K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch:____2____ Roll No.:__1911032______

Experiment No. 8

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Building a VPN Between Google Cloud and AWS with Terraform

Objective: To building a VPN Between Google Cloud and AWS with Terraform

Expected Outcome of Experiment:

CO	Outcome
CO3	Develop cloud applications using Aneka platform

Books/ Journals/ Websites referred:

Abstract:-

This lab will show you 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 lab, you will deploy virtual machine (VM) instances into custom virtual private cloud (VPC) networks in Google Cloud and AWS. You 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 lab is based off of the Automated Network Deployment tutorial.

Related Theory: -

Automation of Cloud Infrastructure is a norm that every company follows. Whenever we think of Cloud platforms, the first thing that comes to our mind is what IAAC (Infrastructure as a Code) tooling to be used. Well all the Cloud platforms out there comes up with their own toolset, like -

AWS — Cloudformation

Azure — ARM Templates

GCP — Deployment Manager

Apart from these one can use Chef, Ansible or Puppet as well to achieve the same functionalities along with simply using Cloud SDK. However, most organizations out there prefer to choose Terraform as their Cloud Automation service. The major advantage of using Terraform is, it’s totally free, comes with a huge community support, provides support for all major cloud providers along with many other things.

Terraform is the infrastructure as code offering from HashiCorp. It is a tool for building, changing, and managing infrastructure in a safe, repeatable way. SRE and DevOps teams can use Terraform to manage environments with a configuration language called the HashiCorp Configuration Language (HCL) for human-readable, automated deployments.

Terraform manages your cloud infrastructure by maintaining a state file. This state file consists of the actual state of your resources at a given point of time. This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures.

By default Terraform stores state in a local file named “terraform.tfstate”, but it can also be stored remotely, which works better in a team environment.

In an organization it’s always recommended to store Terraform State remotely. In this article I’ll show you how to store your state on AWS — S3, Azure — ADLS Gen2 and GCP — Cloud Storage.

Terraform uses this state to create plans and make changes to your infrastructure. Prior to any operation, Terraform does a refresh to update the state with the real infrastructure. Terraform will showcase you the desired state when you run “terraform plan” by comparing it with the actual state which is present in your state file.

Firstly, setup AWS CLI and configure it to access your AWS account locally. We need AWS Access Keys and Secrets to access our AWS account which in further would be used by Terraform as well to generate “terraform plan” and apply changes.

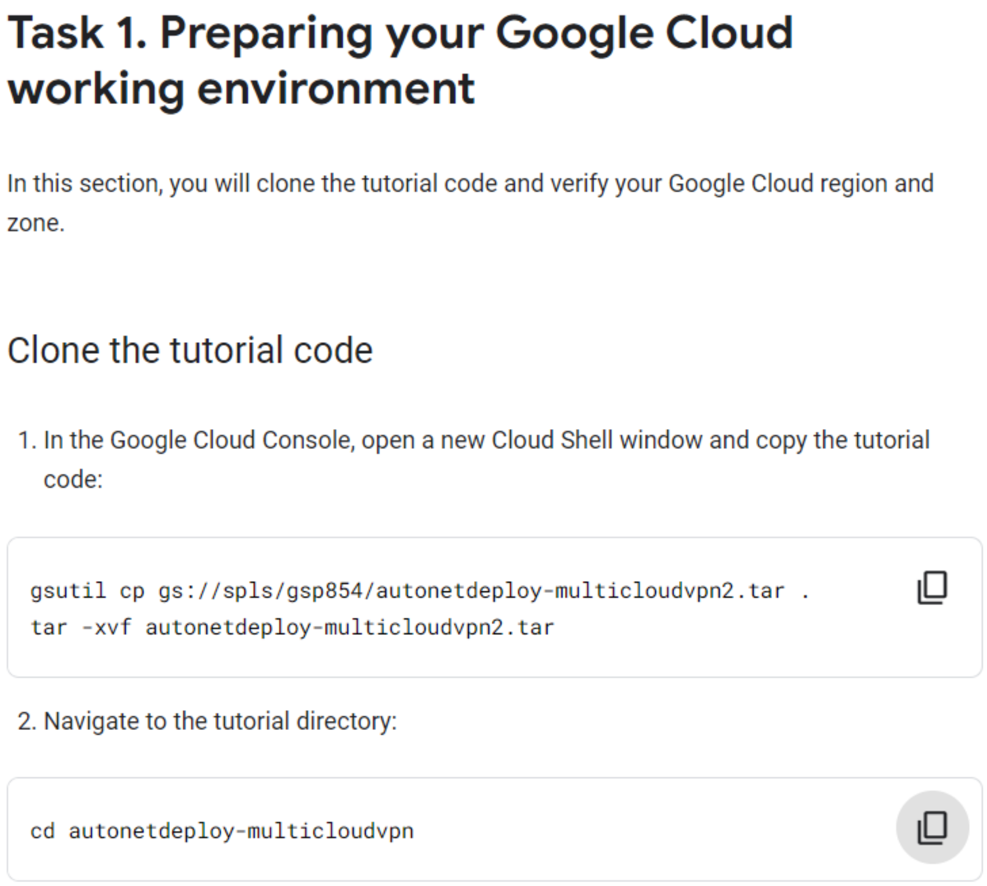
Install and Configure AWS CLI — AWS CLI

Now, we need a S3 bucket. We use S3 as our Backend to store Terraform State Files. Once our S3 bucket is created, we can start using Terraform to create AWS resources. I’ll showcase how to create an IAM User, Group, Custom Policies and S3 Bucket using my modules.

Terraform GitHub Repo — Github Repo

I prefer creating Modules for my resources, so that they’re repeatable and can be used going forward to spin up the same resources again. Terraform Modules work similar to Functions, which are repeatable in nature and come with a modular structure.

Implementation Details:



Task 1. Preparing your Google Cloud working environment

In this section, you will clone the tutorial code and verify your Google Cloud region and zone.

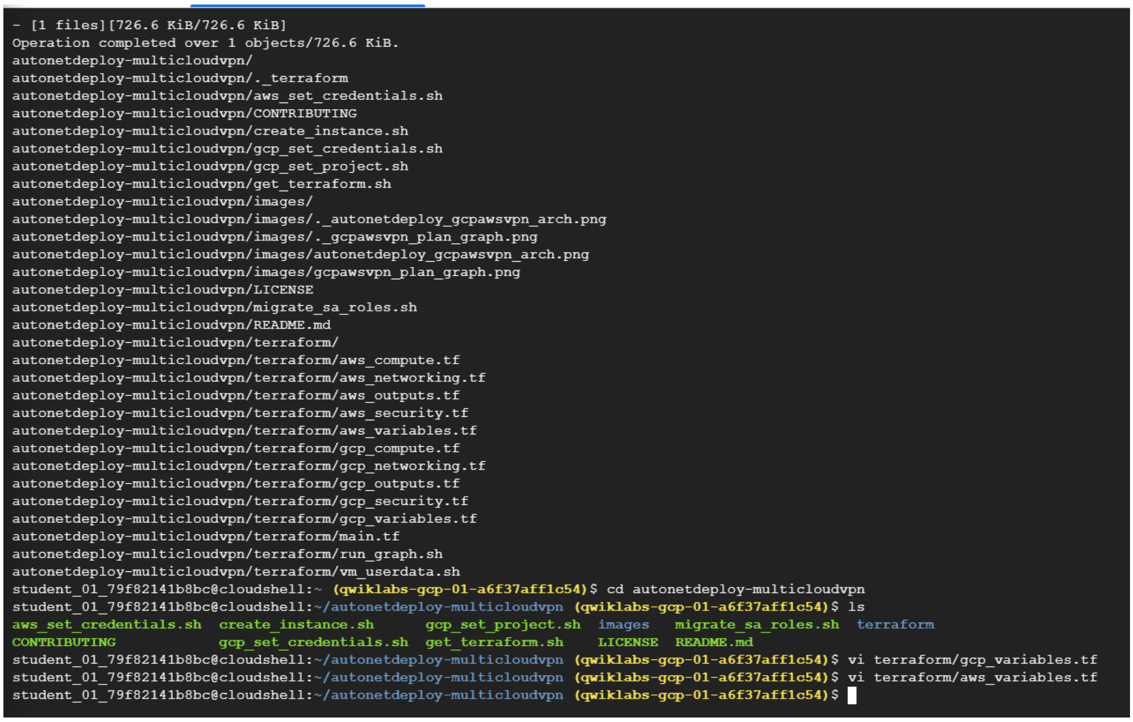
Clone the tutorial code

1. In the Google Cloud Console, open a new Cloud Shell window and copy the tutorial code:

```
gsutil cp gs://spls/gsp854/autonetdeploy-multicloudvpn2.tar .
tar -xvf autonetdeploy-multicloudvpn2.tar
```

2. Navigate to the tutorial directory:

```
cd autonetdeploy-multicloudvpn
```



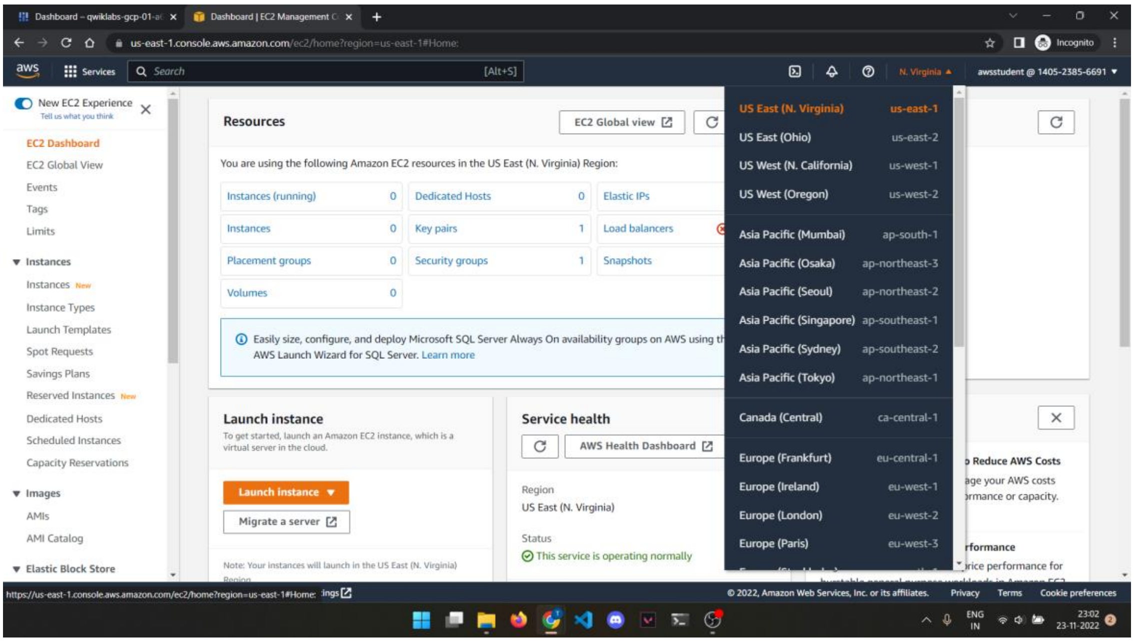
```
- [1 files][726.6 KiB/726.6 KiB]
Operation completed over 1 objects/726.6 KiB.
autonetdeploy-multicloudvpn/
autonetdeploy-multicloudvpn/._terraform
autonetdeploy-multicloudvpn/aws_set_credentials.sh
autonetdeploy-multicloudvpn/CONTRIBUTING
autonetdeploy-multicloudvpn/create_instance.sh
autonetdeploy-multicloudvpn/gcp_set_credentials.sh
autonetdeploy-multicloudvpn/gcp_set_project.sh
autonetdeploy-multicloudvpn/get_terraform.sh
autonetdeploy-multicloudvpn/images/
autonetdeploy-multicloudvpn/images/._autonetdeploy_gcpawsvpn_arch.png
autonetdeploy-multicloudvpn/images/._gcpawsvpn_plan_graph.png
autonetdeploy-multicloudvpn/images/autonetdeploy_gcpawsvpn_arch.png
autonetdeploy-multicloudvpn/images/gcpawsvpn_plan_graph.png
autonetdeploy-multicloudvpn/LICENSE
autonetdeploy-multicloudvpn/migrate_sa_roles.sh
autonetdeploy-multicloudvpn/README.md
autonetdeploy-multicloudvpn/terraform/
autonetdeploy-multicloudvpn/terraform/aws_compute.tf
autonetdeploy-multicloudvpn/terraform/aws_networking.tf
autonetdeploy-multicloudvpn/terraform/aws_outputs.tf
autonetdeploy-multicloudvpn/terraform/aws_security.tf
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_01_79f82141b8bc@cloudshell:~ (qwiklabs-gcp-01-a6f37aff1c54)$ cd autonetdeploy-multicloudvpn
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ ls
aws_set_credentials.sh  create_instance.sh      gcp_set_project.sh  images   migrate_sa_roles.sh  terraform
CONTRIBUTING            gcp_set_credentials.sh  get_terraform.sh    LICENSE  README.md
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ vi terraform/gcp_variables.tf
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ vi terraform/aws_variables.tf
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$
```

Task 2. Preparing for AWS use

In this section, you will verify your AWS region. For details about AWS regions, refer to Regions and Availability Zones for AWS.

1. Sign in to the AWS Management Console (Click the **Open AWS Console** button on the left, and log in with the provided username and password).

2. Navigate to the **EC2 Dashboard** (**Services > Compute > EC2**). Select the **Northern Virginia** region (`us-east-1`) using the pulldown menu in the top right. In the EC2 Dashboard and the VPC Dashboard, you can review the resources deployed later in the lab.



You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	0	Dedicated Hosts	0	Elastic IPs	
Instances	0	Key pairs	1	Load balancers	
Placement groups	0	Security groups	1	Snapshots	
Volumes	0				

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using th AWS Launch Wizard for SQL Server. Learn more

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

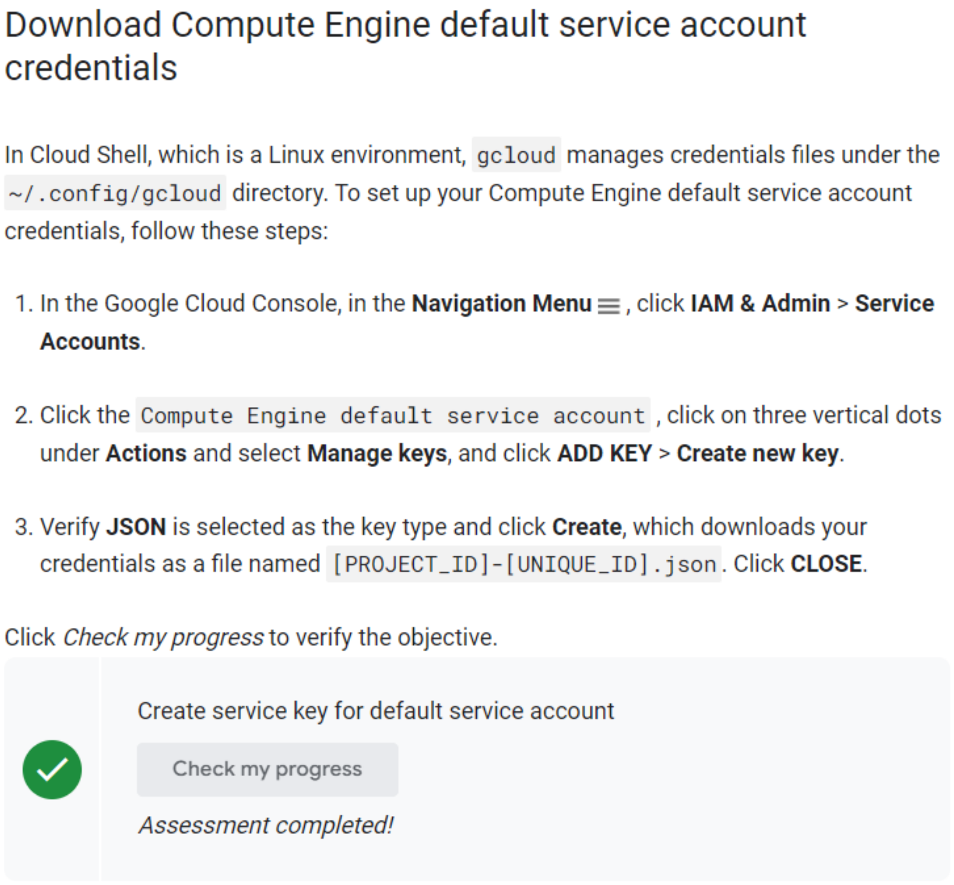
Note: Your instances will launch in the US East (N. Virginia) Region

Service health

Region: US East (N. Virginia)

Status: This service is operating normally

US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3



Download Compute Engine default service account credentials

In Cloud Shell, which is a Linux environment, `gcloud` manages credentials files under the `~/.config/gcloud` directory. To set up your Compute Engine default service account credentials, follow these steps:

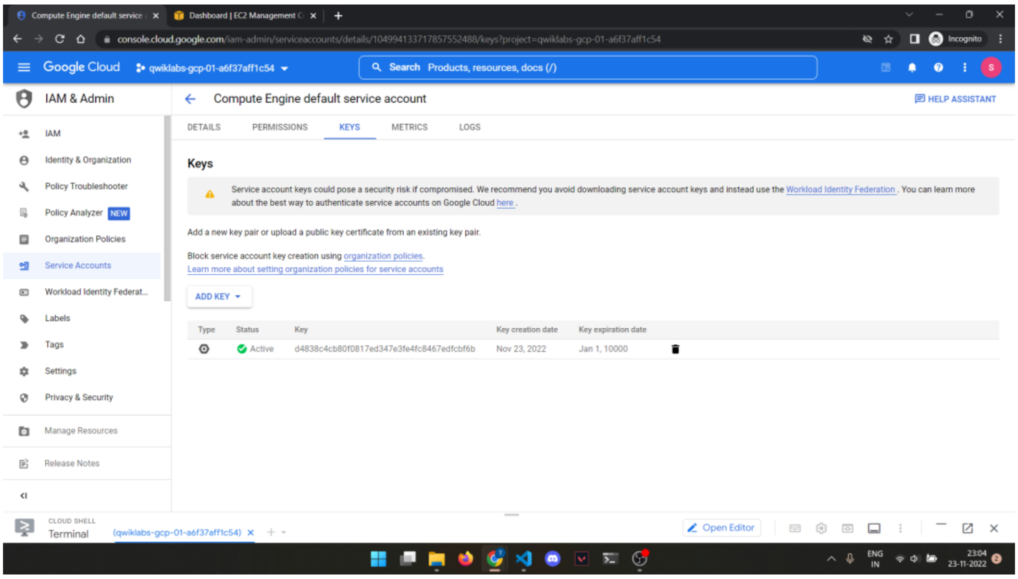
1. In the Google Cloud Console, in the **Navigation Menu** ☰, click **IAM & Admin** > **Service Accounts**.
2. Click the `Compute Engine default service account`, click on three vertical dots under **Actions** and select **Manage keys**, and click **ADD KEY** > **Create new key**.
3. Verify **JSON** is selected as the key type and click **Create**, which downloads your credentials as a file named `[PROJECT_ID]-[UNIQUE_ID].json`. Click **CLOSE**.

Click *Check my progress* to verify the objective.

✓ Create service key for default service account

Check my progress

Assessment completed!



Compute Engine default service account

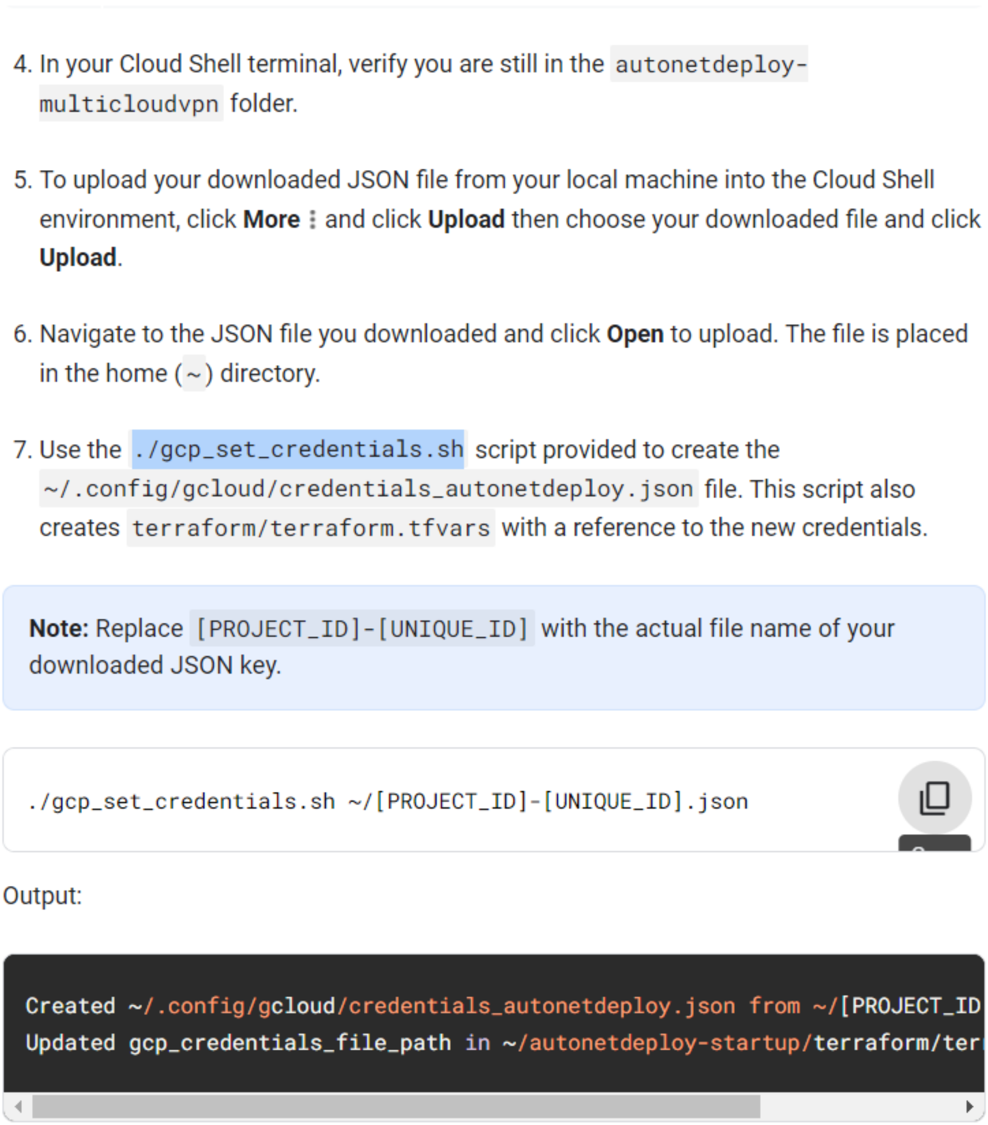
Keys

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

Type	Status	Key	Key creation date	Key expiration date
⊙	✓ Active	d4838c4cb80f0817ed347e3fe4fc8467edfcbf6b	Nov 23, 2022	Jan 1, 10000



4. In your Cloud Shell terminal, verify you are still in the `autonetdeploy-multicloudvpn` folder.

5. To upload your downloaded JSON file from your local machine into the Cloud Shell environment, click **More** ⋮ and click **Upload** then choose your downloaded file and click **Upload**.

6. Navigate to the JSON file you downloaded and click **Open** to upload. The file is placed in the home (`~`) directory.

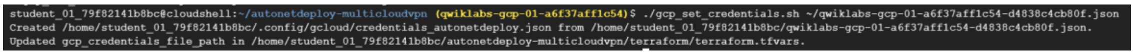
7. Use the `./gcp_set_credentials.sh` script provided to create the `~/.config/gcloud/credentials_autonetdeploy.json` file. This script also creates `terraform/terraform.tfvars` with a reference to the new credentials.

Note: Replace `[PROJECT_ID]-[UNIQUE_ID]` with the actual file name of your downloaded JSON key.

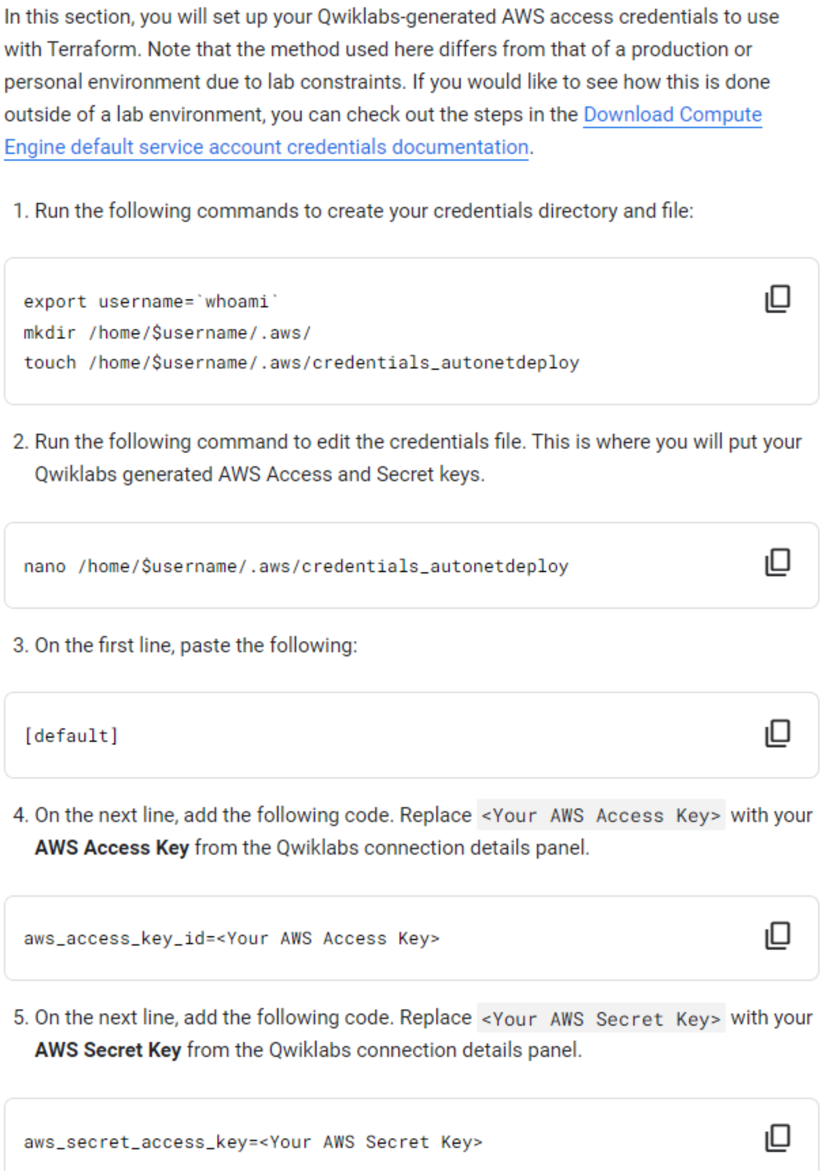
```
./gcp_set_credentials.sh ~/[PROJECT_ID]-[UNIQUE_ID].json
```

Output:

```
Created ~/.config/gcloud/credentials_autonetdeploy.json from ~/[PROJECT_ID
Updated gcp_credentials_file_path in ~/autonetdeploy-startup/terraform/ter
```



```
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ ./gcp_set_credentials.sh ~/qwiklabs-gcp-01-a6f37aff1c54-d4838c4cb80f.json
Created /home/student_01_79f82141b8bc/.config/gcloud/credentials_autonetdeploy.json from /home/student_01_79f82141b8bc/qwiklabs-gcp-01-a6f37aff1c54-d4838c4cb80f.json.
Updated gcp_credentials_file_path in /home/student_01_79f82141b8bc/autonetdeploy-multicloudvpn/terraform/terraform.tfvars.
```



In this section, you will set up your Qwiklabs-generated AWS access credentials to use with Terraform. Note that the method used here differs from that of a production or personal environment due to lab constraints. If you would like to see how this is done outside of a lab environment, you can check out the steps in the Download Compute Engine default service account credentials documentation.

1. Run the following commands to create your credentials directory and file:

```
export username=`whoami`
mkdir /home/$username/.aws/
touch /home/$username/.aws/credentials_autonetdeploy
```

2. Run the following command to edit the credentials file. This is where you will put your Qwiklabs generated AWS Access and Secret keys.

```
nano /home/$username/.aws/credentials_autonetdeploy
```

3. On the first line, paste the following:

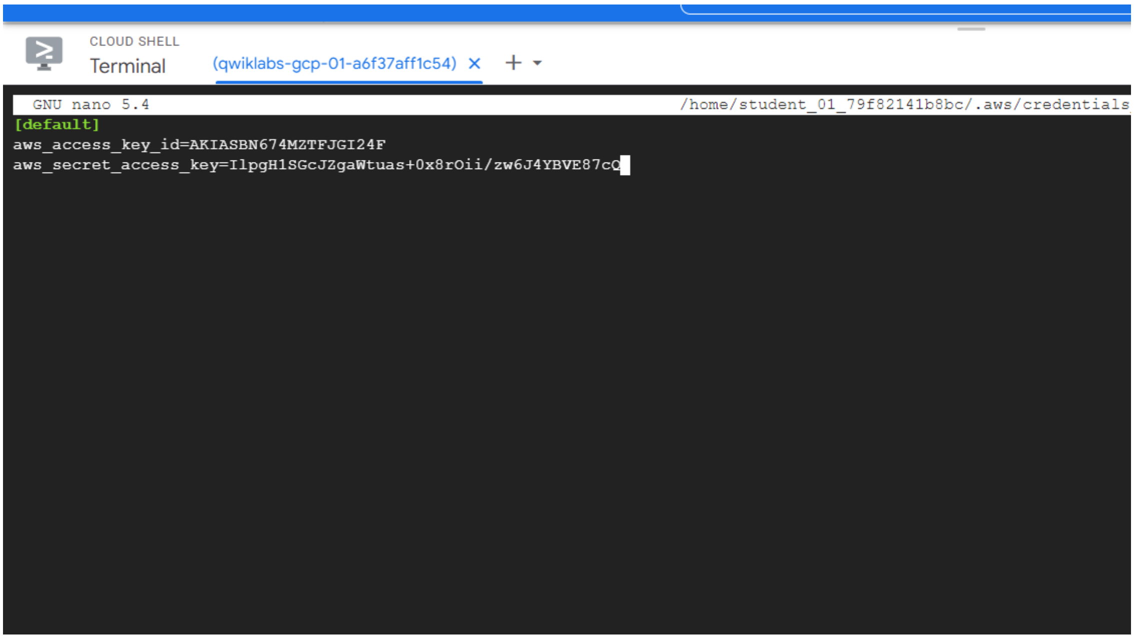
```
[default]
```

4. On the next line, add the following code. Replace `<Your AWS Access Key>` with your **AWS Access Key** from the Qwiklabs connection details panel.

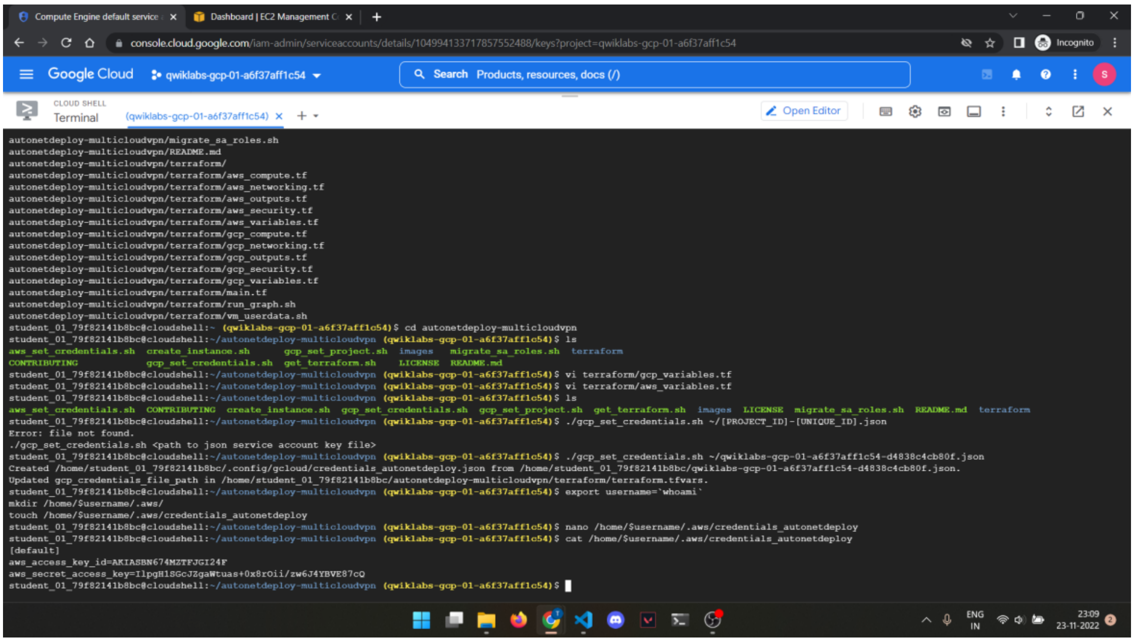
```
aws_access_key_id=<Your AWS Access Key>
```

5. On the next line, add the following code. Replace `<Your AWS Secret Key>` with your **AWS Secret Key** from the Qwiklabs connection details panel.

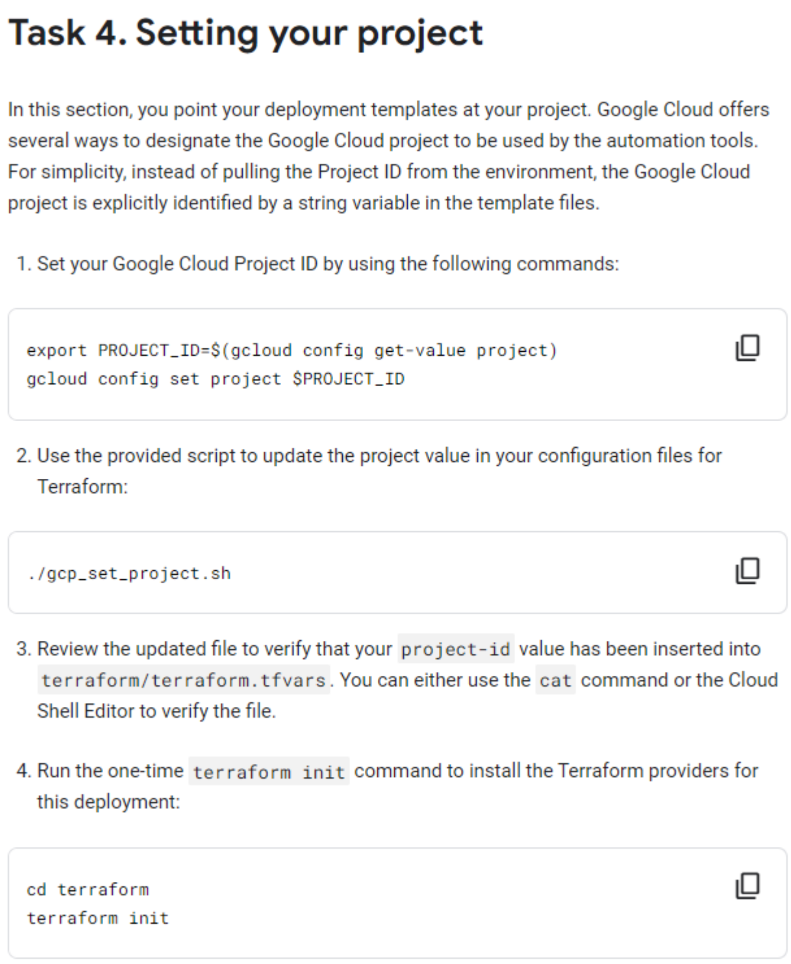
```
aws_secret_access_key=<Your AWS Secret Key>
```



```
  GNU nano 5.4                                                 /home/student_01_79f82141b8bc/.aws/credentials
[default]
aws_access_key_id=AKIASBN674MZTFJGI24F
aws_secret_access_key=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
```


Compute Engine default service
Dashboard | EC2 Management C
console.cloud.google.com/iam-admin/serviceaccounts/details/104994133717857552488/keys?project=qwiklabs-gcp-01-a6f37aff1c54
Incognito
Google Cloud
qwiklabs-gcp-01-a6f37aff1c54
Search Products, resources, docs (/)
CLOUD SHELL
Terminal
(qwiklabs-gcp-01-a6f37aff1c54)
Open Editor
ENG
IN
23:09
23-11-2022

```
autonetdeploy-multicloudvpn/migrate_sa_roles.sh
autonetdeploy-multicloudvpn/README.md
autonetdeploy-multicloudvpn/terraform/
autonetdeploy-multicloudvpn/terraform/aws_compute.tf
autonetdeploy-multicloudvpn/terraform/aws_networking.tf
autonetdeploy-multicloudvpn/terraform/aws_outputs.tf
autonetdeploy-multicloudvpn/terraform/aws_security.tf
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_01_79f82141b8bc@cloudshell:~ (qwiklabs-gcp-01-a6f37aff1c54)$ cd autonetdeploy-multicloudvpn
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ ls
aws_set_credentials.sh  create_instance.sh      gcp_set_project.sh  images   migrate_sa_roles.sh  terraform
CONTRIBUTING            gcp_set_credentials.sh  get_terraform.sh    LICENSE  README.md
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ vi terraform/gcp_variables.tf
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ vi terraform/aws_variables.tf
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ ls
aws_set_credentials.sh  CONTRIBUTING  create_instance.sh  gcp_set_credentials.sh  gcp_set_project.sh  get_terraform.sh  images  LICENSE  migrate_sa_roles.sh  README.md  terraform
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ ./gcp_set_credentials.sh ~/[PROJECT_ID]-[UNIQUE_ID].json
Error: file not found.
./gcp_set_credentials.sh <path to json service account key file>
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ ./gcp_set_credentials.sh ~/qwiklabs-gcp-01-a6f37aff1c54-d4838c4cb80f.json
Created /home/student_01_79f82141b8bc/.config/gcloud/credentials_autonetdeploy.json from /home/student_01_79f82141b8bc/qwiklabs-gcp-01-a6f37aff1c54-d4838c4cb80f.json.
Updated gcp_credentials_file_path in /home/student_01_79f82141b8bc/autonetdeploy-multicloudvpn/terraform/terraform.tfvars.
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ export username=`whoami`
mkdir /home/$username/.aws/
touch /home/$username/.aws/credentials_autonetdeploy
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ nano /home/$username/.aws/credentials_autonetdeploy
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ cat /home/$username/.aws/credentials_autonetdeploy
[default]
aws_access_key_id=AKIASBN674MZTFJGI24F
aws_secret_access_key=IlpgH1SGcJZgaWtuas+0x8rOii/zw6J4YBVE87cQ
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn (qwiklabs-gcp-01-a6f37aff1c54)$ 
```



Task 4. Setting your project

In this section, you point your deployment templates at your project. Google Cloud offers several ways to designate the Google Cloud project to be used by the automation tools. For simplicity, instead of pulling the Project ID from the environment, the Google Cloud project is explicitly identified by a string variable in the template files.

1. Set your Google Cloud Project ID by using the following commands:

```
export PROJECT_ID=$(gcloud config get-value project)
gcloud config set project $PROJECT_ID
```

2. Use the provided script to update the project value in your configuration files for Terraform:

```
./gcp_set_project.sh
```

3. Review the updated file to verify that your `project-id` value has been inserted into `terraform/terraform.tfvars`. You can either use the `cat` command or the Cloud Shell Editor to verify the file.

4. Run the one-time `terraform init` command to install the Terraform providers for this deployment:

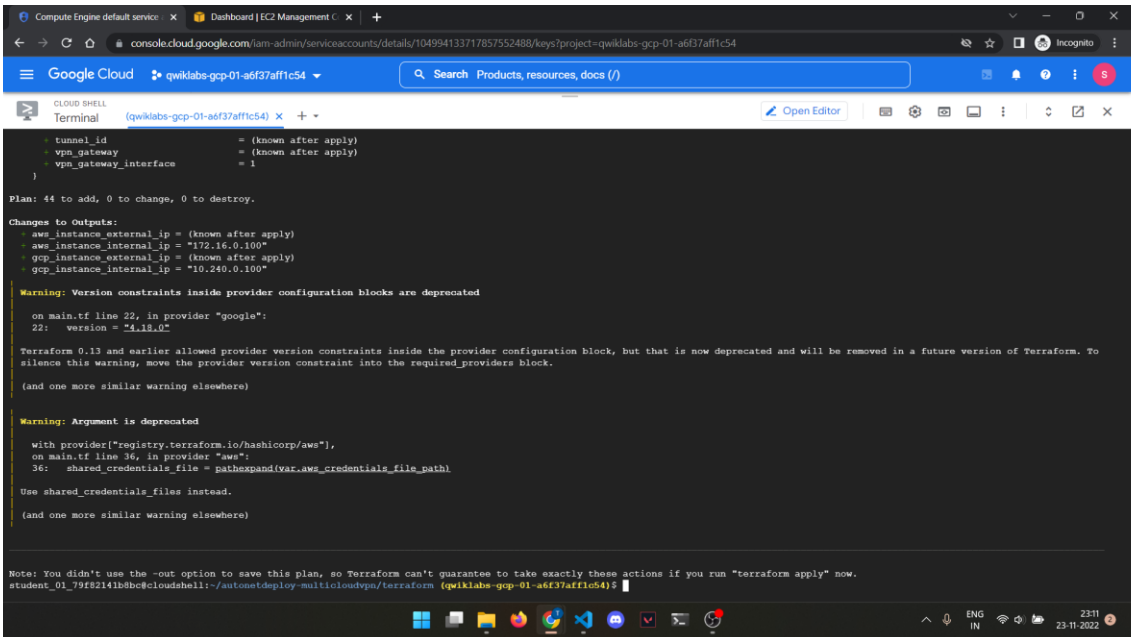
```
cd terraform
terraform init
```

5. Run the Terraform `plan` command to verify your credentials:

```
terraform plan
```

Output:

```
Refreshing Terraform state in-memory prior to plan...
...
 +google_compute_instance.gcp-vm
...
Plan: 34 to add, 0 to change, 0 to destroy.
```



Compute Engine default service
Dashboard | EC2 Management C
console.cloud.google.com/iam-admin/serviceaccounts/details/104994133717857552488/keys?project=qwiklabs-gcp-01-a6f37aff1c54
Incognito
Google Cloud
qwiklabs-gcp-01-a6f37aff1c54
Search Products, resources, docs (/)
CLOUD SHELL
Terminal
(qwiklabs-gcp-01-a6f37aff1c54)
Open Editor
ENG
IN
23:11
23-11-2022

```
      + tunnel_id                          = (known after apply)
      + vpn_gateway                        = (known after apply)
      + vpn_gateway_interface              = 1
    }

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_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-01-a6f37aff1c54)$ 
```

Generate a key pair

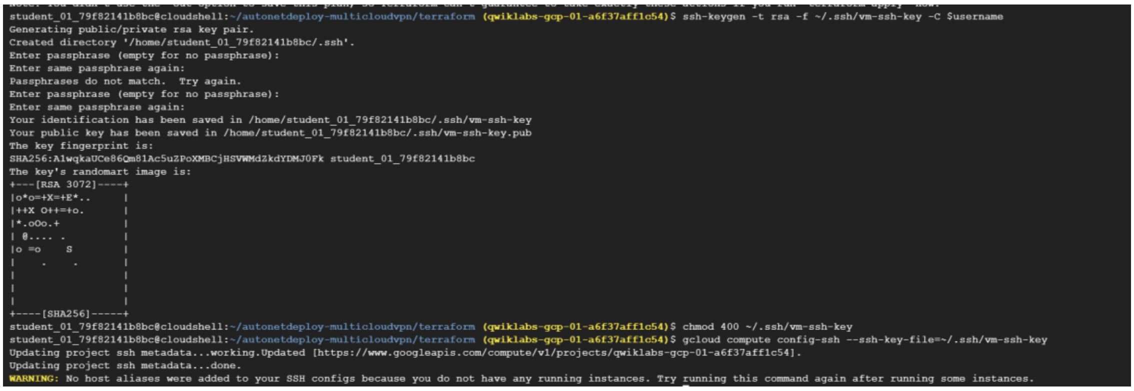
1. In Cloud Shell, use `ssh-keygen` to generate a new key pair:

```
ssh-keygen -t rsa -f ~/.ssh/vm-ssh-key -C $username
```

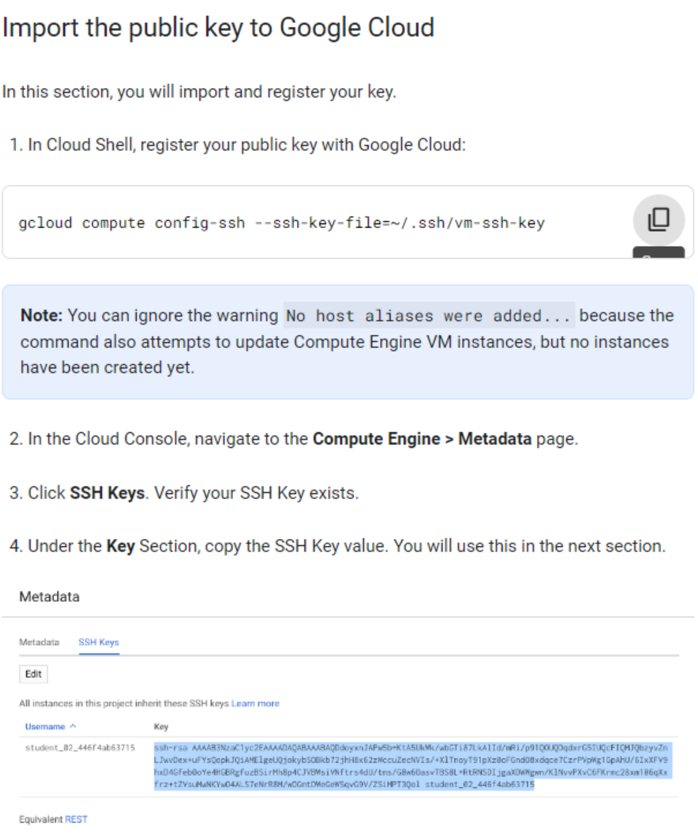
When asked for a passphrase, press **Enter** twice to leave it blank.

2. Restrict access to your private key. This is a best practice.

```
chmod 400 ~/.ssh/vm-ssh-key
```



```
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_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-01-a6f37aff1c54)$ ssh-keygen -t rsa -f ~/.ssh/vm-ssh-key -C $username
Generating public/private rsa key pair.
Created directory '/home/student_01_79f82141b8bc/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/student_01_79f82141b8bc/.ssh/vm-ssh-key
Your public key has been saved in /home/student_01_79f82141b8bc/.ssh/vm-ssh-key.pub
The key fingerprint is:
SHA256:A1wqkaUCe86Qm81Ac5uZPoXMBCjHSVWMdZkdYDMJ0Fk student_01_79f82141b8bc
The key's randomart image is:
+---[RSA 3072]----+
|o*o=+X=+E*..     |
|++X O++=+o.      |
|*.oOo.+          |
| @.... .         |
|o =o    S        |
|    .    .       |
|                 |
|                 |
|                 |
+----[SHA256]-----+
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-01-a6f37aff1c54)$ chmod 400 ~/.ssh/vm-ssh-key
student_01_79f82141b8bc@cloudshell:~/autonetdeploy-multicloudvpn/terraform (qwiklabs-gcp-01-a6f37aff1c54)$ 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-01-a6f37aff1c54].
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.
```



Import the public key to Google Cloud

In this section, you will import and register your key.

1. In Cloud Shell, register your public key with Google Cloud:

```
gcloud compute config-ssh --ssh-key-file=~/.ssh/vm-ssh-key
```

Note: You can ignore the warning `No host aliases were added...` because the command also attempts to update Compute Engine VM instances, but no instances have been created yet.

2. In the Cloud Console, navigate to the **Compute Engine > Metadata** page.

3. Click **SSH Keys**. Verify your SSH Key exists.

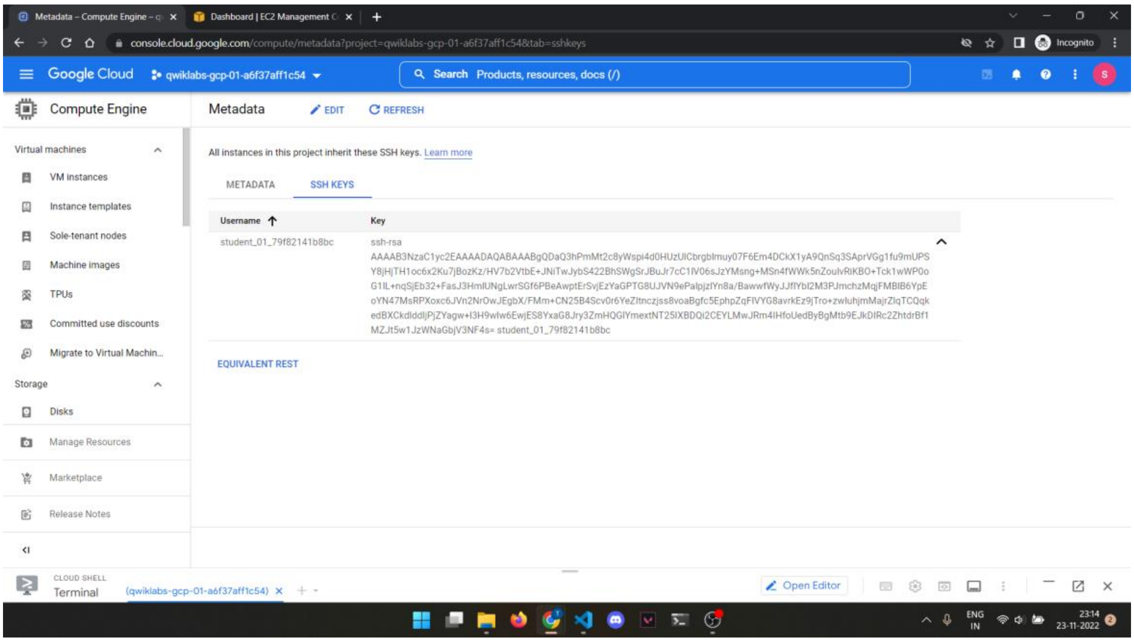
4. Under the **Key** Section, copy the SSH Key value. You will use this in the next section.

Metadata

All instances in this project inherit these SSH keys Learn more

Username ^	Key
student_02_446f4ab63715	ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDdoyxnJAPw5b+KtA5UkMk/wbGTi87LkAlId/mRi/p91QOUQOqdxrG5IUQcFIQMJQbzyvZnLJwvDex+uFYsQopkJQiAMElgeUQjokybSOBkb72jhH8x62zWccuZecNVIs/+XlTnoyT91pXz0oFGnd08xdqce7CzrPVpWg1GpAhU/6IxXFV9hxD4Gfeb0oYe4HGBRgfuzBSirMh8p4CJVBMsiVNftrs4dU/tms/G8w60asvTBS8L+RtRN5DIjgaXDWWgwn/KlNvvPXvC6FKrmc28xm106qXxfrz+tZVsuMwNKYwO4ALS7eNrR8M/wDGntDMeGeWSqvG9V/ZSiMPT3Qol student_02_446f4ab63715

Equivalent REST



Metadata – Compute Engine – q
Dashboard | EC2 Management C
console.cloud.google.com/compute/metadata?project=qwiklabs-gcp-01-a6f37aff1c54&tab=sshkeys
Incognito
Google Cloud
qwiklabs-gcp-01-a6f37aff1c54
Search Products, resources, docs (/)
Compute Engine
Virtual machines
VM instances
Instance templates
Sole-tenant nodes
Machine images
TPUs
Committed use discounts
Migrate to Virtual Machin...
Storage
Disks
Manage Resources
Marketplace
Release Notes
CLOUD SHELL
Terminal
(qwiklabs-gcp-01-a6f37aff1c54)
Open Editor
ENG
IN
23:14
23-11-2022

Metadata EDIT REFRESH

All instances in this project inherit these SSH keys. Learn more

METADATA SSH KEYS

Username ↑	Key
student_01_79f82141b8bc	ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDaQ3hPmMt2c8yWspi4d0HUzUlCbrgblmuy07F6Em4DCkX1yA9QnSq3SAprVGg1fu9mUPSY8jHjTH1oc6x2Ku7jBozKz/HV7b2VtbE+JNiTwJybS422BhSWgSrJBuJr7cC1lV06sJzYMsng+MSn4fWWk5nZoulvRiKBO+Tck1wWP0oG1IL+nqSjEb32+FasJ3HmlUNgLwrSGf6PBeAwptErSvjEzYaGPTG8UJVN9ePalpjzlYn8a/BawwfWyJJflYbI2M3PJmchzMqjFMBlB6YpEoYN47MsRPXoxc6JVn2NrOwJEgbX/FMm+CN25B4Scv0r6YeZltnczjss8voaBgfc5EphpZqFlVYG8avrkEz9jTro+zwluhjmMajrZlqTCQqkedBXCkdlddljPjZYagw+I3H9wlw6EwjES8YxaG8Jry3ZmHQGlYmextNT25lXBDQi2CEYLMwJRm4lHfoUedByBgMtb9EJkDlRc2ZhtdrBf1MZJt5w1JzWNaGbjV3NF4s= student_01_79f82141b8bc

EQUIVALENT REST

Import the public key to AWS

You can reuse the public key file generated with Google Cloud.

1. In the AWS Management Console, navigate to **Services > Compute > EC2**.

> **Note: Verify** that you are in the **US-East (N. Virginia)** `us-east-1` region.

2. In **EC2 Dashboard**, under the **Network & Security** group on the left, click **Key Pairs**.

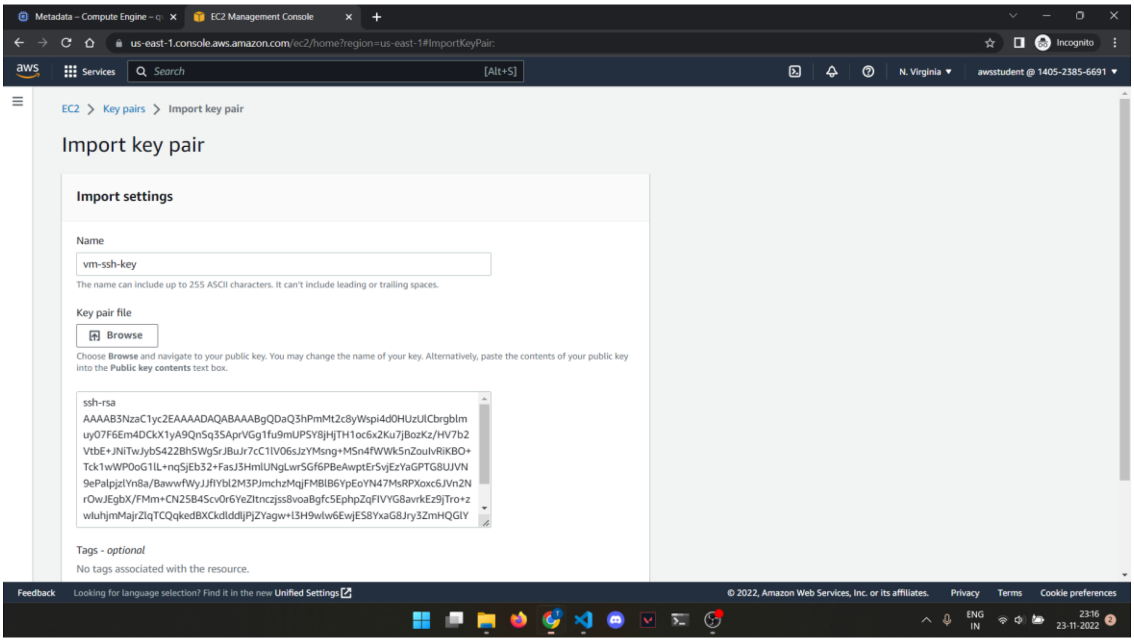
3. Click **Actions > Import Key Pair**.

4. For the name, enter: `vm-ssh-key`.

5. Paste the contents of your Google Cloud public key (**Compute Engine > Metadata > SSH Keys**) into the **Public key contents** box.

6. Verify that the contents are of the expected form: `ssh-rsa [KEY_DATA] [USERNAME]`.

7. Click **Import Key Pair**.



EC2 > Key pairs > Import key pair

Import key pair

Import settings

Name: vm-ssh-key

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

Key pair file: Browse

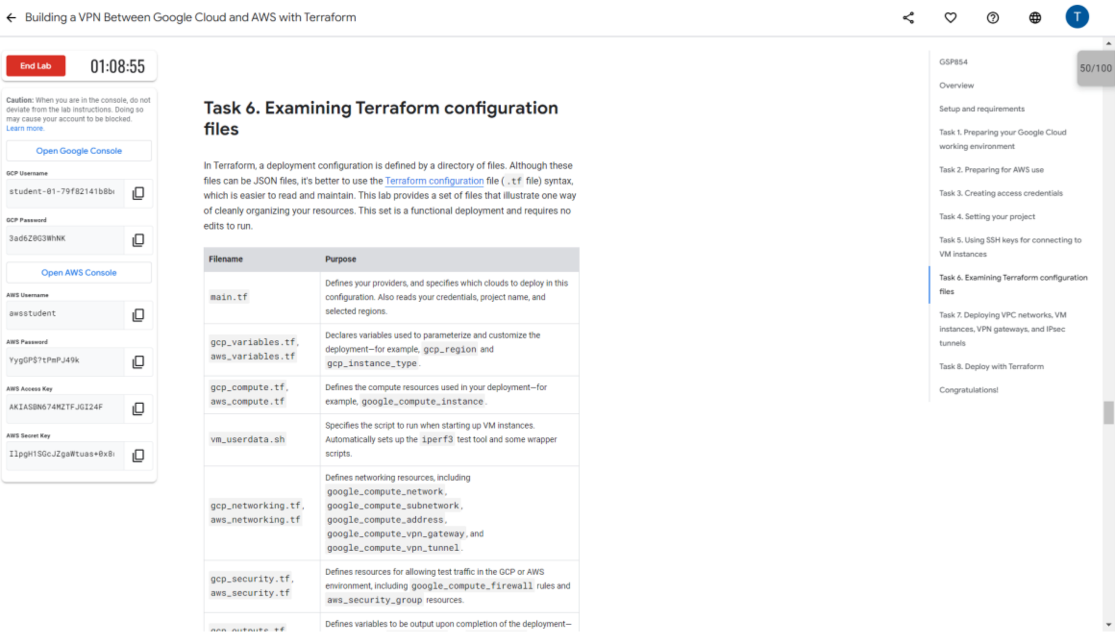
Choose **Browse** and navigate to your public key. You may change the name of your key. Alternatively, paste the contents of your public key into the **Public key contents** text box.

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDaQ3hPmMt2c8yWspi4d0HUzUlCbrgblm
uy07F6Em4DCkX1yA9QnSq3SAprVGg1fu9mUPSY8jHjTH1oc6x2Ku7jBozKz/HV7b2
VtbE+JNiTwJybS422BhSWgSrJBuJr7cC1lV06sJzYMsng+MSn4fWWk5nZoulvRiKBO+
Tck1wWP0oG1lL+nqSjEb32+FasJ3HmlUNgLwrSGf6PBeAwptErSvjEzYaGPTG8UJVN
9ePalpjzlYn8a/BawwfWyJJflYbl2M3PJmchzMqjFMBlB6YpEoYN47MsRPXoxc6JVn2N
rOwJEgbX/FMm+CN25B4Scv0r6YeZltnczjss8voaBgfc5EphpZqFIVYG8avrkEz9jTro+z
wluhjmMajrZlqTCQqkedBXCkdlddljPjZYagw+l3H9wlw6EwjES8YxaG8Jry3ZmHQGlY
```

Tags - *optional*

No tags associated with the resource.

Metadata – Compute Engine
EC2 Management Console
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ImportKeyPair:
Incognito
Services
Search
[Alt+S]
N. Virginia
awsstudent @ 1405-2385-6691
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
ENG IN
23:16
23-11-2022



← Building a VPN Between Google Cloud and AWS with Terraform
T
End Lab
01:08:55

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

GCP Username: student-01-79f82141b8b

GCP Password: 3ad6Z0G3WhNK

AWS Username: awsstudent

AWS Password: YygGP$?tPmPJ49k

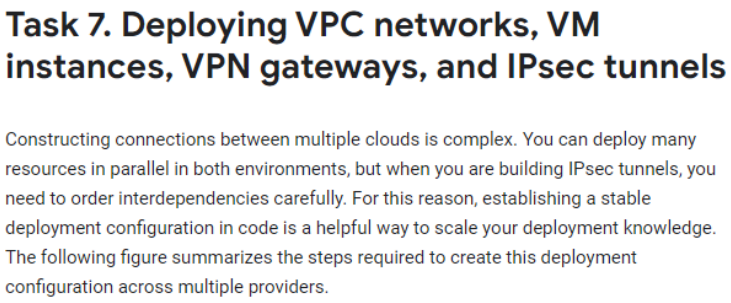
AWS Access Key: AKIASBN674MZTFJGI24F

AWS Secret Key: IlpgH1SGcJZgaWtuas+0x8

Task 6. Examining Terraform configuration files

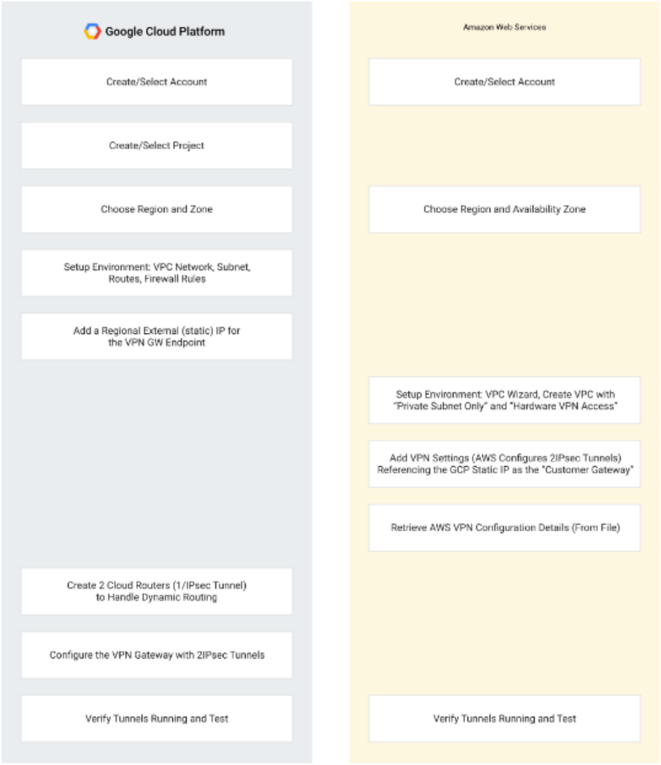
In Terraform, a deployment configuration is defined by a directory of files. Although these files can be JSON files, it's better to use the Terraform configuration file (`.tf` file) syntax, which is easier to read and maintain. This lab provides a set of files that illustrate one way of cleanly organizing your resources. This set is a functional deployment and requires no edits to run.

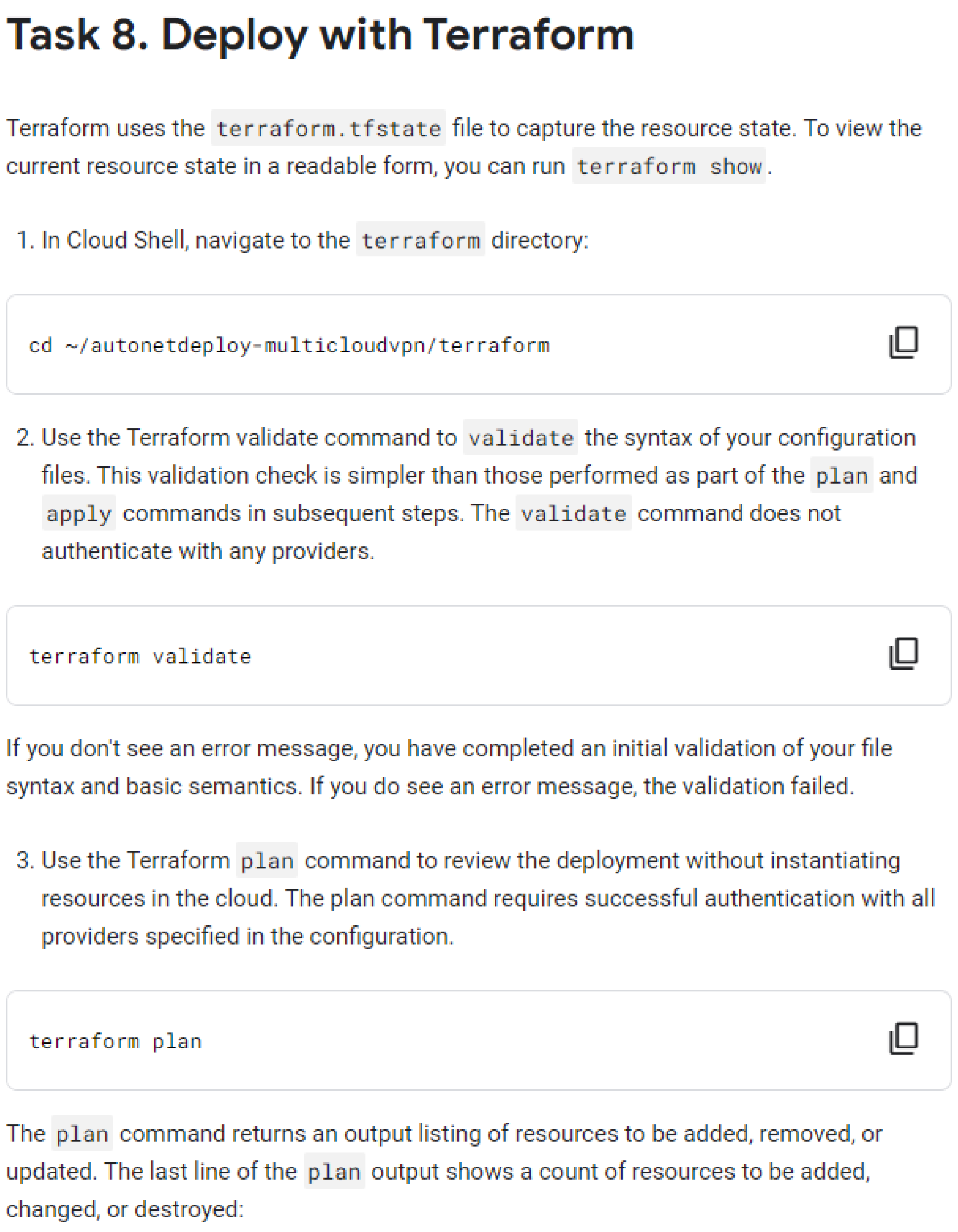
Filename	Purpose
`main.tf`	Defines your providers, and specifies which clouds to deploy in this configuration. Also reads your credentials, project name, and selected regions.
`gcp_variables.tf`, `aws_variables.tf`	Declares variables used to parameterize and customize the deployment—for example, `gcp_region` and `gcp_instance_type`.
`gcp_compute.tf`, `aws_compute.tf`	Defines the compute resources used in your deployment—for example, `google_compute_instance`.
`vm_userdata.sh`	Specifies the script to run when starting up VM instances. Automatically sets up the `iperf3` test tool and some wrapper scripts.
`gcp_networking.tf`, `aws_networking.tf`	Defines networking resources, including `google_compute_network`, `google_compute_subnetwork`, `google_compute_address`, `google_compute_vpn_gateway`, and `google_compute_vpn_tunnel`.
`gcp_security.tf`, `aws_security.tf`	Defines resources for allowing test traffic in the GCP or AWS environment, including `google_compute_firewall` rules and `aws_security_group` resources.
`gcp_outputs.tf`	Defines variables to be output upon completion of the deployment—



Task 7. Deploying VPC networks, VM instances, VPN gateways, and IPsec tunnels

Constructing connections between multiple clouds is complex. You can deploy many resources in parallel in both environments, but when you are building IPsec tunnels, you need to order interdependencies carefully. For this reason, establishing a stable deployment configuration in code is a helpful way to scale your deployment knowledge. The following figure summarizes the steps required to create this deployment configuration across multiple providers.


Google Cloud Platform
Create/Select Account
Create/Select Project
Choose Region and Zone
Setup Environment: VPC Network, Subnet, Routes, Firewall Rules
Add a Regional External (static) IP for the VPN GW Endpoint
Create 2 Cloud Routers (1/IPsec Tunnel) to Handle Dynamic Routing
Configure the VPN Gateway with 2IPsec Tunnels
Verify Tunnels Running and Test
Amazon Web Services
Create/Select Account
Choose Region and Availability Zone
Setup Environment: VPC Wizard, Create VPC with "Private Subnet Only" and "Hardware VPN Access"
Add VPN Settings (AWS Configures 2IPsec Tunnels) Referencing the GCP Static IP as the "Customer Gateway"
Retrieve AWS VPN Configuration Details (From File)
Verify Tunnels Running and Test



Task 8. Deploy with Terraform

Terraform uses the `terraform.tfstate` file to capture the resource state. To view the current resource state in a readable form, you can run `terraform show`.

1. In Cloud Shell, navigate to the `terraform` directory:

```
cd ~/autonetdeploy-multicloudvpn/terraform
```

2. Use the Terraform validate command to `validate` the syntax of your configuration files. This validation check is simpler than those performed as part of the `plan` and `apply` commands in subsequent steps. The `validate` command does not authenticate with any providers.

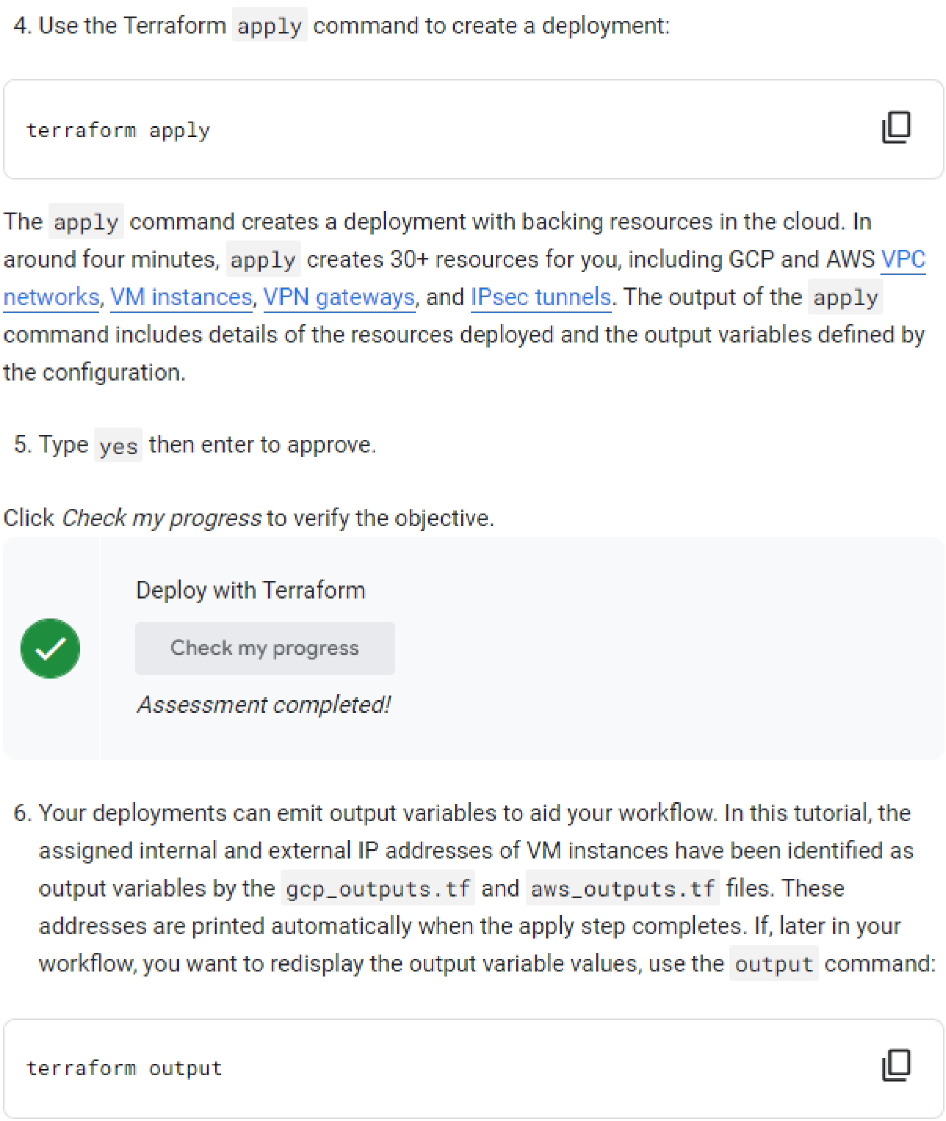
```
terraform validate
```

If you don't see an error message, you have completed an initial validation of your file syntax and basic semantics. If you do see an error message, the validation failed.

3. Use the Terraform `plan` command to review the deployment without instantiating resources in the cloud. The plan command requires successful authentication with all providers specified in the configuration.

```
terraform plan
```

The `plan` command returns an output listing of resources to be added, removed, or updated. The last line of the `plan` output shows a count of resources to be added, changed, or destroyed:



4. Use the Terraform `apply` command to create a deployment:

```
terraform apply
```

The `apply` command creates a deployment with backing resources in the cloud. In around four minutes, `apply` creates 30+ resources for you, including GCP and AWS VPC networks, VM instances, VPN gateways, and IPsec tunnels. The output of the `apply` command includes details of the resources deployed and the output variables defined by the configuration.

5. Type `yes` then enter to approve.

Click *Check my progress* to verify the objective.

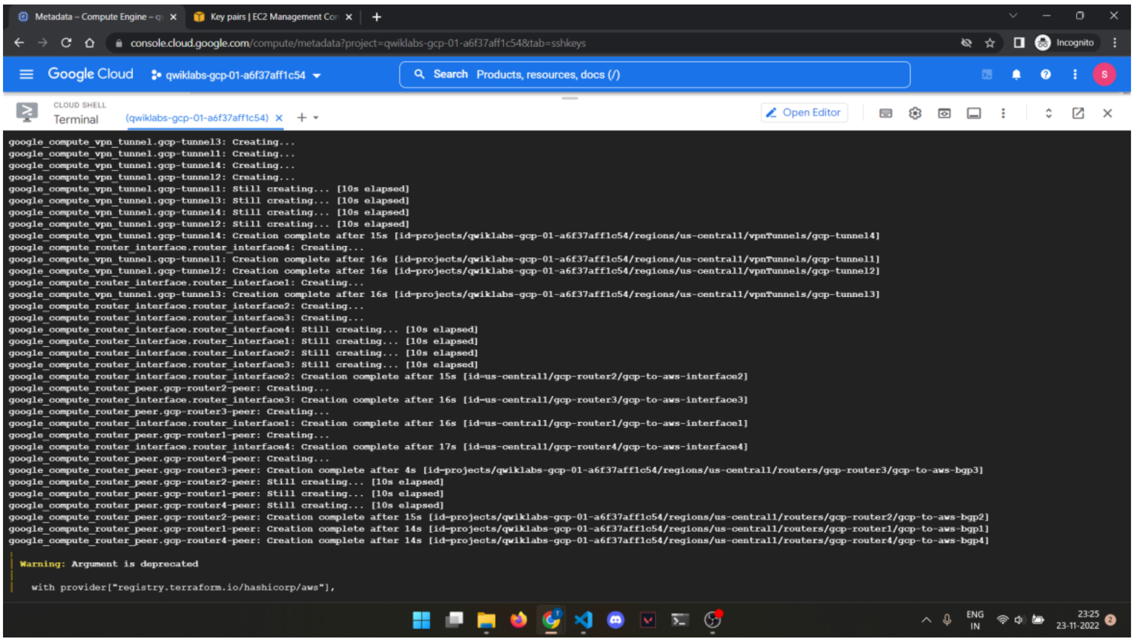
✓ Deploy with Terraform

Check my progress

Assessment completed!

6. Your deployments can emit output variables to aid your workflow. In this tutorial, the assigned internal and external IP addresses of VM instances have been identified as output variables by the `gcp_outputs.tf` and `aws_outputs.tf` files. These addresses are printed automatically when the apply step completes. If, later in your workflow, you want to redisplay the output variable values, use the `output` command:

```
terraform output
```



Metadata – Compute Engine – q
Key pairs | EC2 Management Con
console.cloud.google.com/compute/metadata?project=qwiklabs-gcp-01-a6f37aff1c54&tab=sshkeys
Incognito
Google Cloud
qwiklabs-gcp-01-a6f37aff1c54
Search Products, resources, docs (/)
CLOUD SHELL
Terminal
(qwiklabs-gcp-01-a6f37aff1c54)
Open Editor
ENG
IN
23:25
23-11-2022

```
google_compute_vpn_tunnel.gcp-tunnel3: Creating...
google_compute_vpn_tunnel.gcp-tunnel1: Creating...
google_compute_vpn_tunnel.gcp-tunnel4: Creating...
google_compute_vpn_tunnel.gcp-tunnel2: Creating...
google_compute_vpn_tunnel.gcp-tunnel1: Still creating... [10s elapsed]
google_compute_vpn_tunnel.gcp-tunnel3: Still creating... [10s elapsed]
google_compute_vpn_tunnel.gcp-tunnel4: Still creating... [10s elapsed]
google_compute_vpn_tunnel.gcp-tunnel2: Still creating... [10s elapsed]
google_compute_vpn_tunnel.gcp-tunnel4: Creation complete after 15s [id=projects/qwiklabs-gcp-01-a6f37aff1c54/regions/us-central1/vpnTunnels/gcp-tunnel4]
google_compute_router_interface.router_interface4: Creating...
google_compute_vpn_tunnel.gcp-tunnel1: Creation complete after 16s [id=projects/qwiklabs-gcp-01-a6f37aff1c54/regions/us-central1/vpnTunnels/gcp-tunnel1]
google_compute_vpn_tunnel.gcp-tunnel2: Creation complete after 16s [id=projects/qwiklabs-gcp-01-a6f37aff1c54/regions/us-central1/vpnTunnels/gcp-tunnel2]
google_compute_router_interface.router_interface1: Creating...
google_compute_vpn_tunnel.gcp-tunnel3: Creation complete after 16s [id=projects/qwiklabs-gcp-01-a6f37aff1c54/regions/us-central1/vpnTunnels/gcp-tunnel3]
google_compute_router_interface.router_interface2: Creating...
google_compute_router_interface.router_interface3: Creating...
google_compute_router_interface.router_interface4: Still creating... [10s elapsed]
google_compute_router_interface.router_interface1: Still creating... [10s elapsed]
google_compute_router_interface.router_interface2: Still creating... [10s elapsed]
google_compute_router_interface.router_interface3: Still creating... [10s elapsed]
google_compute_router_interface.router_interface2: Creation complete after 15s [id=us-central1/gcp-router2/gcp-to-aws-interface2]
google_compute_router_peer.gcp-router2-peer: Creating...
google_compute_router_interface.router_interface3: Creation complete after 16s [id=us-central1/gcp-router3/gcp-to-aws-interface3]
google_compute_router_peer.gcp-router3-peer: Creating...
google_compute_router_interface.router_interface1: Creation complete after 16s [id=us-central1/gcp-router1/gcp-to-aws-interface1]
google_compute_router_peer.gcp-router1-peer: Creating...
google_compute_router_interface.router_interface4: Creation complete after 17s [id=us-central1/gcp-router4/gcp-to-aws-interface4]
google_compute_router_peer.gcp-router4-peer: Creating...
google_compute_router_peer.gcp-router3-peer: Creation complete after 4s [id=projects/qwiklabs-gcp-01-a6f37aff1c54/regions/us-central1/routers/gcp-router3/gcp-to-aws-bgp3]
google_compute_router_peer.gcp-router2-peer: Still creating... [10s elapsed]
google_compute_router_peer.gcp-router1-peer: Still creating... [10s elapsed]
google_compute_router_peer.gcp-router4-peer: Still creating... [10s elapsed]
google_compute_router_peer.gcp-router2-peer: Creation complete after 15s [id=projects/qwiklabs-gcp-01-a6f37aff1c54/regions/us-central1/routers/gcp-router2/gcp-to-aws-bgp2]
google_compute_router_peer.gcp-router1-peer: Creation complete after 14s [id=projects/qwiklabs-gcp-01-a6f37aff1c54/regions/us-central1/routers/gcp-router1/gcp-to-aws-bgp1]
google_compute_router_peer.gcp-router4-peer: Creation complete after 14s [id=projects/qwiklabs-gcp-01-a6f37aff1c54/regions/us-central1/routers/gcp-router4/gcp-to-aws-bgp4]
│
│ Warning: Argument is deprecated
│
│   with provider["registry.terraform.io/hashicorp/aws"],
```

Conclusion:

VPN Between Google Cloud and AWS with Terraform was successfully created.