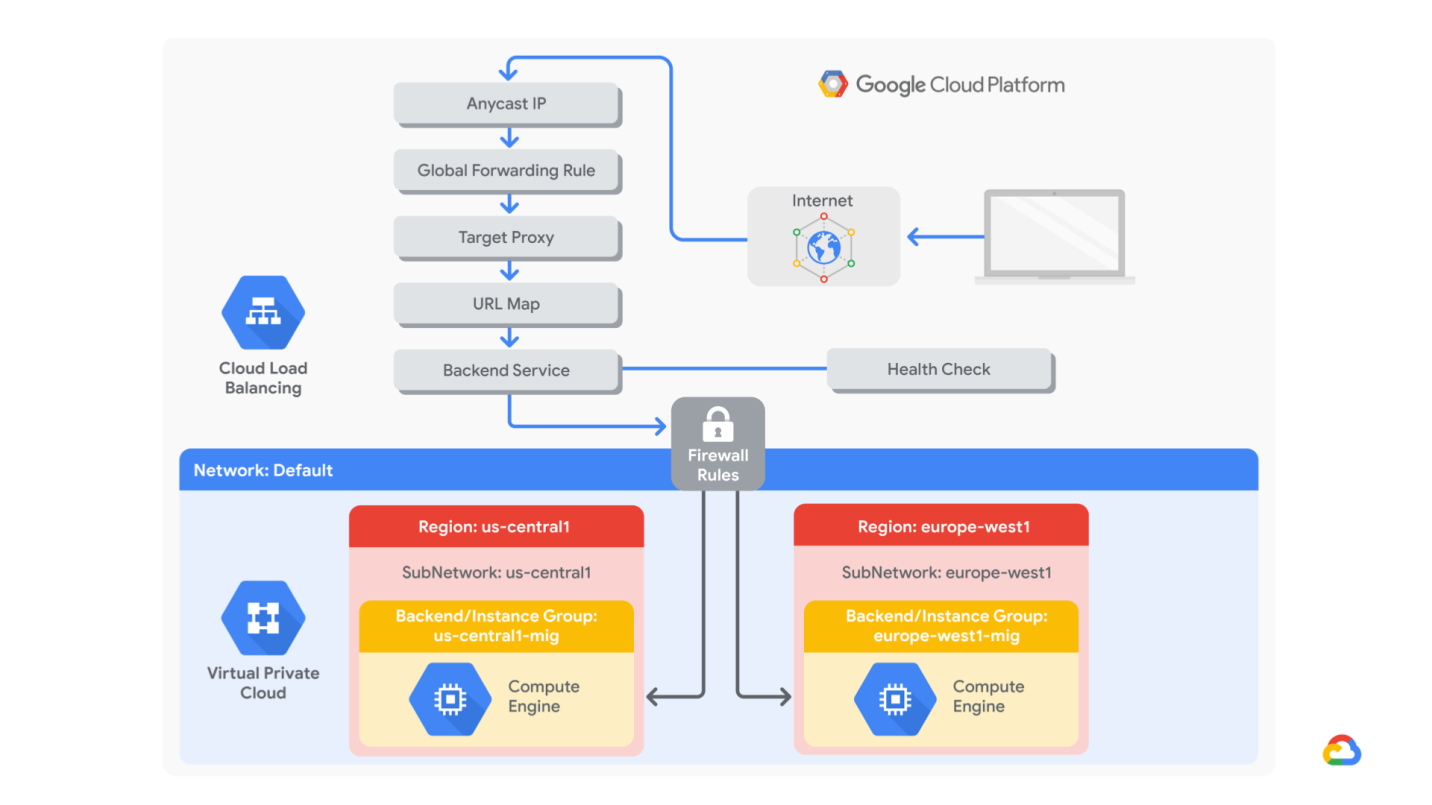
**Configuring an HTTP Load Balancer with Autoscaling**

**Overview**

Google Cloud HTTP(S) load balancing is implemented at the edge of Google's network in Google's points of presence (POP) around the world. User traffic directed to an HTTP(S) load balancer enters the POP closest to the user and is then load-balanced over Google's global network to the closest backend that has sufficient available capacity.

In this lab, you configure an HTTP load balancer as shown in the diagram below. Then, you stress test the load balancer to demonstrate global load balancing and autoscaling.



Objectives

In this lab, you learn how to perform the following tasks:

Create a health check firewall rule

Create a NAT configuration using Cloud Router

Create a custom image for a web server

Create an instance template based on the custom image

Create two managed instance groups

Configure an HTTP load balancer with IPv4 and IPv6

Stress test an HTTP load balancer

**Task 1. Configure a health check firewall rule**

Health checks determine which instances of a load balancer can receive new connections. For HTTP load balancing, the health check probes to your load-balanced instances come from addresses in the ranges **130.211.0.0/22** and **35.191.0.0/16**. Your firewall rules must allow these connections.

**Create the health check rule**

Create a firewall rule to allow health checks:

gcloud compute --project=qwiklabs-gcp-ac8ed9421b0d139d firewall-rules create fw-allow-health-checks --direction=INGRESS --priority=1000 --network=default --action=ALLOW --rules=tcp:80 --source-ranges=130.211.0.0/22,35.191.0.0/16 --target-tags=allow-health-checks

**Task 2: Create a NAT configuration using Cloud Router**

The Google Cloud VM backend instances that you setup in Task 3 will not be configured with external IP addresses.

Instead, you will setup the Cloud NAT service to allow these VM instances to receive incoming traffic only through the Cloud NAT, and return traffic from the outbound NAT through the load balancer.

Create the Cloud Router instance

In the Cloud Console, on the **Navigation menu** (Navigation menu), click **Network services** > **Cloud NAT**.

Click **Get started**.

Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (type value or select option as specified)** |
| Gateway name | nat-config |
| Region | us-central1 |

Click **Cloud Router**, and select **Create new router**.

For **Name**, type **nat-router-us-central1**.

Click **Create**.

In Create a NAT gateway, click **Create**.

Wait until the NAT Gateway Status changes to Running before moving onto the next task.

**Task 3: Create a custom image for a web server**

Create a custom web server image for the backend of the load balancer.

Create a VM

gcloud beta compute --project=qwiklabs-gcp-ac8ed9421b0d139d instances create webserver --zone=us-central1-a --machine-type=f1-micro --subnet=default --no-address --maintenance-policy=MIGRATE --service-account=438000131250-compute@developer.gserviceaccount.com --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/servicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleapis.com/auth/trace.append --tags=allow-health-checks --image=debian-10-buster-v20200902 --image-project=debian-cloud --boot-disk-size=10GB --boot-disk-type=pd-standard --boot-disk-device-name=webserver --no-shielded-secure-boot --no-shielded-vtpm --no-shielded-integrity-monitoring --reservation-affinity=any

Customize the VM

For **webserver**, click **SSH** to launch a terminal and connect.

If you see the **Connection via Cloud Identity-Aware Proxy Failed** popup, click **Retry**.

To install Apache2, run the following commands:

sudo apt-get update

sudo apt-get install -y apache2

To start the Apache server, run the following command:

sudo service apache2 start

To test the default page for the Apache2 server, run the following command:

curl localhost

The default page for the Apache2 server should be displayed.

Set the Apache service to start at boot

The software installation was successful. However, when a new VM is created using this image, the freshly booted VM does not have the Apache web server running. Use the following command to set the Apache service to automatically start on boot. Then test it to make sure it works.

In the webserver SSH terminal, set the service to start on boot:

sudo update-rc.d apache2 enable

In the Cloud Console, select **webserver**, and then click **Reset.**

In the confirmation dialog, click **Reset**.

Reset will stop and reboot the machine. It keeps the same IPs and the same persistent boot disk, but memory is wiped. Therefore, if the Apache service is available after the reset, the **update-rc** command was successful.

Check the server by connecting via SSH to the VM and entering the following command:

sudo service apache2 status

**NOTE:** If you see the **Connection via Cloud Identity-Aware Proxy Failed** popup, click **Retry** .

The result should show **Started The Apache HTTP Server**.

Prepare the disk to create a custom image

Verify that the boot disk will not be deleted when the instance is deleted.

On the VM instances page, click **webserver** to view the VM instance details.

Under **Boot disk**, verify that **When deleting instance** is set to **Keep disk**.

Return to the VM instances page, click **webserver**, and click **Delete.**

In the confirmation dialog, click **Delete**.

In the left pane, click **Disks** and verify that the **webserver** disk exists.

Create the custom image

gcloud compute images create my-web-server --project=qwiklabs-gcp-ac8ed9421b0d139d --source-disk=webserver --source-disk-zone=us-central1-a --storage-location=us

You have created a custom image that multiple identical webservers can be started from. At this point, you could delete the **webserver** disk.

The next step is to use that image to define an instance template that can be used in the managed instance groups.

**Task 4. Configure an instance template and create instance groups**

A managed instance group uses an instance template to create a group of identical instances. Use these to create the backends of the HTTP load balancer.

**Configure the instance template**

An instance template is an API resource that you can use to create VM instances and managed instance groups. Instance templates define the machine type, boot disk image, subnet, labels, and other instance properties.

gcloud beta compute --project=qwiklabs-gcp-ac8ed9421b0d139d instance-templates create mywebserver-template --machine-type=f1-micro --network=projects/qwiklabs-gcp-ac8ed9421b0d139d/global/networks/default --no-address --maintenance-policy=MIGRATE --service-account=438000131250-compute@developer.gserviceaccount.com --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/servicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleapis.com/auth/trace.append --tags=allow-health-checks --image=my-web-server --image-project=qwiklabs-gcp-ac8ed9421b0d139d --boot-disk-size=10GB --boot-disk-type=pd-standard --boot-disk-device-name=mywebserver-template --no-shielded-secure-boot --shielded-vtpm --shielded-integrity-monitoring --reservation-affinity=any

**Create the managed instance groups**

Create a managed instance group in **us-central1**

gcloud compute --project "qwiklabs-gcp-ac8ed9421b0d139d" health-checks create tcp "http-health-check" --timeout "5" --check-interval "10" --unhealthy-threshold "3" --healthy-threshold "2" --port "80"

gcloud beta compute --project=qwiklabs-gcp-ac8ed9421b0d139d instance-groups managed create us-central1-mig --base-instance-name=us-central1-mig --template=mywebserver-template --size=1 --zones=us-central1-b,us-central1-c,us-central1-f --instance-redistribution-type=PROACTIVE --health-check=http-health-check --initial-delay=60

gcloud beta compute --project "qwiklabs-gcp-ac8ed9421b0d139d" instance-groups managed set-autoscaling "us-central1-mig" --region "us-central1" --cool-down-period "60" --max-num-replicas "2" --min-num-replicas "1" --target-load-balancing-utilization "0.8" --mode "on"

Create a managed instance group in **europe-west1**.

cloud beta compute --project=qwiklabs-gcp-ac8ed9421b0d139d instance-groups managed create europe-west1-mig --base-instance-name=europe-west1-mig --template=mywebserver-template --size=1 --zones=europe-west1-b,europe-west1-c,europe-west1-d --instance-redistribution-type=PROACTIVE

gcloud beta compute --project "qwiklabs-gcp-ac8ed9421b0d139d" instance-groups managed set-autoscaling "europe-west1-mig" --region "europe-west1" --cool-down-period "60" --max-num-replicas "2" --min-num-replicas "1" --target-load-balancing-utilization "0.8" --mode "on"

Managed instance groups offer **autoscaling** capabilities that allow you to automatically add or remove instances from a managed instance group based on increases or decreases in load. Autoscaling helps your applications gracefully handle increases in traffic and reduces cost when the need for resources is lower. You just define the autoscaling policy, and the autoscaler performs automatic scaling based on the measured load.

For **Health check**, select **Create a health check**.

Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (select option as specified)** |
| Name | http-health-check |
| Protocol | TCP |
| Port | 80 |

Managed instance group health checks proactively signal to delete and recreate instances that become unhealthy.

Click **Save and continue**.

For **Initial delay**, type 60. This is how long the Instance Group waits after initializing the boot-up of a VM before it tries a health check. You don't want to wait 5 minutes for this during the lab, so you set it to 1 minute.

Click **Create**.

Click **OK**.

**NOTE:** If a warning window will appear stating that **There is no backend service attached to the instance group**. Ignore this; you will configure the load balancer with a backend service in the next section of the lab.

Repeat the same procedure for **europe-west1-mig** in **europe-west1**:

Click **Create Instance group**.

Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (type value or select option as specified)** |
| Name | europe-west1-mig |
| Location | Multiple zones |
| Region | europe-west1 |
| Instance template | mywebserver-template |
| Autoscaling metrics > Metric Type | HTTP load balancing utilization |
| Target HTTP load balancing utilization | 80 |
| Minimum number of instances | 1 |
| Maximum number of instances | 2 |
| Cool down period | 60 |

For **Health check**, select **http-health-check (TCP)**.

For **Initial delay**, type 60.

Click **Create**.

Click **OK** in the dialog window.

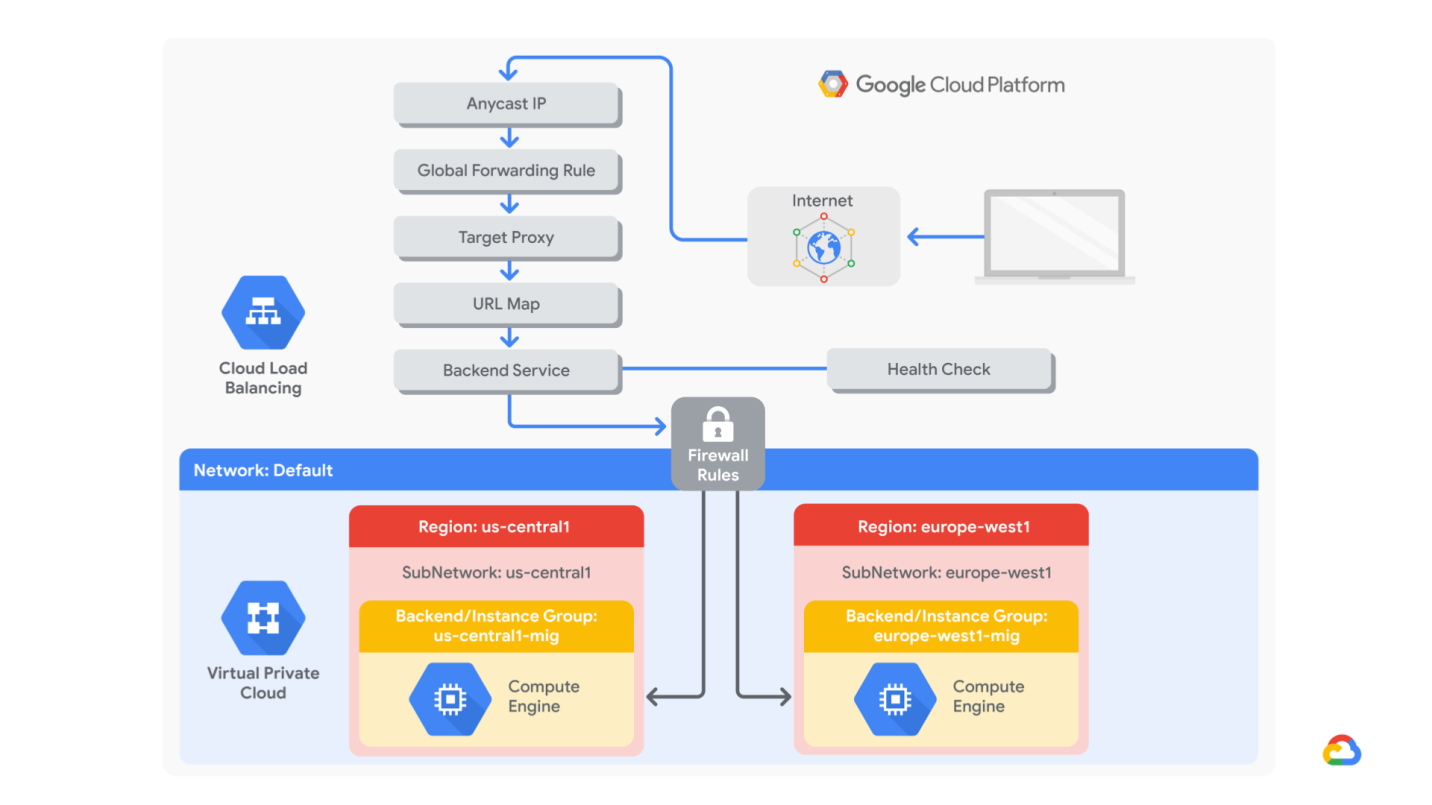
**Verify the backends**

Verify that VM instances are being created in both regions.

On the **Navigation menu**, click **Compute Engine** > **VM instances**. Notice the instances that start with *us-central1-mig* and *europe-west1-mig*. These instances are part of the managed instance groups.

**Task 5. Configure the HTTP load balancer**

Configure the HTTP load balancer to balance traffic between the two backends (**us-central1-mig** in us-central1 and **europe-west1-mig** in europe-west1) as illustrated in the network diagram:



**Start the configuration**

On the **Navigation menu**, click **Network Services** > **Load balancing**.

Click **Create load balancer**.

Under **HTTP(S) Load Balancing**, click **Start configuration**.

Select **From Internet to my VMs**, then click **Continue**.

For **Name**, type **http-lb**.

**Configure the backend**

Backend services direct incoming traffic to one or more attached backends. Each backend is composed of an instance group and additional serving capacity metadata.

gcloud beta compute --project=qwiklabs-gcp-03-3bf6948e3787 instance-templates create instance-template-1 --machine-type=e2-medium --subnet=projects/qwiklabs-gcp-03-3bf6948e3787/regions/us-central1/subnetworks/subnet-a --no-address --metadata=startup-script-url=gs://cloud-training/gcpnet/ilb/startup.sh --maintenance-policy=MIGRATE --service-account=412584802085-compute@developer.gserviceaccount.com --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/servicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleapis.com/auth/trace.append --region=us-central1 --tags=backend-service --image=debian-10-buster-v20200910 --image-project=debian-cloud --boot-disk-size=10GB --boot-disk-type=pd-standard --boot-disk-device-name=instance-template-1 --no-shielded-secure-boot --no-shielded-vtpm --no-shielded-integrity-monitoring --reservation-affinity=any

**Configure the frontend**

The host and path rules determine how your traffic will be directed. For example, you could direct video traffic to one backend and direct static traffic to another backend. However, you are not configuring the host and path rules in this lab.

Click **Frontend configuration**.

Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (type value or select option as specified)** |
| Protocol | HTTP |
| IP version | IPv4 |
| IP address | Ephemeral |
| Port | 80 |

Click **Done**.

Click **Add Frontend IP and port**.

Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (type value or select option as specified)** |
| Protocol | HTTP |
| IP version | IPv6 |
| IP address | Ephemeral |
| Port | 80 |

Click **Done**.

HTTP(S) load balancing supports both IPv4 and IPv6 addresses for client traffic. Client IPv6 requests are terminated at the global load balancing layer and then proxied over IPv4 to your backends.

**Review and create the HTTP load balancer**

Click **Review and finalize**.

Review the **Backend services** and **Frontend**.

Click **Create**. Wait for the load balancer to be created.

Click on the name of the load balancer (**http-lb**).

Note the IPv4 and IPv6 addresses of the load balancer for the next task. They will be referred to as [LB\_IP\_v4] and [LB\_IP\_v6], respectively.

The IPv6 address is the one in hexadecimal format.

**Access the HTTP load balancer**

Open a new tab in your browser and navigate to http://[LB\_IP\_v4]. Make sure to replace [LB\_IP\_v4] with the IPv4 address of the load balancer.

Accessing the HTTP load balancer might take a couple of minutes. In the meantime, you might get a 404 or 502 error. Keep trying until you see the page of one of the backends.

**Stress test the HTTP load balancer**

Create a new VM to simulate a load on the HTTP load balancer. Then determine whether traffic is balanced across both backends when the load is high.

In the Cloud Console, on the **Navigation menu** (Navigation menu), click **Compute Engine** > **VM instances**.

Click **Create instance**.

Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (type value or select option as specified)** |
| Name | stress-test |
| Region | us-west1 |
| Zone | us-west1-c |
| Machine type | f1-micro (1 vCPU) |

Because **us-west1** is closer to **us-central1** than to **europe-west1**, traffic should be forwarded only to **us-central1-mig** (unless the load is too high).

For **Boot Disk**, click **Change**.

Click **Custom images**.

For **Image**, select **mywebserver**.

Click **Select**.

Click **Create**. Wait for the **stress-test** instance to be created.

For **stress-test**, click **SSH** to launch a terminal and connect.

To create an environment variable for your load balancer IP address, run the following command:

export LB\_IP=<Enter your [LB\_IP\_v4] here>

Verify it with echo:

echo $LB\_IP

To place a load on the load balancer, run the following command:

ab -n 500000 -c 1000 http://$LB\_IP/

In the Cloud Console, on the **Navigation menu** (Navigation menu), click **Network Services** > **Load balancing**.

Click **Backends**.

Click **http-backend**.

Monitor the **Frontend Location (Total inbound traffic)** between North America and the two backends for a couple of minutes.

At first, traffic should just be directed to **us-central1-mig**, but as the RPS increases, traffic is also directed to **europe-west1-mig**. This demonstrates that by default traffic is forwarded to the closest backend, but if the load is very high, traffic can be distributed across the backends.

In the Cloud Console, on the **Navigation menu** (Navigation menu), click **Compute Engine** > **Instance groups**.

Click on **us-central1-mig** to open the instance group page.

Click **Monitoring** to monitor the number of instances and LB capacity.

Repeat the same for the **europe-west1-mig** instance group.

Depending on the load, you might see the backends scale to accommodate the load.

**Task 7. Review**

In this lab, you configured an HTTP load balancer with backends in us-central1 and europe-west1. Then you stress-tested the load balancer with a VM to demonstrate global load balancing and autoscaling.

End of lab