

Set Up Network and HTTP Load Balancers | Google Cloud Skills Boost

Qwiklabs : 17-21 minutes

GSP007



Google Cloud Self-Paced Labs

Overview

In this hands-on lab you'll learn the differences between a network load balancer and an HTTP load balancer and how to set them up for your applications running on Compute Engine virtual machines (VMs).

There are several ways you can [load balance on Google Cloud](#). This lab takes you through the set up of the following load balancers:

- [Network Load Balancer](#)
- [HTTP\(s\) Load Balancer](#)

You are encouraged to type the commands yourself, which can help you learn the core concepts. Many labs include a code block that contains the required commands. You can easily copy and paste the commands from the code block into the appropriate places during the lab.

What you'll learn

- Set up a network load balancer.
- Set up an HTTP load balancer.
- Get hands-on experience learning the differences between network load balancers and HTTP load balancers.

Setup and requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab---remember, once you start, you cannot pause a lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:
 - The **Open Google Console** button
 - Time remaining
 - The temporary credentials that you must use for this lab
 - Other information, if needed, to step through this lab

2. Click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.

Tip: Arrange the tabs in separate windows, side-by-side.

Note: If you see the **Choose an account** dialog, click **Use Another Account**.

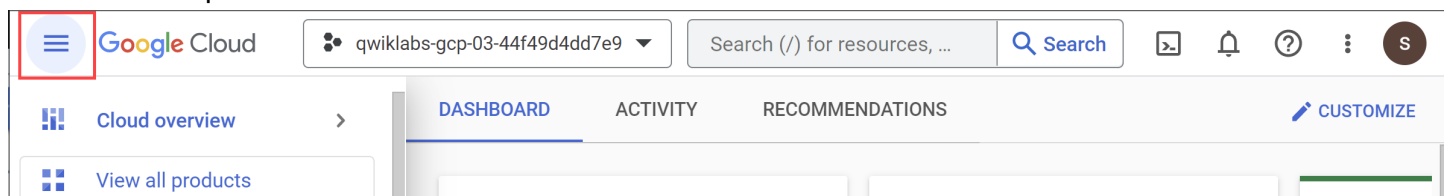
3. If necessary, copy the **Username** from the **Lab Details** panel and paste it into the **Sign in** dialog. Click **Next**.
4. Copy the **Password** from the **Lab Details** panel and paste it into the **Welcome** dialog. Click **Next**.

Important: You must use the credentials from the left panel. Do not use your Google Cloud Skills Boost credentials. **Note:** Using your own Google Cloud account for this lab may incur extra charges.

5. Click through the subsequent pages:
 - Accept the terms and conditions.
 - Do not add recovery options or two-factor authentication (because this is a temporary account).
 - Do not sign up for free trials.


After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell**  at the top of the Google Cloud console.

When you are connected, you are already authenticated, and the project is set to your **PROJECT_ID**. The output contains a line that declares the **PROJECT_ID** for this session:

Your Cloud Platform project in this session is set to YOUR_PROJECT_ID

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

2. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

3. Click **Authorize**.

4. Your output should now look like this:

Output:

```
ACTIVE: * ACCOUNT: student-01-xxxxxxxxxxxx@qwiklabs.net To set the active account, run: $ gcloud
config set account `ACCOUNT`
```

5. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

Output:

```
[core] project = <project_ID>
```

Example output:

```
[core] project = qwiklabs-gcp-44776a13dea667a6 Note: For full documentation of gcloud, in Google Cloud,
refer to the gcloud CLI overview guide.
```

Task 1. Set the default region and zone for all resources

1. In Cloud Shell, set the default zone:

```
gcloud config set compute/zone {{{project_0.startup_script.project_zone}}}
```

2. Set the default region:

```
gcloud config set compute/region {{{project_0.startup_script.project_region}}}
```

Learn more about choosing zones and regions in Compute Engine's documentation [Regions and Zones Guide](#).

Task 2. Create multiple web server instances

For this load balancing scenario, create three Compute Engine VM instances and install Apache on them, then add a firewall rule that allows HTTP traffic to reach the instances.

The code provided sets the zone to . Setting the tags field lets you reference these instances all at once, such as with a firewall rule. These commands also install Apache on each instance and give each instance a unique home page.

1. Create a virtual machine `www1` in your default zone.

```
gcloud compute instances create www1 \ --zone={{project_0.startup_script.project_zone}} \ --tags=network-lb-tag \ --machine-type=e2-small \ --image-family=debian-11 \ --image-project=debian-cloud \ --metadata=startup-script='#!/bin/bash apt-get update apt-get install apache2 -y service apache2 restart echo "
```

Web Server: `www1`

```
" | tee /var/www/html/index.html'
```

2. Create a virtual machine `www2` in your default zone.

```
gcloud compute instances create www2 \ --zone={{project_0.startup_script.project_zone}} \ --tags=network-lb-tag \ --machine-type=e2-small \ --image-family=debian-11 \ --image-project=debian-cloud \ --metadata=startup-script='#!/bin/bash apt-get update apt-get install apache2 -y service apache2 restart echo "
```

Web Server: `www2`

```
" | tee /var/www/html/index.html'
```

3. Create a virtual machine `www3` in your default zone.

```
gcloud compute instances create www3 \ --zone={{project_0.startup_script.project_zone}} \ --tags=network-lb-tag \ --machine-type=e2-small \ --image-family=debian-11 \ --image-project=debian-cloud \ --metadata=startup-script='#!/bin/bash apt-get update apt-get install apache2 -y service apache2 restart echo "
```

Web Server: `www3`

```
" | tee /var/www/html/index.html'
```

4. Create a firewall rule to allow external traffic to the VM instances:

```
gcloud compute firewall-rules create www-firewall-network-lb \ --target-tags network-lb-tag --allow tcp:80
```

Now you need to get the external IP addresses of your instances and verify that they are running.

5. Run the following to list your instances. You'll see their IP addresses in the `EXTERNAL_IP` column:

```
gcloud compute instances list
```

6. Verify that each instance is running with `curl`, replacing `[IP_ADDRESS]` with the IP address for each of your VMs:

```
curl http://[IP_ADDRESS]
```

Click **Check my progress** below to verify that you've created a group of web servers.

Create multiple web server instances

Task 3. Configure the load balancing service

When you configure the load balancing service, your virtual machine instances will receive packets that are destined for the static external IP address you configure. Instances made with a Compute Engine image are automatically configured to handle this IP address.

Note: Learn more about how to set up network load balancing from the [External TCP/UDP Network Load Balancing overview Guide](#).

1. Create a static external IP address for your load balancer:

```
gcloud compute addresses create network-lb-ip-1 \ --region  
{{{project_0.startup_script.project_region}}}
```

Output:

```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-  
xxxxxxxxxxxx/regions/{{{project_0.startup_script.project_region}}}/addresses/network-lb-ip-1].
```

2. Add a legacy HTTP health check resource:

```
gcloud compute http-health-checks create basic-check
```

3. Add a target pool in the same region as your instances. Run the following to create the target pool and use the health check, which is required for the service to function:

```
gcloud compute target-pools create www-pool \ --region {{{project_0.startup_script.project_region}}} --  
http-health-check basic-check
```

4. Add the instances to the pool:

```
gcloud compute target-pools add-instances www-pool \ --instances www1,www2,www3
```

5. Add a forwarding rule:

```
gcloud compute forwarding-rules create www-rule \ --region  
{{{project_0.startup_script.project_region}}} \ --ports 80 \ --address network-lb-ip-1 \ --target-pool www-  
pool
```

Click **Check my progress** below to verify that you've created an L4 network load balancer that points to the web servers.

Configure the load balancing service

Task 4. Sending traffic to your instances

Now that the load balancing service is configured, you can start sending traffic to the forwarding rule and watch the traffic be dispersed to different instances.

1. Enter the following command to view the external IP address of the www-rule forwarding rule used by the load balancer:

```
gcloud compute forwarding-rules describe www-rule --region  
{{{project_0.startup_script.project_region}}}
```

2. Access the external IP address

```
IPADDRESS=$(gcloud compute forwarding-rules describe www-rule --region  
{{{project_0.startup_script.project_region}}} --format="json" | jq -r .IPAddress)
```

3. Show the external IP address

```
echo $IPADDRESS
```

4. Use `curl` command to access the external IP address, replacing `IP_ADDRESS` with an external IP address from the previous command:

```
while true; do curl -m1 $IPADDRESS; done
```

The response from the `curl` command alternates randomly among the three instances. If your response is initially unsuccessful, wait approximately 30 seconds for the configuration to be fully loaded and for your instances to be marked healthy before trying again.

5. Use **Ctrl + c** to stop running the command.

Task 5. Create an HTTP load balancer

HTTP(S) Load Balancing is implemented on Google Front End (GFE). GFEs are distributed globally and operate together using Google's global network and control plane. You can configure URL rules to route some URLs to one set of instances and route other URLs to other instances.

Requests are always routed to the instance group that is closest to the user, if that group has enough capacity and is appropriate for the request. If the closest group does not have enough capacity, the request is sent to the closest group that **does** have capacity.

To set up a load balancer with a Compute Engine backend, your VMs need to be in an instance group. The managed instance group provides VMs running the backend servers of an external HTTP load balancer. For this lab, backends serve their own hostnames.

1. First, create the load balancer template:

```
gcloud compute instance-templates create lb-backend-template \ --region=
{{{project_0.startup_script.project_region}}} \ --network=default \ --subnet=default \ --tags=allow-
health-check \ --machine-type=e2-medium \ --image-family=debian-11 \ --image-project=debian-cloud
\ --metadata=startup-script='#!/bin/bash apt-get update apt-get install apache2 -y a2ensite default-ssl
a2enmod ssl vm_hostname="$(curl -H "Metadata-Flavor:Google" \
http://169.254.169.254/computeMetadata/v1/instance/name)" echo "Page served from:
$vm_hostname" | tee /var/www/html/index.html systemctl restart apache2'
```

[Managed instance groups](#) (MIGs) let you operate apps on multiple identical VMs. You can make your workloads scalable and highly available by taking advantage of automated MIG services, including: autoscaling, autohealing, regional (multiple zone) deployment, and automatic updating.

2. Create a managed instance group based on the template:

```
gcloud compute instance-groups managed create lb-backend-group \ --template=lb-backend-template
--size=2 --zone={{project_0.startup_script.project_zone}}
```

3. Create the fw-allow-health-check firewall rule.

```
gcloud compute firewall-rules create fw-allow-health-check \ --network=default \ --action=allow \ --
direction=ingress \ --source-ranges=130.211.0.0/22,35.191.0.0/16 \ --target-tags=allow-health-check \
--rules=tcp:80 Note: The ingress rule allows traffic from the Google Cloud health checking systems
(130.211.0.0/22 and 35.191.0.0/16). This lab uses the target tag allow-health-check to identify
the VMs
```

4. Now that the instances are up and running, set up a global static external IP address that your customers use to reach your load balancer:

```
gcloud compute addresses create lb-ipv4-1 \ --ip-version=IPV4 \ --global
```

Note the IPv4 address that was reserved:

```
gcloud compute addresses describe lb-ipv4-1 \ --format="get(address)" \ --global
```

5. Create a health check for the load balancer:

```
gcloud compute health-checks create http http-basic-check \ --port 80 Note: Google Cloud provides
health checking mechanisms that determine whether backend instances respond properly to traffic.
For more information, please refer to the Creating health checks document.
```

6. Create a backend service:

```
gcloud compute backend-services create web-backend-service \ --protocol=HTTP \ --port-name=http \
--health-checks=http-basic-check \ --global
```

7. Add your instance group as the backend to the backend service:

```
gcloud compute backend-services add-backend web-backend-service \ --instance-group=lb-backend-
group \ --instance-group-zone={{project_0.startup_script.project_zone}}} \ --global
```

8. Create a [URL map](#) to route the incoming requests to the default backend service:

`gcloud compute url-maps create web-map-http \ --default-service web-backend-service` **Note:** URL map is a Google Cloud configuration resource used to route requests to backend services or backend buckets. For example, with an external HTTP(S) load balancer, you can use a single URL map to route requests to different destinations based on the rules configured in the URL map:

- Requests for `https://example.com/video` go to one backend service.
- Requests for `https://example.com/audio` go to a different backend service.
- Requests for `https://example.com/images` go to a Cloud Storage backend bucket.
- Requests for any other host and path combination go to a default backend service.

9. Create a target HTTP proxy to route requests to your URL map:

```
gcloud compute target-http-proxies create http-lb-proxy \ --url-map web-map-http
```

10. Create a global forwarding rule to route incoming requests to the proxy:

```
gcloud compute forwarding-rules create http-content-rule \ --address=lb-ipv4-1 \ --global \ --target-http-proxy=http-lb-proxy \ --ports=80
```

Note: A [forwarding rule](#) and its corresponding IP address represent the frontend configuration of a Google Cloud load balancer. Learn more about the general understanding of forwarding rules from the [Forwarding rule overview Guide](#).

Click **Check my progress** below to verify that you've created an L7 HTTP(S) load balancer.

Create an HTTP load balancer

Task 6. Testing traffic sent to your instances

1. In the Cloud Console, from the **Navigation menu**, go to **Network services > Load balancing**.
2. Click on the load balancer that you just created (`web-map-http`).
3. In the **Backend** section, click on the name of the backend and confirm that the VMs are **Healthy**. If they are not healthy, wait a few moments and try reloading the page.
4. When the VMs are healthy, test the load balancer using a web browser, going to `http://IP_ADDRESS/`, replacing `IP_ADDRESS` with the load balancer's IP address.

This may take three to five minutes. If you do not connect, wait a minute, and then reload the browser.

Your browser should render a page with content showing the name of the instance that served the page, along with its zone (for example, Page served from: `lb-backend-group-xxxx`).

Congratulations!

You built a network load balancer and an HTTP(s) load balancer and practiced using instance templates and managed instance groups.

Finish your Quest

This self-paced lab is part of the [Google Cloud Essentials](#) quest. A quest is a series of related labs that form a learning path. Completing this quest earns you a badge to recognize your achievement. You can make your badge or badges public and link to them in your online resume or social media account. [Enroll in this](#)

[quest](#) and get immediate completion credit.

See the [Google Cloud Skills Boost catalog](#) to see all available quests.

Take your next lab

Continue your quest with [Hello Node Kubernetes](#), or check out these suggested labs:

- [Provision Services with Google Cloud Marketplace](#)
- [Cloud Monitoring: Qwik Start](#)

Next steps / Learn more

- [Setting up a network load balancer with a backend service](#)
- [Setting up a simple external HTTPS load balancer](#)
- [External HTTP\(S\) Load Balancing Overview](#)

Google Cloud training and certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated August 12, 2022

Lab Last Tested November 14, 2022

Copyright 2023 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.