsed]
google_compute_router_peer.gcp-router3-peer: Creation complete after 15s [id=projects/qwiklabs-gcp-00-3e76ed7847d6/regions/us-central1/routers/gcp-router3/gcp-to-aws-bgp3
]
google_compute_router_peer.gcp-router1-peer: Creation complete after 16s [id=projects/qwiklabs-gcp-00-3e76ed7847d6/regions/us-central1/routers/gcp-router1/gcp-to-aws-bgp1
]

 Warning: Argument is deprecated

   with provider["registry.terraform.io/hashicorp/aws"],
   on main.tf line 36, in provider "aws":
   36:    shared_credentials_file = pathexpand(var.aws_credentials_file_path)

 Use shared_credentials_files instead.


Apply complete! Resources: 44 added, 0 changed, 0 destroyed.

Outputs:

aws_instance_external_ip = "52.3.192.22"
aws_instance_internal_ip = "172.16.0.100"
gcp_instance_external_ip = <<EOT
34.170.211.7

EOT
gcp_instance_internal_ip = "10.240.0.100"
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$ terraform output
aws_instance_external_ip = "52.3.192.22"
aws_instance_internal_ip = "172.16.0.100"
gcp_instance_external_ip = <<EOT
34.170.211.7

EOT
gcp_instance_internal_ip = "10.240.0.100"
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$
```

gcloud compose instance list



```
    name                           = "gcp-tunnel4"
    peer_external_gateway          = "https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-00-3e76ed7847d6/global/externalVpnGateways/aws-gateway"
    peer_external_gateway_interface = 3
    peer_ip                        = "34.200.188.120"
    project                        = "qwiklabs-gcp-00-3e76ed7847d6"
    region                         = "us-central1"
    remote_traffic_selector        = [
        "0.0.0.0/0",
    ]
    router                         = "https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-00-3e76ed7847d6/regions/us-central1/routers/gcp-router4"
    self_link                      = "https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-00-3e76ed7847d6/regions/us-central1/vpnTunnels/gcp-tunnel4"
    shared_secret                  = (sensitive value)
    shared_secret_hash             = "FtSu9e1QKme7OIzf5NewZnda5uQ_"
    tunnel_id                      = "832801862700136376"
    vpn_gateway                    = "https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-00-3e76ed7847d6/regions/us-central1/vpnGateways/gcp-vpn-gw-us-central1"
    vpn_gateway_interface          = 1
}


Outputs:

aws_instance_external_ip = "52.3.192.22"
aws_instance_internal_ip = "172.16.0.100"
gcp_instance_external_ip = <<-EOT
    34.170.211.7
EOT
gcp_instance_internal_ip = "10.240.0.100"
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$ gcloud compute instances list
NAME: gcp-vm-us-central1
ZONE: us-central1-a
MACHINE_TYPE: n1-standard-1
PREEMPTIBLE:
INTERNAL_IP: 10.240.0.100
EXTERNAL_IP: 34.170.211.7
STATUS: RUNNING
student_02_f1f3dd91e450@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-00-3e76ed7847d6)$
```

CONCLUSION

In conclusion, building a VPN between Google Cloud and AWS with Terraform is a powerful way to establish a secure, private connection between two of the most popular cloud providers. By using Terraform to automate the infrastructure deployment and configuration, organizations can ensure consistency and repeatability of the infrastructure setup, reduce the risk of human error, and make it easier to manage and scale the infrastructure as needed. The principles of infrastructure as code, security, scalability, resilience, and automation are key to building a VPN that is secure, scalable, resilient, and automated. The future scope for Building a VPN between Google Cloud and AWS with Terraform is promising, with potential developments in integration with other cloud providers, enhanced security features, simplified deployment and management, and performance optimization. Overall, building a VPN between Google Cloud and AWS with Terraform is a powerful tool for organizations that want to adopt multi-cloud or hybrid cloud strategies and need a secure and efficient way to communicate between cloud providers.

REFERENCES

[1] Chen-Hsiang Chen; Yu-An Lin; Wei-Te Wu; Yao-Te Huang; Cha-Chin Chu : Design and Implementation of IPv4 and IPv6 Provisioning Technologies for VPC Architecture

[2] Liang-Yu Shyu; Ying-Hsuan Wu; W. Hu:Using wavelet transform and fuzzy neural network for VPC detection from the holter ECG

[3] Ying-Hsuan Wu; Liang-Yu Shyu : Detection of VPC using wavelet transform and fuzzy neural network

[4] Misuk Huh; DaeYoub Kim; Eunah Kim; Byoung-Joon Lee : Secure virtual personal cloud service based on CCN/VPC

[5] Jose Carrasco; Javier Cubo; Francisco Durán; Ernesto PimentelBidimensional Cross-Cloud Management with TOSCA and Brooklyn

[6] Application Migration Architecture for Cross Clouds Analysis on the Strategies Methods and Frameworks

[7] Philippe Abdoulaye : Digitizing the Business Model Using AWS

[8] Philippe Abdoulaye : Developing World-Class Digital Products and Services Using AWS

[9] R.Christy Pushpaleela; S. Sankar; K. Viswanathan; S. Aathithya Kumar : Application Modernization Strategies for AWS Cloud

[10] Huibin Yin;Jun Han;Jing Liu;Jing Dong : The application research of GAE on E-learning — Taking Google CloudCourse for example

[11] Huibin Yin;Jun Han;Jing Liu;Xu Hongyun : Development and research of multimedia courseware sharing platform based on GAE

[12] Tarun Goyal;Ajit Singh;Aakanksha Agrawal : Cloudtarun Application deployed over GAE and android emulator

[13] Manu Gupta;Mandepudi Nobel Chowdary;Sankeerth Bussa;Chennupati Kumar Chowdary :Deploying Hadoop Architecture Using Ansible and terraform

[14] Leonardo Rebouças de Carvalho;Aleteia Patricia Favacho de Araujo : Performance Comparison of Terraform and Cloudify as Multicloud Orchestrators

[15] Jagdish Chandra Patni;Souradeep Banerjee;Devyanshi Tiwari : Infrastructure as a Code (IaC) to Software Defined Infrastructure using Azure Resource Manager (ARM)