VPC deployment on Google Cloud using Terraform

Contents

Overview

This lab will take you through coding a multi-component cloud deployment into Google Cloud using Terraform. You will deploy a relatively simple VPC, but even this has several components that need either to exist already or be created as part of the deployment.

Elements required in this lab are the VPC itself, Subnets, Routing, NAT Service and Firewall Rules.

This lab breaks the deployment into phases, each a separate task below, to demonstrate component interdependencies:

Task 1: VPC with 2 subnets

Task 2: Task 1 components plus a Router and NAT service

Task 3: Task 1,2 components plus 2 Firewall Rules

Task 4: Task 1,2 & 3 components plus 2 GCE instances for testing

Between each task you will destroy the previous task deployment.

Objectives

In this lab, you will:

1. Deploy a custom VPC
2. In the us-central1 region, create a Public subnet using CIDR 10.0.1.0/24 and a Private subnet using CIDR 10.0.2.0/24
3. Create a Router with a NAT service
4. Create a GCE instance on the public subnet with an internal and external IP and one on the private subnet with just an internal IP.
5. Restrict inbound traffic to the Public Subnet to only SSH traffic from any other source
6. Restrict inbound traffic to the Private subnet, allowing any traffic from the Public subnet only
7. Allow unrestricted outbound traffic
8. Test the configuration by establishing a ssh session to the instance on the public subnet and then from it test communication to the instance on the private subnet.

There is a Solution section at the very end of these instructions. Try to use this only as a last resort if you are struggling to complete the tasks.

Lab Setup

For each lab, you are provided with a new Google Cloud project and a set of resources for a fixed time at no cost.

1. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time. There is no pause feature. You can restart if needed, but you have to start at the beginning.

2. When ready, click **Start lab.**

3. Note your lab credentials (Username and Password). You will use them to sign in to the Google Cloud Console.

4. Click **Open Google Console**.

5. Click **Use another account** and copy/paste credentials for this lab into the prompts. If you use other credentials, you'll receive errors or incur charges.

6. Accept the terms and skip the recovery resource page.

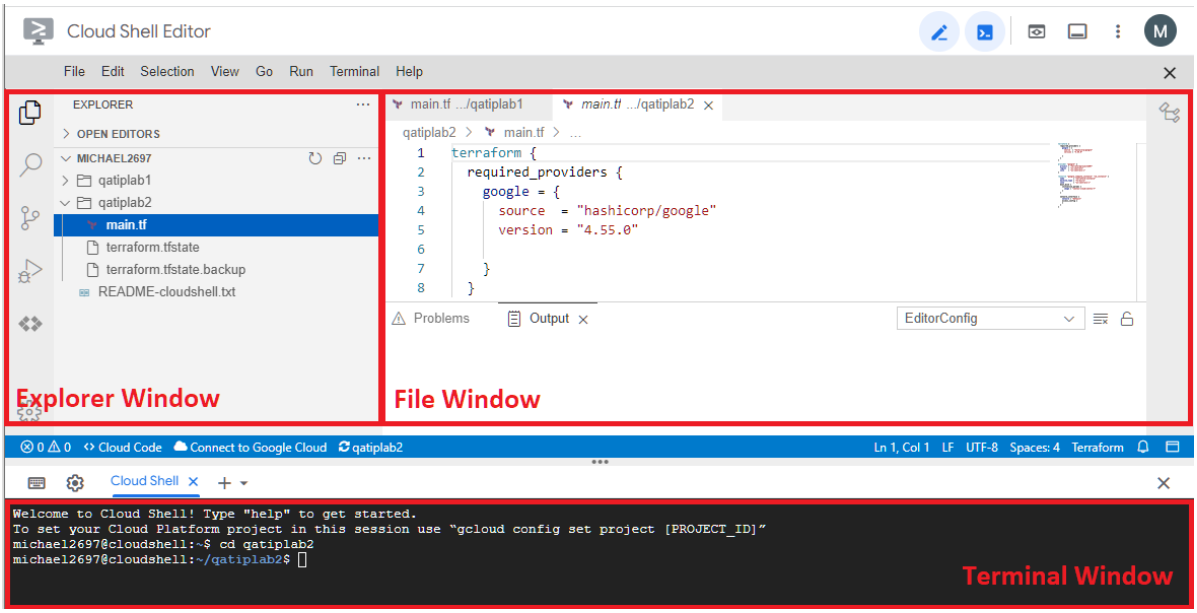
Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Terraform and Cloud Shell Setup

1. In the Cloud Console, click Activate Cloud Shell 
2. If prompted, click Continue.
3. Select the **Open in a new window** icon 
4. Select the **Open Editor** icon 
5. Select **Open Home Workspace** if explorer pane does not display.
6. Create a directory for your Terraform configuration and initial files by running the following commands in the terminal:

   cd ~
   mkdir ./lab
   cd ./lab
   touch main.tf

7. In the Explorer, navigate to and click on main.tf to open the empty file for editing...



8. You now have 2 browser tabs open. The current tab will be referred to as the `IDE` throughout these instructions.

9. Switch to the Google console tab and close the cloud shell. This tab will be referred to as the `Console` throughout these instructions.
10. Switch back to the `IDE`
11. Execute all terraform commands from within the lab directory

Task 1. Create the basic VPC

References

https://registry.terraform.io/providers/hashicorp/google/4.55.0

https://registry.terraform.io/providers/hashicorp/google/4.55.0/docs/resources/compute_network

https://registry.terraform.io/providers/hashicorp/google/4.55.0/docs/resources/compute_subnetwork

Try it yourself

The aim of this task is to:

1. Create a VPC resource **google_compute_network.lab_vpc** named **lab-vpc**
2. On this VPC, create a subnet resource **google_compute_subnetwork.public_subnet** in **us-central1** named **public-subnet** using CIDR **10.0.1.0\24**
3. Create a second subnet resource **google_compute_subnetwork.private_subnet**, named **private-subnet** using CIDR **10.0.2.0\24**
4. Destroy your deployment ahead of enhancing your code in Task 2.

If you feel comfortable doing so, then attempt to complete this task without referencing the step-by-step instructions below. You can verify your attempt by comparing your code with the Task 1 section of the solution code at the end of this document.

Note: Provider version 4.55.0 should be used throughout this lab.

Step by Step

1. Review Terraform Google Provider documentation: https://registry.terraform.io/providers/hashicorp/google/4.55.0/docs
2. Click `Use Provider`
3. Copy the code block into main.tf in the `IDE` For convenience the code is listed below…

```
terraform {
 required_providers {
  google = {
```

```
      source = "hashicorp/google"
      version = "4.55.0"
    }
  }
}

provider "google" {
  # Configuration options
}
```

4. From within the documentation, we see that the provider configuration options typically include the project and region. Copy the sample code and overwrite the empty provider sub-block in main.tf…

```
provider "google" {
  project     = "my-project-id"
  region      = "us-central1"
}
```

5. Update the project value to reflect your lab project id, which can be found on the qwiklabs instruction screen.

6. Review the documentation for creating a VPC; https://registry.terraform.io/providers/hashicorp/google/4.55.0/docs/resources/compute_network

7. Copy the **Example Usage - Network Basic** sample code into **main.tf**

```
resource "google_compute_network" "vpc_network" {
  name = "vpc-network"
}
```

8. Change the resource block name from **vpc_network** to **lab_vpc** and the resource name from **vpc-network** to **lab-vpc**

9. We will only be creating selective subnets on the VPC and therefore need to toggle off the default behaviour which is to create a subnet in every region. Add the argument **auto_create_subnetworks** and set its value to **false**

10. Your modified resource block should now be...

```
resource "google_compute_network" "lab_vpc" {
    name = "lab-vpc"
    auto_create_subnetworks = false
}
```

11. We now need to add 2 subnets to this VPC. Review the documentation for sample code on the google_compute_subnetwork resource, selecting Version 4.55.0;

https://registry.terraform.io/providers/hashicorp/google/4.55.0/docs/resources/compute_subnetwork

12. We need 2 subnets, so the simplest option is to create 2 copies of this resource block. What if we wanted many more though ? We will discuss more efficient options in a later lab. For now, though, copy the first example code block into main.tf.

```
resource "google_compute_subnetwork" "network-with-private-secondary-ip-ranges" {
 name        = "test-subnetwork"
 ip_cidr_range = "10.2.0.0/16"
 region      = "us-central1"
 network     = google_compute_network.custom-test.id
 secondary_ip_range {
  range_name    = "tf-test-secondary-range-update1"
  ip_cidr_range = "192.168.10.0/24"
 }
}
```

13. Change the resource block name from **network-with-private-secondary-ip-ranges** to **public_subnet**
14. Change the resource name from **test-subnetwork** to **public-subnet**,
15. Change the cidr block from **10.2.0.0/16** to **10.0.1.0/24**
16. Change the network argument value from **google_compute_network.custom-test.id** to **google_compute_network.lab_vpc.id**.
17. Finally, delete the secondary_ip_range sub-block.
18. The modified block should now be as follows..

```
resource "google_compute_subnetwork" "public_subnet" {
 name = "public-subnet"
 ip_cidr_range = "10.0.1.0/24"
 region      = "us-central1"
 network = google_compute_network.lab_vpc.id
}
```

19. Duplicate this resource block
20. Change the resource block name from **public_subnet** to **private_subnet**
21. Change the resource name from **public-subnet** to **private-subnet**
22. Change the cidr block from **10.0.1.0/24** to **10.0.2.0/24**.
23. The modified block should now be as follows..

```
resource "google_compute_subnetwork" "private_subnet" {
 name = "private-subnet"
 ip_cidr_range = "10.0.2.0/24"
```

```
  region      = "us-central1"
  network = google_compute_network.lab_vpc.id
}
```

24. Save main.tf and run **terraform init**
25. Run **terraform plan** Note any errors and fix if appropriate. Refer to the Task 1 block in the solution code if necessary
26. Run **terraform apply**, entering **yes** when prompted. 3 resources should be added.
27. Switch to the Console
28. Use the Search bar to search for **VPC networks**
29. Verify you see **Lab-VPC**, the VPC we have just deployed
30. Select into **Lab-VPC** and verify you see the 2 subnets we have just deployed.
31. Switch back to the IDE and run **terraform destroy**, entering **yes** when prompted.

Task 2. Provision a Cloud Router and NAT Service

References

https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/compute_router_nat

Try it yourself

If you feel comfortable doing so, then attempt to complete this task without referencing the step-by-step instructions below. You can verify your attempt by comparing your code with the Task 2 block in the solution code at the end of this document.

Important. If you attempted the previous task yourself, then your code, whilst hopefully achieving the objectives specified, may vary slightly from the solution provided at the end of this lab. You are encouraged to continue attempting each task without precise guidance if you feel comfortable doing so. An alternative approach is to 'reset' your code at the start of each task to align it with the solution code prior to moving forward.

To do this now:

1. Ensure you have destroyed any resources
2. Clear the contents of your current main.tf file
3. Scroll down to the Solution section at the end of this document and copy the Task 1 block into main.tf
4. Update the project id in the provider block and save main.tf

The aim of this task is to:

1. Create a Cloud Router resource **google_compute_router.lab_router** named **lab-router** and associate it with **google_compute_network.lab_vpc**
2. Create a router nat resource **google_compute_router_nat.lab_nat** named **lab-nat**, associate it with **google_compute_network.lab_vpc** and **google_compute_subnetwork.private_subnet**

Step by Step

1. Review the Terraform Registry documentation regarding the creation of a Google Cloud resources **google_compute_router_nat** and **google_compute_router_nat** at https://registry.terraform.io/providers/hashicorp/google/4.55.0/docs/resources/compute_router_nat
2. Copy the code examples relating to `google_compute_router`and `google_compute_router_nat` from Example Usage - Router NAT Basic, into main.tf

```
resource "google_compute_router" "router" {
 name    = "my-router"
 region  = google_compute_subnetwork.subnet.region
 network = google_compute_network.net.id

 bgp {
  asn = 64514
 }
}

resource "google_compute_router_nat" "nat" {
 name                               = "my-router-nat"
 router                             = google_compute_router.router.name
 region                             = google_compute_router.router.region
 nat_ip_allocate_option             = "AUTO_ONLY"
 source_subnetwork_ip_ranges_to_nat = "ALL_SUBNETWORKS_ALL_IP_RANGES"

 log_config {
  enable = true
  filter = "ERRORS_ONLY"
 }
}
```

3. Change the name of the resource block **google_compute_router** from **router** to **lab_router**
4. Change the **google_compute_router** name from **my-router** to **lab-router**

5. Change the **google_compute_router** region from **google_compute_subnetwork.subnet.region** to **google_compute_subnetwork.private_subnet.region**
6. Change the **google_compute_router** network from **google_compute_network.net.id** to **google_compute_network.lab_vpc.id**
7. Delete the **bgp** sub-block
8. Change the name of the resource block **google_compute_router_nat** from **nat** to **lab-nat**
9. Change the **google_compute_router_nat** name from **my-router-nat** to **lab-nat**
10. Change the **google_compute_router_nat** router from **google_compute_router.router.name** to **google_compute_router.lab_router.name**
11. Change the **google_compute_router_nat** region from **google_compute_router.router.region** to **google_compute_router.lab_router.region**
12. Change the **source_subnetwork_ip_ranges_to_nat** from **ALL_SUBNETWORKS_ALL_IP_RANGES** to **LIST_OF_SUBNETWORKS**
13. Delete the **log_config sub-block** and replace it with the code below to apply use of the nat gateway to the private subnet...

```
subnetwork {
   name = google_compute_subnetwork.private-subnet.id
   source_ip_ranges_to_nat = ["ALL_IP_RANGES"]
 }
```

14. Confirm the modified blocks are as shown below...

```
resource "google_compute_router" "lab_router" {
  name = "lab-router"
 network = google_compute_network.lab_vpc.id
}

resource "google_compute_router_nat" "lab_nat" {
 name = "lab-nat"
 router = google_compute_router.lab_router.name
 region = google_compute_router.lab_router.region
 nat_ip_allocate_option = "AUTO_ONLY"
 source_subnetwork_ip_ranges_to_nat = "LIST_OF_SUBNETWORKS"

 subnetwork {
   name = google_compute_subnetwork.private_subnet.id
   source_ip_ranges_to_nat = ["ALL_IP_RANGES"]
 }
}
```

15. Save main.tf

16. Now run **terraform plan.** Note any errors and fix if appropriate. Refer to the Task 2 section of the solution code if necessary.
17. Finally, run **terraform apply**, entering **yes** when prompted. 5 resources are created.
18. Switch to the Console
19. Search for **Cloud NAT** and verify the existence of the new NAT Gateway and Cloud Router. Drill into the nat gateway to verify that the gateway will provide **Cloud NAT mapping** to the private-subnet only.
20. As well as network connectivity, we must also consider firewall rules, needed to permit inbound and outbound traffic flows. Every VPC has default rules that deny all inbound traffic and allow all outbound traffic.
21. In Task 3 you will enhance your code to include appropriate firewall rules to your VPC.
22. Switch back to the IDE and run **terraform destroy**, entering **yes** when prompted.

Task 3. Provision Routing tables and Firewall Rules

References:

https://registry.terraform.io/providers/hashicorp/google/4.55.0/docs/resources/compute_firewall

Try it yourself

If you feel comfortable doing so, then attempt to complete this task without referencing the step-by-step instructions below. You can verify your attempt by comparing your code with the Task 3 section of the solution code at the end of this document.

Important If you attempted a previous task yourself, then your code, whilst hopefully achieving the objectives specified, may vary slightly from the solution provided at the end of this lab. You are encouraged to continue attempting each task without precise guidance if you feel comfortable doing so. An alternative approach is to 'reset' your code at the start of each task to align it with the solution code, prior to moving forward.

To do this now:

1. Ensure you have destroyed any resources
2. Clear the contents of your current main.tf file
3. Scroll down to the Solution section at the end of this document and copy the Task 1 and Task 2 blocks into main.tf
4. Update the project id in the provider block and save main.tf

The aim of this task is to:

1. Create a Firewall Rule **lab-private-fw-rule** that will be applied to instances on the private subnet, with a rule that allows unrestricted inbound traffic from **public-subnet** instances. Note: In production we would only open selective ports between the public and private subnets.
2. Use source tag **pub-subnet-vm** and target tag **priv-subnet-vm** as these will be used by instances on the public and private subnets respectively.
3. Create a second Firewall Rule **lab-public-fw-rule** that will be applied to instances on the public subnet, with a rule that allow TCP port 22 inbound traffic from any source IP. Note: In production we would be more selective regarding source ranges.
4. Use target tag **pub-subnet-vm** as this will be used by instances on the public subnets.

Step by Step

1. Review the Terraform Registry documentation regarding the creation of a Google Cloud Firewall Rule resource **google_compute_firewall** https://registry.terraform.io/providers/hashicorp/google/4.55.0/docs/resources/compute_firewall
2. Copy the **Example Usage - Firewall Basic** code block into **main.tf**…

```
resource "google_compute_firewall" "default" {
 name    = "test-firewall"
 network = google_compute_network.default.name

 allow {
  protocol = "icmp"
 }

 allow {
  protocol = "tcp"
  ports    = ["80", "8080", "1000-2000"]
 }

 source_tags = ["web"]
}
```

3. Change the resource block name to **lab_private_fw_rule**
4. Change the resource name to **lab-private-firewall**
5. Change the network value to **google_compute_network.lab_vpc.name**
6. Remove the **allow.icmp** sub-block
7. Change the remaining **allow** sub-block to allow all protocols across all ports (delete the port argument)

8. Add a **source_tag** argument with a list including **pub-subnet-vm**
9. Add a **target_tag** argument with a list including **priv_subnet_vm**
10. The modified block should now be as shown below…

```
resource "google_compute_firewall" "lab_private_fw_rule" {
 name    = "lab-private-firewall"
 network = google_compute_network.lab_vpc.name

 allow {
  protocol = "all"
 }
 source_tags = ["pub-subnet-vm"]
 target_tags = ["priv-subnet-vm"]
}
```

11. Duplicate the entire **google_compute_firewall** resource block and paste a copy into the end of **main.tf**
12. Rename the new resource block to **lab_public_fw_rule**
13. Change the resource name to **lab_public_firewall**
14. Change the protocol argument value from **all** to **tcp** and add a port argument with list value of **"22"**
15. Replace the **source_tags** argument with a **source_ranges** argument containing **"0.0.0.0/0"**
16. Change the target_tags from **priv-subnet-vm** to **pub-subnet-vm**.
17. The modified block should now be as shown below…

```
resource "google_compute_firewall" "lab_public_fw_rules" {
 name    = "lab-public-firewall"
 network = google_compute_network.lab_vpc.name

 allow {
   protocol = "tcp"
   ports    = ["22"]
 }
 source_ranges =["0.0.0.0/0"]
 target_tags = ["pub-subnet-vm"]
}
```

18. Save main.tf
19. Run **terraform plan** Note any errors and fix if appropriate. Refer to Task 3 block in the solution code if necessary.
20. Run **terraform apply**, entering **yes** when prompted. 7 resources are created.

21. Switch to the Console
22. Navigate to the VPCs console and verify the existence of the new firewall rules.
23. In Task 4 you will create GCE instances on each subnet to test the VPC network connectivity and Firewall rules.
24. Switch back to the IDE and run **terraform destroy**, entering **yes** when prompted.

Task 4. Create GCE Instances

Important If you attempted a previous task yourself, then your code, whilst hopefully achieving the objectives specified, may vary slightly from the solution provided at the end of this lab. You are encouraged to continue attempting each task without precise guidance if you feel comfortable doing so. An alternative approach is to 'reset' your code at the start of each task to align it with the solution code, prior to moving forward.

To do this now:

1. Ensure you have destroyed any resources
2. Clear the contents of your current main.tf file
3. Scroll down to the Solution section at the end of this document and copy the Task1, 2 and 3 code blocks into main.tf
4. Update the project id in the provider block and save main.tf

During this task you are required to:

1. Create 2 EC2 Instances of size **e2-small** based on **debian-cloud/debian-11** public image
2. Both instances should be in availability zone **us-central1-a**
3. Name the first instance **PubVM**
4. Place **PubVM** on subnet **Public-Subnet**
5. Allocate **PubVM** a dynamic public IP address
6. Name the second instance **PrivVM**
7. Place **PrivVM** on subnet **Private-Subnet**
8. Do not allocate a public IP address to **PrivVM**
9. Attempt this task without step-by-step guidance. Refer to the Task 4 code block in the solution code if needed.
10. Save and apply the now complete main.tf
11. Do not destroy the deployment as we will move onto testing next.

12. Use the boilerplate code below to achieve these objectives…

```
resource "google_compute_instance" "" {
 name         = ""
 machine_type = ""
 zone         = ""
 tags = [""]

 boot_disk {
  initialize_params {
    image = ""
  }
 }
 network_interface {
  subnetwork =

  access_config {
  }
 }
}
```

Task 5. Testing

1. Using the EC2 Dashboard, note the private IP address of PrivVM
2. Click on SSH against PubVM to connect using SSH-in-browser
3. Ping PrivVM from within the PubVM ssh session **ping -c 3 {private IP of PrivVM}**
4. Pinging PrivVM from PubVM should succeed because there is a route from the public subnet to the private subnet and the firewall rules allow traffic from the public subnet to the private subnet.
5. Modify the lab-private-fw-rules firewall rule with an erroneous source tag entry, changing it from **pub-subnet-vm** to **pub-subnet-vm1**
6. Save main.tf and apply changes with terraform apply
7. Switch back to the ssh session and repeat the ping command.
8. This ping attempt should fail
9. Undo the change made to main.tf, resetting tag to **pub-subnet-vm**
10. Save main.tf and apply changes
11. Repeat the ping attempt
12. The pings should succeed.

13. Type exit to close the SSH session to PubVM
14. End the lab, this will destroy all resources

Congratulations, you have now completed this lab

Solution Code

```
################################################## Task 1 start ####
terraform {
 required_providers {
  google = {
   source = "hashicorp/google"
   version = "4.55.0"
  }
 }
}

provider "google" {
 project     = "{your project id here}"
 region      = "us-central1"
}

resource "google_compute_network" "lab_vpc" {
 name = "lab-vpc"
 auto_create_subnetworks = false
}

resource "google_compute_subnetwork" "public_subnet" {
 name          = "public-subnetwork"
 ip_cidr_range = "10.0.1.0/24"
 region        = "us-central1"
 network       = google_compute_network.lab_vpc.id
}

resource "google_compute_subnetwork" "private_subnet" {
 name          = "private-subnetwork"
 ip_cidr_range = "10.0.2.0/24"
 region        = "us-central1"
 network       = google_compute_network.lab_vpc.id
}

################################################## Task 1 end ####

################################################## Task 2 start ####
resource "google_compute_router" "lab_router" {
  name = "lab-router"
 network = google_compute_network.lab_vpc.id
}

resource "google_compute_router_nat" "lab_nat" {
 name = "lab-nat"
 router = google_compute_router.lab_router.name
```

```
  region = google_compute_router.lab_router.region
  nat_ip_allocate_option = "AUTO_ONLY"
  source_subnetwork_ip_ranges_to_nat = "LIST_OF_SUBNETWORKS"

  subnetwork {
    name = google_compute_subnetwork.private_subnet.id
    source_ip_ranges_to_nat = ["ALL_IP_RANGES"]
  }
}

################################################## Task 2 end ####

################################################## Task 3 start ####
resource "google_compute_firewall" "lab_private_fw_rule" {
  name    = "lab-private-firewall"
  network = google_compute_network.lab_vpc.name

  allow {
    protocol = "all"
  }

  source_tags = ["pub-subnet-vm"]
  target_tags = ["priv-subnet-vm"]
}

resource "google_compute_firewall" "lab_public_fw_rule" {
  name    = "lab-public-firewall"
  network = google_compute_network.lab_vpc.name

  allow {
    protocol = "tcp"
    ports =["22"]
  }

  source_ranges = ["0.0.0.0/0"]
  target_tags = ["pub-subnet-vm"]
}

################################################## Task 3 end ####

################################################## Task 4 start ####
resource "google_compute_instance" "pub_vm" {
  name         = "pub-vm"
  machine_type = "e2-small"
  zone         = "us-central1-a"
  allow_stopping_for_update = true
  tags = ["pub-subnet-vm"]
```

```
  boot_disk {
   initialize_params {
     image = "debian-cloud/debian-11"
    }
  }
  network_interface {
    subnetwork = google_compute_subnetwork.public_subnet.id

    access_config {
    }
  }
}

resource "google_compute_instance" "priv_vm" {
  name         = "priv-vm"
  machine_type = "e2-small"
  zone         = "us-central1-a"
  allow_stopping_for_update = true
  tags = ["priv-subnet-vm"]

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11"
    }
  }
  network_interface {
    subnetwork = google_compute_subnetwork.private_subnet.id
  }
}
################################################## Task 4 end ####
```