Application Load Balancer with Autoscaling on Google Cloud using Terraform

Contents

Overview

This lab will deploy a Global external application load balancer to distribute web traffic across an auto-scaling group of GCE instances.

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 step-by-step tasks.

Objectives

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
touch loadbalancer.tf
touch managed-instance-group.tf
touch template.tf
touch outputs.tf
touch variables.tf
mkdir ./network
cd ./network
touch main.tf
touch outputs.tf
cd ..
```

7. Use Environment Explorer to confirm the file structure now exists.

8. Paste the following into **lab/network/main.tf** and save the changes

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

resource "google_compute_subnetwork" "lab_subnet" {
  name = "lab-subnet"
  ip_cidr_range = "10.0.1.0/24"
  region       = "us-east1"
  network = google_compute_network.lab_vpc.id
}

resource "google_compute_firewall" "default" {
  name          = "fw-allow-health-check"
  direction     = "INGRESS"
  network    = google_compute_network.lab_vpc.id
  priority      = 1000
  source_ranges = ["130.211.0.0/22", "35.191.0.0/16"]
  target_tags   = ["allow-health-check"]
  allow {
    ports    = ["80"]
    protocol = "tcp"
  }
}

resource "google_compute_global_address" "lab_lb_ip" {
  name       = "lb-ipv4-1"
  ip_version = "IPV4"
}

resource "google_compute_router" "lab_router" {
  name    = "lab-router"
  network = google_compute_network.lab_vpc.name
  region = "us-east1"
}

resource "google_compute_router_nat" "nat" {
  name                          = "my-router-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 = "ALL_SUBNETWORKS_ALL_IP_RANGES"
}
```

9. Paste the following into **lab/network/outputs.tf** and save the changes

```
output "vpc-id" {
  description = "ID of the VPC"
  value       = google_compute_network.lab_vpc.id
}

output "lab-subnet" {
  description = "id of lab subnet"
  value = google_compute_subnetwork.lab_subnet.id
}


output "lb-ip-id" {
  description = "ip id of lab load balancer"
  value =   google_compute_global_address.lab_lb_ip.id
}

output "lb-ip" {
  description = "ip of lab load balancer"
  value =   google_compute_global_address.lab_lb_ip.address
}
```

10. Paste the following into **lab/variables.tf** and save the changes

```
variable "lab-project" {
  default = "{your project id here}"
}

variable "lab-region" {
  default = "us-east1"
}
```

11. Paste the following into **lab/main.tf** and save the changes

```
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
      version = "4.77.0"
    }
  }
}

provider "google" {
  project = var.lab-project
  region  = var.lab-region
}

module "vpc" {
  source = "./network"
}
```

12. Update **var.lab-project** in **variables.tf,** replacing {your project id here} with the project id of your lab (retaining the quotes), then run

```
terraform init

terraform plan
```

13. Verify that there are no errors and then run

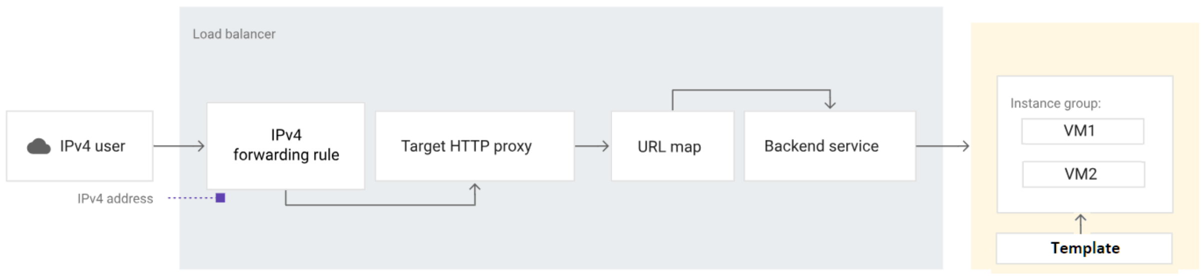
```
terraform apply
```

14. Enter yes to deploy.

15. Switch the console to review the deployment of a VPC, Subnet, Firewall Rule, Global public IP address, Router and NAT service.

The Challenge

Having been provided with a module that deploys all the necessary networking components, you are required to deploy the front-facing layer of an n-tier solution into this network stack. This will be an http web service hosted on linux gce instances. Multiple web instances are to be deployed to a single zone, and there should be a minimum of 2 instances running at any time, automatically increasing to a maximum of 5 should demand dictate. An autoscaling managed instance group should be used to achieve this. Traffic should be distributed across the running instances using a global external application load balancer. Successful completion of the challenge will be the generation of an http url which, when accessed, will display a message indicating from which instance the page is being served from. You are not to modify any supplied code and the provider version to be used is 4.77.0.



Understanding the traffic flow between these components in Google Cloud Load Balancing is crucial for designing and managing your load balancer setup. A high-level explanation of the traffic flow:

Forwarding Rule: The Global Forwarding Rule is the entry point for external traffic. It listens for incoming requests from users or clients. It defines the IP protocol, port range etc and maps incoming traffic to the appropriate target, which is a Target HTTP Proxy or Target HTTPS Proxy.

Target Proxy: The Target Proxy receives incoming traffic from the Global Forwarding Rule. It determines how the incoming requests are mapped to backend services using URL Maps and forwards the requests to the appropriate URL Map.

URL Map: The URL Map defines how incoming requests are matched to backend services based on rules and conditions. It can route requests based on URL path, host, header, query parameters, etc and determines which backend service will handle the request.

Backend Service: The Backend Service defines a group of instances that can handle incoming requests. It may also include health checks to ensure the instances are healthy. The Backend Service handles distributing requests to the instances based on its configuration (e.g., round-robin, session affinity, etc.).

Instance Group / Serverless NEG: The backend instances are organized into an Instance Group or a Serverless Network Endpoint Group (NEG). An Instance Group can contain VM instances, while a Serverless NEG can contain VMs or other resources. These groups are responsible for managing the instances and distributing traffic according to the load balancing algorithm configured in the Backend Service.

Instance / Backend Instance: Instances are the actual virtual machines or resources that handle incoming requests. They run applications and services that users interact with. The load balancer sends traffic to instances based on the rules defined in the Backend Service and the load balancing algorithm. After processing the incoming request, the instance sends its response back to the load balancer.

References

1. https://registry.terraform.io/providers/hashicorp/google/4.77.0/docs/resources/compute_instance_template.html

2. https://registry.terraform.io/providers/hashicorp/google/4.77.0/docs/resources/compute_instance_group_manager

3. https://registry.terraform.io/providers/hashicorp/google/4.77.0/docs/resources/compute_autoscaler

4. https://registry.terraform.io/providers/hashicorp/google/4.77.0/docs/resources/compute_health_check.html

5. https://registry.terraform.io/providers/hashicorp/google/4.77.0/docs/resources/compute_global_forwarding_rule.html

Try it yourself

You have the option of attempting this lab yourself with minimal guidance. Use the step-by-step guides and solution section at the end of this document if you need assistance.

The high-level lab goals are:

1. Using **template.tf,** create a **google_compute_instance_template.lab_instance_template** resource with the following settings...

   resource block name: lab_instance_template

   name: lab-launch-template

   size: e2-small

   instance: latest version of google cloud Debian linux image

   health checks: permit health checks to instances created from this template

   service account: use the default service account but with narrow scope

   instance metadata: apply the following startup script..

   "#! /bin/bash\n   sudo apt-get update\n   sudo apt-get install apache2 -y\n   sudo a2ensite default-ssl\n   sudo a2enmod ssl\n   vm_hostname=\"$(curl -H \"Metadata-Flavor:Google\" \\\n http://169.254.169.254/computeMetadata/v1/instance/name)\"\n   sudo echo \"Page served from: $vm_hostname\" | \\\n   tee /var/www/html/index.html\n   sudo systemctl restart apache2"

2. Using **managed-instance-group.tf,** create the following resources...

   google_compute_health_check.http_health_check

   google_compute_instance_group_manager.lb_backend_group

   google_compute_autoscaler.lab_autoscaler

3. The managed instance group should use your template, follow the challenge requirements, and reside in **us-east1-b**. The **google_compute_health_check.http-health-check** should check for port **80** responses.

4. Using loadbalancer.tf create the following resources...

   google_compute_backend_service.backend_service

google_compute_url_map.http_url_map

google_compute_target_http_proxy.http_proxy

google_compute_global_forwarding_rule.http_forwarding

5. The **google_compute_backend_service.backend_service** backend should be the managed instance group from step 2.

6. The **google_compute_global_forwarding_rule** should receive http traffic on the **lab-lb-ip** IP address created in the network module.

7. Using **outputs.tf**, output the IP address of the load balancer to your screen for testing.

Step-by-Step Guides

Task 1. Create Image Template

1. Open Ref 1 above and familiarize yourself with the resource type **compute_instance_template**

2. Our template will be used in conjunction with a managed instance group so scroll down to the **Using with Instance Group Manager** section.

3. Copy the example code into **template.tf**. For convenience the code is shown below..

```
resource "google_compute_instance_template" "instance_template" {
  name_prefix  = "instance-template-"
  machine_type = "e2-medium"
  region       = "us-central1"

  // boot disk
  disk {
    # ...
  }

  // networking
  network_interface {
    # ...
  }

  lifecycle {
    create_before_destroy = true
  }
}
```

4. The lifecyle argument is discussed at: https://developer.hashicorp.com/terraform/language/meta-arguments/lifecycle

5. Rename the resource block from "instance-template" to **"lab_instance_template"**

6. Replace the name_prefix "instance-template-" with **var.instance-prefix** Add this variable to **lab\variables.tf** with a default value of **"lab-template-"**
7. Replace the machine_type "e2-medium" with **var.machine-type**. Add this variable to **lab\variables.tf** with a default value of **"e2-small"**

8. Delete the `region` argument, because, reading the argument reference documentation regarding region, we see that if not specified, then the Provider region is used.

9. We need to add disk parameters for our template. Refer to the documentation for understanding and then paste the following code within the `disk` braces { }

```
boot           = true
source_image   = "projects/debian-cloud/global/images/family/debian-11"
mode           = "READ_WRITE"
type           = "PERSISTENT"
```

10. We need to add network interface parameters for our template. Refer to the documentation for understanding and then paste the following code within the `network_interface` braces { } Note the absence of an access_config sub-block, indicating we do not want the template to be given a public IP address.

```
network    = module.vpc.vpc-id
subnetwork = module.vpc.lab-subnet
```

11. A startup script has been provided that will install updates and Apache, configuring index.html to display a message that includes the vm name. Add the following code to the end of your resource block, inside the closing brace...

```
metadata = {

  startup-script = "#! /bin/bash\n    sudo apt-get update\n    sudo apt-get install apache2 -y\n    sudo a2ensite default-ssl\n    sudo a2enmod ssl\n    vm_hostname=\"$(curl -H \"Metadata-Flavor:Google\" \\\n http://169.254.169.254/computeMetadata/v1/instance/name)\"\n   sudo echo \"Page served from: $vm_hostname\" | \\\n   tee /var/www/html/index.html\n   sudo systemctl restart apache2"

}
```

12. This template will be used to create a managed instance group which will have health checks performed against them. Firewall rules must be applied to allow this and therefore we must use the tag already defined in the VPC firewall rules. Add the following block to the end of your resource block, inside the closing brace...

```
tags = ["allow-health-check"]
```

13. Save changes to template.tf

14. Run **terraform plan** Address any issues found, referring to the solution section if necessary.

15. Run **terraform apply** and use the Console to verify the deployment of the template.

Task 2. Create Managed Instance Group

1. Open Refs 2, 3 and 4 above to familiarize yourself with the architecture surrounding managed instance groups.

2. Focussing on Ref 2; https://registry.terraform.io/providers/hashicorp/google/4.77.0/docs/resources/compute_instance_group_manager, copy the **google_compute_health_check** example resource block into **managed-instance-group.tf** For convenience the code is shown below..

```
resource "google_compute_health_check" "autohealing" {
  name                = "autohealing-health-check"
  check_interval_sec  = 5
  timeout_sec         = 5
  healthy_threshold   = 2
  unhealthy_threshold = 10 # 50 seconds

  http_health_check {
    request_path = "/healthz"
    port         = "8080"
  }
}
```

3. Change the resource block name from "autohealing" to **"lab_health_check"**

4. Change the health check name from "autohealing-health-check" to **"lab-health-check"**

5. Reading the Argument reference, you will find that the next 4 lines are optional parameters which, if unspecified, adopt default values. For code brevity, delete these lines.

6. Change the http health check request path from "/healthz" to **"/"**

7. Change the http health check port from "8080" to **"80"**

8. The modified health check block should now read as shown below...

```
resource "google_compute_health_check" "lab_health_check" {
  name             = "lab-http-check"
  http_health_check {
    port             = 80
    request_path      = "/"
  }
}
```

9. Copy the **google_compute_instance_group_manager** resource block example code into the bottom of **managed-instance-group.tf**

10. For convenience the code is shown below..

```
resource "google_compute_instance_group_manager" "appserver" {
  name = "appserver-igm"

  base_instance_name = "app"
  zone             = "us-central1-a"

  version {
    instance_template  = google_compute_instance_template.appserver.self_link_unique
  }

  all_instances_config {
    metadata = {
      metadata_key = "metadata_value"
    }
    labels = {
      label_key = "label_value"
    }
  }

  target_pools = [google_compute_target_pool.appserver.id]
  target_size  = 2

  named_port {
    name = "customhttp"
    port = 8888
  }

  auto_healing_policies {
    health_check      = google_compute_health_check.autohealing.id
    initial_delay_sec = 300
  }
}
```

11. Change the resource block name from "appserver" to **"lab_mig"**

12. Change the name from "appserver-igm" to **"lab-mig"**

13. Change the base_instance_name from "app" to **"vm"**

14. Replace the zone value with variable `var.lab-zone`, declare this in **variables.tf** with a default value of **"us-east1-b"**

15. Change the instance template resource reference from **google_compute_instance_template.appserver.self_link_unique** to your template **google_compute_instance_template.lab_instance_template.self_link_unique**

16. Remove the **all_instances_config** block

17. Remove the **target_pools** argument (do not remove target_size argument)

18. Change the named_port name from "customhttp" to **"http"**

19. Change to port from 8888 to **80**

20. Change the health_check reference from **google_compute_health_check.autohealing.id** to your health check **google_compute_health_check.lab_health_check.id**

21. Change the initial delay to **120** seconds

22. The updated code should now be as shown below...

```
resource "google_compute_instance_group_manager" "lab_mig" {
  name = "lab-mig"

  base_instance_name = "vm"
  zone               = var.lab-zone

  version {
    instance_template  =
google_compute_instance_template.lab_instance_template.self_link_unique
  }

  target_size  = 2

  named_port {
    name = "http"
    port = 80
  }
```

```
 auto_healing_policies {
   health_check      = google_compute_health_check.lab_health_check.id
   initial_delay_sec = 120
 }
}
```

23. From Ref 3; https://registry.terraform.io/providers/hashicorp/google/4.77.0/docs/resources/compute_autoscaler, copy the **google_compute_autoscaler** resource block example from the **Example Usage - Autoscaler Basic** section and paste into the bottom of **managed-instance-group.tf** For convenience the code is shown below..

```
resource "google_compute_autoscaler" "example_autoscaler" {
 name           = "example-autoscaler"
 target         = google_compute_instance_group_manager.lb-backend-
group.instance_group
 zone = "us-east1-b"

 autoscaling_policy {
   min_replicas = 2
   max_replicas = 5

   load_balancing_utilization {
     target = 0.5
   }
 }
}
```

24. Change to resource block name from "example_autoscaler" to **"lab_autoscaler"**

25. Change the name from "example-autoscaler" to **"lab-autoscaler"**

26. Change the target reference from **google_compute_instance_group_manager.lb-backend-group.instance_group** to your instance group **google_compute_instance_group_manager.lab_mig.instance_group**

27. Change the zone to variable **var.lab-zone**

28. The updated code should be as shown below...

```
resource "google_compute_autoscaler" "lab_autoscaler" {
 name           = "lab-autoscaler"
 target         = google_compute_instance_group_manager.lab_mig.instance_group
 zone           = var.lab-zone

 autoscaling_policy {
   min_replicas = 2
```

```
    max_replicas = 5

    load_balancing_utilization {
      target = 0.5
    }
  }
}
```

29. Save changes to files and then **apply**.

30. Navigate to the Instance groups section of Compute Engine in the console. Select the **lab-mig** instance group to view the newly created instances. Wait for a minute or so and then refresh your browser to verify that the instances show as Healthy, indicating they have passed the health checks. Note the warning that there is no backend service attached to the instance group. This will be our next task.

Task 3. Create Load Balancer

1. At Ref 3; https://registry.terraform.io/providers/hashicorp/google/4.77.0/docs/resources/compute_global_forwarding_rule.html, navigate down to **Example Usage - Global Forwarding Rule External Managed**

2. Copy the **google_compute_global_forwarding_rule** resource example into **loadbalancer.tf**

3. For convenience the code is shown below ...

```
resource "google_compute_global_forwarding_rule" "default" {
  name                  = "global-rule"
  target                = google_compute_target_http_proxy.default.id
  port_range            = "80"
  load_balancing_scheme = "EXTERNAL_MANAGED"
}
```

4. Change resource block name from "default" to **"http_forwarding"**

5. Change the target reference from **google_compute_target_http_proxy.default.id** to **google_compute_target_http_proxy.http_proxy.id** (we will create this next)

6. Add argument **ip_protocol** with value of **"TCP"**

7. Add argument **ip_address** with value of **module.vpc.lb-ip-id**

8. Your modified code should be as shown below...

```
resource "google_compute_global_forwarding_rule" "http_forwarding" {
  name                  = "global-rule"
  target                = google_compute_target_http_proxy.http_proxy.id
  port_range            = "80"
  load_balancing_scheme = "EXTERNAL_MANAGED"
  ip_protocol           = "TCP"
  ip_address            = module.vpc.lb-ip-id
}
```

9. Copy the **google_compute_target_http_proxy** resource example into **loadbalancer.tf**

10. For convenience the code is shown below ...

```
resource "google_compute_target_http_proxy" "default" {
  name        = "target-proxy"
  description = "a description"
  url_map     = google_compute_url_map.default.id
}
```

11. Change the resource block name from "default" to **"http_proxy"**

12. Change the name from "target-proxy" to **"http-lb-proxy"**

13. Change the url_map reference from **google_compute_url_map.default.id** to **google_compute_url_map.http_url_map.id** (we will create this next)

14. Delete the optional description argument. Your code should now be as shown below...

```
resource "google_compute_target_http_proxy" "http_proxy" {
  name    = "http-lb-proxy"
  url_map = google_compute_url_map.http_url_map.id
}
```

15. Copy the **google_compute_url_map** resource example into **loadbalancer.tf**

16. For convenience the code is shown below ...

```
resource "google_compute_url_map" "default" {
  name            = "url-map-target-proxy"
  description     = "a description"
  default_service = google_compute_backend_service.default.id

  host_rule {
    hosts        = ["mysite.com"]
    path_matcher = "allpaths"
  }

  path_matcher {
    name            = "allpaths"
    default_service = google_compute_backend_service.default.id

    path_rule {
      paths   = ["/*"]
      service = google_compute_backend_service.default.id
    }
  }
}
```

17. Delete all arguments except name and default_service.

18. Change the resource block name from "default" to **"http_url_map"**

19. Change the default service from **google_compute_backend_service.default.id** to **google_compute_backend_service.backend_service.id** (we will create this next)

20. Your modified code should be as shown below...

```
resource "google_compute_url_map" "http_url_map" {
  name            = "url-map-target-proxy"
  default_service = google_compute_backend_service.backend_service.id
}
```

21. Copy the **google_compute_backend_service** resource example into **loadbalancer.tf**

22. For convenience the code is shown below ...

```
resource "google_compute_backend_service" "default" {
  name                  = "backend"
  port_name             = "http"
  protocol              = "HTTP"
  timeout_sec           = 10
  load_balancing_scheme = "EXTERNAL_MANAGED"
}
```

23. Change the resource block name from "default" to **"backend_service"**

24. Add the following lines...

```
 health_checks = [google_compute_health_check.lab_health_check.id]
  backend {
    group           = google_compute_instance_group_manager.lab_mig.instance_group
    balancing_mode  = "UTILIZATION"
    capacity_scaler = 1.0
  }
```

25. Your modified code should be as shown below...

```
resource "google_compute_backend_service" "backend_service" {
  name                      = "backend"
  port_name                 = "http"
  protocol                  = "HTTP"
  timeout_sec               = 10
  load_balancing_scheme     = "EXTERNAL_MANAGED"
  health_checks             = [google_compute_health_check.lab_health_check.id]
  backend {
    group         = google_compute_instance_group_manager.lab_mig.instance_group
    balancing_mode  = "UTILIZATION"
    capacity_scaler = 1.0
  }
}
```

26. Add the following code to lab\outputs to output the load balance IP address to screen...

```
output "lb-ip" {
  description = "ip of lab load balancer"
  value =   module.vpc.lb-ip
}
```

27. Ensure all changed files are saved.

28. To test the complete deployment, first destroy all currently deployed resources using terraform destroy This may take 5 minutes or so to complete.

29. Use terraform apply to deploy all 14 resources.

30. Note the IP address of the load balancer. Open a local browser tab and navigate to this address. You will receive error messages for a while whilst the load balancer is deployed globally. Refresh periodically until you are shown the name of the backend vm you are being connected to; this may take 3-4 minutes. Refresh your browser to verify load balancing across the 2 backend instances.

31. You do not need to destroy your resources, simply end the lab via qwiklabs as this will destroy the entire project and all resources.

Solution

template.tf

```
resource "google_compute_instance_template" "lab_instance_template" {
  name_prefix = var.instance-prefix
  machine_type = var.machine-type
  metadata = {
    startup-script = "#! /bin/bash\n     sudo apt-get update\n     sudo apt-get install apache2 -y\n
sudo a2ensite default-ssl\n     sudo a2enmod ssl\n     vm_hostname=\"$(curl -H \"Metadata-
Flavor:Google\" \\\n   http://169.254.169.254/computeMetadata/v1/instance/name)\"\n   sudo
echo \"Page served from: $vm_hostname\" | \\\n   tee /var/www/html/index.html\n   sudo
systemctl restart apache2"
  }
  disk {
    boot         = true
    source_image = "projects/debian-cloud/global/images/family/debian-11"
    mode         = "READ_WRITE"
    type         = "PERSISTENT"
  }

  network_interface {
    network    = module.vpc.vpc-id
    subnetwork = module.vpc.lab-subnet
  }

  lifecycle {
    create_before_destroy = true
  }

  service_account {
    email  = "default"
    scopes = ["https://www.googleapis.com/auth/devstorage.read_only",
"https://www.googleapis.com/auth/logging.write",
"https://www.googleapis.com/auth/monitoring.write",
"https://www.googleapis.com/auth/pubsub",
"https://www.googleapis.com/auth/service.management.readonly",
"https://www.googleapis.com/auth/servicecontrol",
"https://www.googleapis.com/auth/trace.append"]
  }
  tags = ["allow-health-check"]
}
```

managed-instance-group.tf

```
resource "google_compute_health_check" "lab_health_check" {
  name              = "lab-http-check"
  http_health_check {
    port              = 80
    request_path       = "/"
  }
}

resource "google_compute_instance_group_manager" "lab_mig" {
  name = "lab-mig"

  base_instance_name = "vm"
  zone               = var.lab-zone
  version {
    instance_template  =
google_compute_instance_template.lab_instance_template.self_link_unique
  }

  target_size  = 2

  named_port {
    name = "http"
    port = 80
  }

  auto_healing_policies {
    health_check      = google_compute_health_check.lab_health_check.id
    initial_delay_sec = 120
  }
}

resource "google_compute_autoscaler" "lab_autoscaler" {
  name             = "lab-autoscaler"
  target           = google_compute_instance_group_manager.lab_mig.instance_group
  zone = var.lab-zone

  autoscaling_policy {
    min_replicas = 2
    max_replicas = 5

    load_balancing_utilization {
      target = 0.5
    }
  }
}
```

load-balancer.tf

```
resource "google_compute_global_forwarding_rule" "http_forwarding" {
  name                  = "global-rule"
  target                = google_compute_target_http_proxy.http_proxy.id
  port_range            = "80"
  load_balancing_scheme = "EXTERNAL_MANAGED"
  ip_protocol           = "TCP"
  ip_address            = module.vpc.lb-ip-id
}

resource "google_compute_target_http_proxy" "http_proxy" {
  name    = "http-lb-proxy"
  url_map = google_compute_url_map.http_url_map.id
}

resource "google_compute_url_map" "http_url_map" {
  name            = "url-map-target-proxy"
  default_service = google_compute_backend_service.backend_service.id
}

resource "google_compute_backend_service" "backend_service" {
  name                            = "backend"
  port_name                       = "http"
  protocol                        = "HTTP"
  timeout_sec                     = 10
  load_balancing_scheme           = "EXTERNAL_MANAGED"
  health_checks                   = [google_compute_health_check.lab_health_check.id]
  backend {
    group           = google_compute_instance_group_manager.lab_mig.instance_group
    balancing_mode  = "UTILIZATION"
    capacity_scaler = 1.0
  }
}
```

variables.tf

```
variable "lab-project" {
  default = "terraform-2697"
}

variable "lab-region" {
  default = "us-east1"
}

variable "instance-prefix" {
  default = "lab-template-"
}

variable "machine-type" {
  default = "e2-small"
}

variable "lab-zone" {
  default = "us-east1-b"
}
```

outputs.tf

```
output "lb_address" {
  value = module.vpc.lb-ip
}
```