

# HTTP Load Balancer with Cloud Armor

**GSP215**



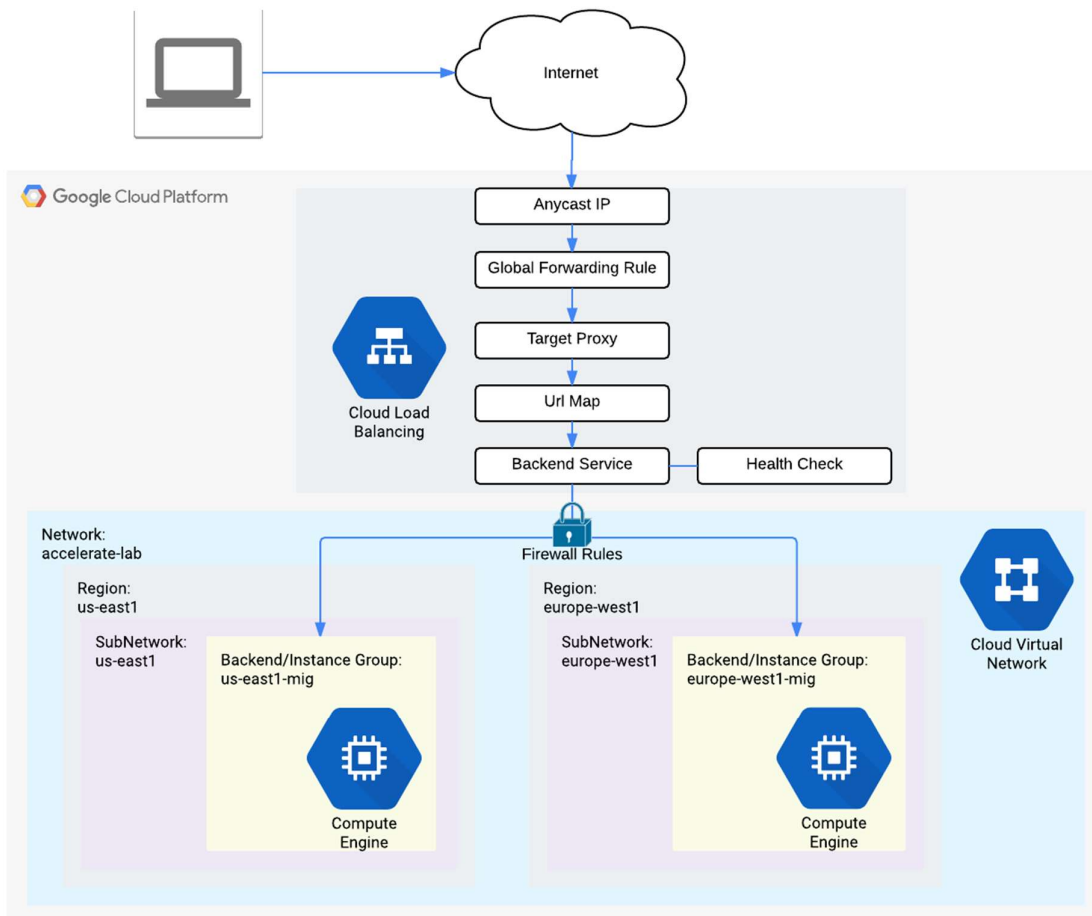
Google Cloud Self-Paced Labs

# 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 capacity available.

Cloud Armor IP allowlist/denylist enable you to restrict or allow access to your HTTP(S) load balancer at the edge of the Google Cloud, as close as possible to the user and to malicious traffic. This prevents malicious users or traffic from consuming resources or entering your virtual private cloud (VPC) networks.

In this lab, you configure an HTTP Load Balancer with global backends, as shown in the diagram below. Then, you stress test the Load Balancer and denylist the stress test IP with Cloud Armor.



# Objectives

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

- Create HTTP and health check firewall rules
- Configure two instance templates
- Create two managed instance groups
- Configure an HTTP Load Balancer with IPv4 and IPv6
- Stress test an HTTP Load Balancer
- Denylist an IP address to restrict access to an HTTP Load Balancer

## 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 Qwiklabs 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.

### What you need

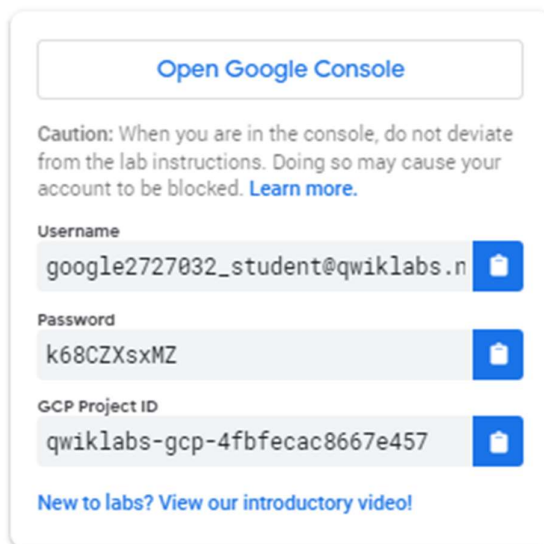
To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).
  - Time to complete the lab.
- Note:** If you already have your own personal Google Cloud account or project, do not use it for this lab.

**Note:** If you are using a Pixelbook, open an Incognito window to run this lab.


## 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 a panel populated with the temporary credentials that you must use for this lab.




Open Google Console

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.](#)

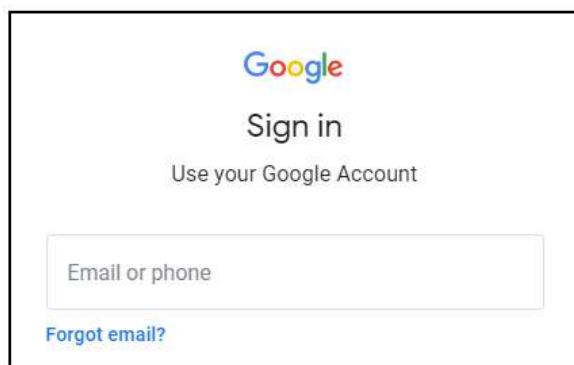
Username  
google2727032\_student@qwiklabs.n 

Password  
k68CZxsxMZ 

GCP Project ID  
qwiklabs-gcp-4fbfecac8667e457 

[New to labs? View our introductory video!](#)

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



Google

Sign in

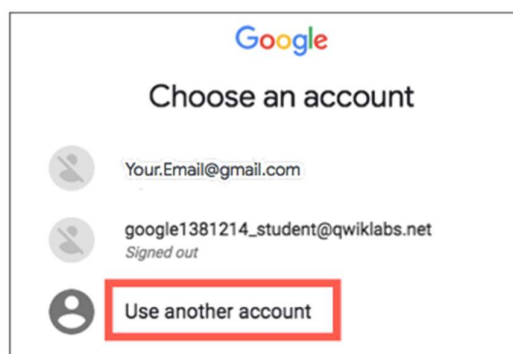
Use your Google Account

Email or phone

[Forgot email?](#)


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


If you see the **Choose an account** page, click **Use Another Account**.




Google

Choose an account

 Your.Email@gmail.com

 google1381214\_student@qwiklabs.net  
Signed out

 **Use another account**

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

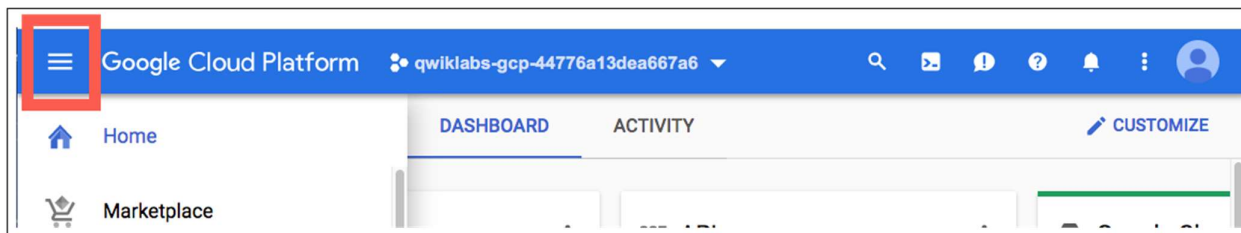
**Important:** You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. 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.



# Configure HTTP and health check firewall rules

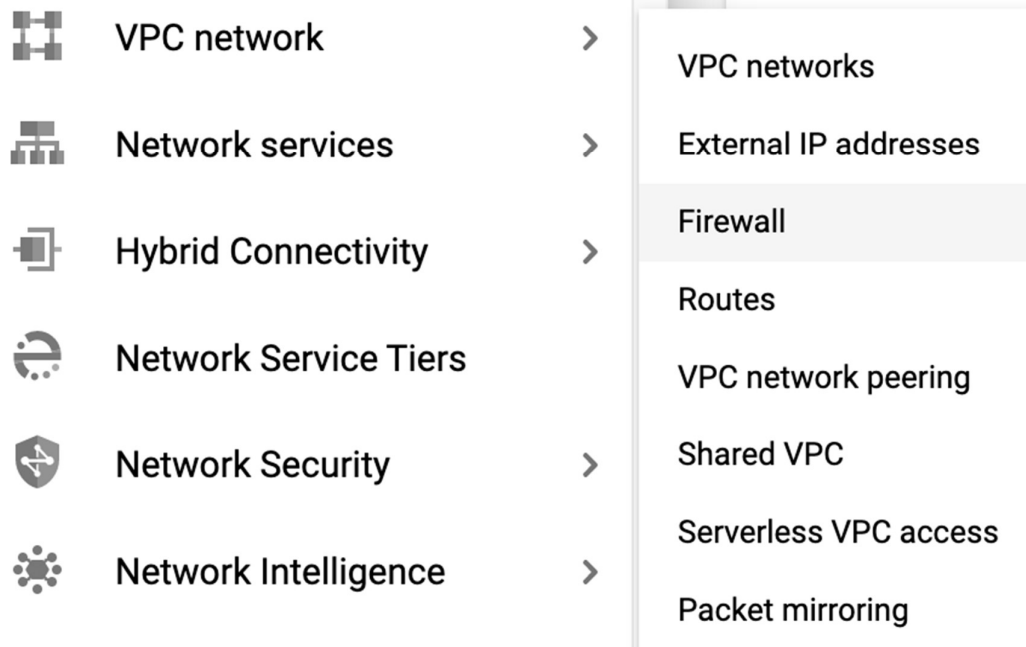
Configure firewall rules to allow HTTP traffic to the backends and TCP traffic from the Google Cloud health checker.

## Create the HTTP firewall rule

Create a firewall rule to allow HTTP traffic to the backends.

1. In the Cloud Console, navigate to **Navigation menu** (☰) > **VPC network** > **Firewall**.

### NETWORKING



2. Notice the existing **ICMP**, **internal**, **RDP**, and **SSH** firewall rules.

Each Google Cloud project starts with the **default** network and these firewall rules.

3. Click **Create Firewall Rule**.
4. Set the following values, leave all other values at their defaults:

Property	Value (type value or select option as specified)
Name	default-allow-http
Network	default
Targets	Specified target tags
Target tags	http-server
Source filter	IP Ranges
Source IP ranges	0.0.0.0/0
Protocols and ports	Specified protocols and ports, and then <i>check tcp, type: 80</i>

Make sure to include the **/0** in the **Source IP ranges** to specify all networks.

5. Click **Create**.

## Create the health check firewall rules

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.

1. Still in the **Firewall rules** page, click **Create Firewall Rule**.
2. Set the following values, leave all other values at their defaults:

Property	Value (type value or select option as specified)
Name	default-allow-health-check
Network	default
Targets	Specified target tags
Target tags	http-server
Source filter	IP Ranges
Source IP ranges	<code>130.211.0.0/22, 35.191.0.0/16</code>
Protocols and ports	Specified protocols and ports, and then <i>check tcp</i>

3. Make sure to enter the two **Source IP ranges** one-by-one and pressing SPACE in between them.
4. Click **Create**.


Click *Check my progress* to verify the objective.

# Configure instance templates 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 templates

An instance template is an API resource that you use to create VM instances and managed instance groups. Instance templates define the machine type, boot disk image, subnet, labels, and other instance properties. Create one instance template for **us-east1** and one for **europa-west1**.

1. In the Cloud Console, navigate to **Navigation menu** () > **Compute Engine** > **Instance templates**, and then click **Create instance template**.
2. For **Name**, type **us-east1-template**.
3. For **Series**, select **N1**.
4. Click **Management, security, disks, networking, sole tenancy**.

### Identity and API access

#### Service account

Compute Engine default service account 

#### Access scopes

- ☒ Allow default access
- ☐ Allow full access to all Cloud APIs
- ☐ Set access for each API

### Firewall

Add tags and firewall rules to allow specific network traffic from the Internet

- ☐ Allow HTTP traffic
- ☐ Allow HTTPS traffic

 [Management, security, disks, networking, sole tenancy](#)

---

You can create this instance template free of charge

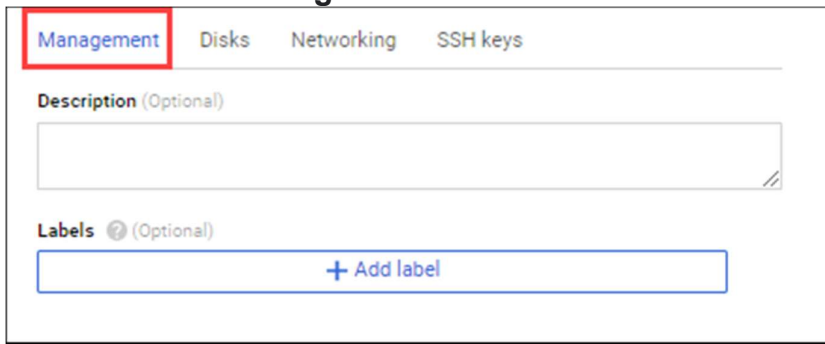
Create

Cancel

Equivalent [REST](#) or [command line](#)



5. Click the **Management** tab.



The screenshot shows the 'Management' tab selected in the Google Cloud Platform console. Below the tab are two optional fields: 'Description (Optional)' and 'Labels (Optional)'. The 'Labels (Optional)' field includes a '+ Add label' button.

6. Under **Metadata**, specify the following:

Key	Value
startup-script-url	gs://cloud-training/gcpnet/httplb/startup.sh

The `startup-script-url` specifies a script that executes when instances are started. This script installs Apache and changes the welcome page to include the client IP and the name, region, and zone of the VM instance. Feel free to explore this script [here](#).

7. Click **Networking**.

8. For **Network interfaces**, set the following values, leave all other values at their defaults:

Property	Value (type value or select option as specified)
Network	default
Subnet	default (us-east1)
Network tags	http-server

The network tag **http-server** ensures that the **HTTP** and **Health Check** firewall rules apply to these instances.

9. Click **Create**.

10. Wait for the instance template to be created.

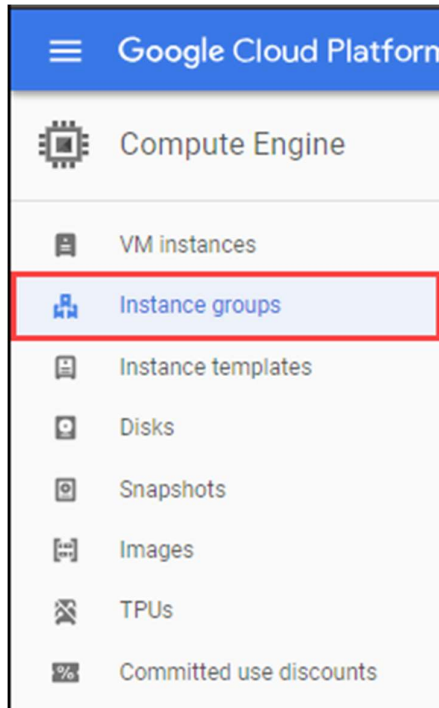
Now create another instance template for **subnet-b** by copying **us-east1-template**:

1. Click on **us-east1-template** and then click on **Create Similar** option from the top.
2. For **Name**, type **europe-west1-template**.
3. Click **Management, security, disks, networking, sole tenancy**.
4. Click **Networking**.
5. For **Network interfaces**, select **default (europe-west1)** as the **Subnet**.
6. Click **Create**.

# Create the managed instance groups

Create a managed instance group in **us-east1** and one in **europa-west1**.

1. Still in **Compute Engine**, click **Instance groups** in the left menu.



2. Click **Create instance group**.
3. Set the following values, leave all other values at their defaults:

Property	Value (type value or select option as specified)
Name	us-east1-mig
Location	Multiple zones
Region	us-east1
Instance template	us-east1-template
Autoscaling > Autoscaling metrics > Click Pencil icon > Metric type	CPU utilization
Target CPU utilization	80
Minimum number of instances	1
Maximum number of instances	5
Cool-down period	45

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.

4. Click **Done**.

5. Click **Create**.

Now repeat the same procedure for create a second instance group for **europe-west1-mig** in **europe-west1**:

1. Click **Create Instance group**.

2. Set the following values, leave all other values at their defaults:

Property	Value (type value or select option as specified)
Name	europe-west1-mig
Location	Multiple zones
Region	europe-west1
Instance template	europe-west1-template
Autoscaling > Autoscaling metrics > Click Pencil icon > Metric type	CPU utilization
Target CPU utilization	80
Minimum number of instances	1
Maximum number of instances	5
Cool-down period	45

3. Click **Done**.

4. Click **Create**.

Click *Check my progress* to verify the objective.

## Verify the backends

Verify that VM instances are being created in both regions and access their HTTP sites.

1. Still in **Compute Engine**, click **VM instances** in the left menu.
2. Notice the instances that start with `us-east1-mig` and `europa-west1-mig`.

These instances are part of the managed instance groups.

3. Click on the **External IP** of an instance of `us-east1-mig`.

You should see the **Client IP** (your IP address), the **Hostname** (starts with `us-east1-mig`) and the **Server Location** (a zone in `us-east1`).

4. Click on the **External IP** of an instance of `europa-west1-mig`.

You should see the **Client IP** (your IP address), the **Hostname** (starts with `europa-west1-mig`) and the **Server Location** (a zone in `europa-west1`).

### HTTP Load Balancing Lab

**Client IP**  
Your IP address : 104.133.0.93

**Hostname**  
Server Hostname: europa-west1-mig-wdr8

**Server Location**  
Region and Zone: europa-west1-b

The **Hostname** and **Server Location** identifies where the HTTP Load Balancer sends traffic.

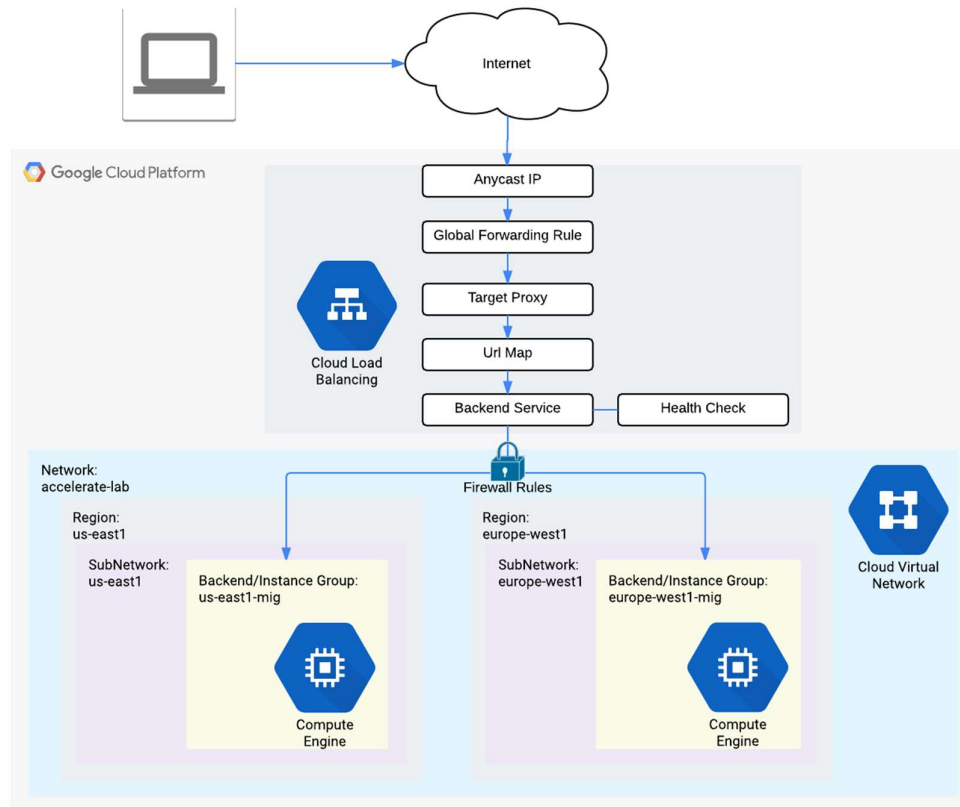
Which of these fields identify the region of the backend?

Server Location


Hostname

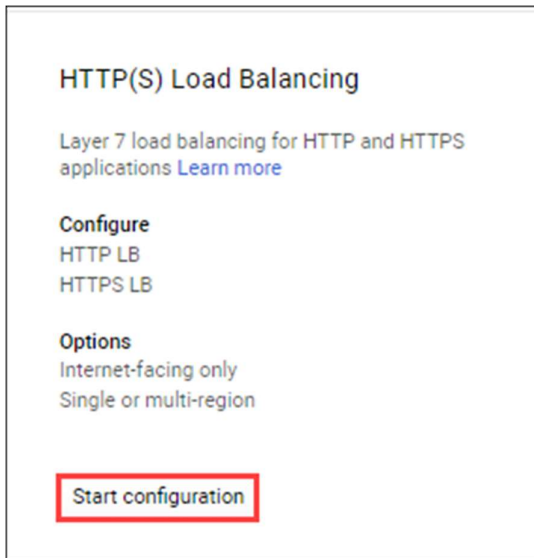
# Configure the HTTP Load Balancer

Configure the HTTP Load Balancer to balance traffic between the two backends (**us-east1-mig** in us-east1 and **europa-west1-mig** in europa-west1), as illustrated in the network diagram:



## Start the configuration

1. In the Cloud Console, click **Navigation menu** () > click **Network Services** > **Load balancing**, and then click **Create load balancer**.
2. Under **HTTP(S) Load Balancing**, click on **Start configuration**.



3. Select **From Internet to my VMs**, and click **Continue**.
4. Set the **Name** to `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.

1. Click on **Backend configuration**.
2. For **Backend services & backend buckets**, click **Create or select backend services & backend buckets**, then click **Backend services**, and then click **Create a backend service**.
3. Set the following values, leave all other values at their defaults:

Property	Value (select option as specified)
Name	http-backend
Instance group	us-east1-mig
Port numbers	80
Balancing mode	Rate
Maximum RPS	50
Capacity	100

This configuration means that the load balancer attempts to keep each instance of **us-east1-mig** at or below 50 requests per second (RPS).

4. Click **Done**.
5. Click **Add backend**.
6. Set the following values, leave all other values at their defaults:

Property	Value (select option as specified)
Instance group	europe-west1-mig
Port numbers	80
Balancing mode	Utilization
Maximum backend utilization	80
Capacity	100

This configuration means that the load balancer attempts to keep each instance of **europe-west1-mig** at or below 80% CPU utilization.

7. Click **Done**.
8. For **Health Check**, select **Create a health check**.

Create backend service

Name <sup>?</sup>

http-backend

⌵ Description

Protocol: HTTP Named port: http Timeout: 30 seconds

Backends

Regions: europe-west1, us-east1

us-east1-mig (Zone: us-east1, Port: 80) Not saved

europe-west1-mig (Zone: europe-west1, Port: 80) Not saved

+ Add backend

Cloud CDN <sup>?</sup>

☐ Enable Cloud CDN

Health check <sup>?</sup>

Create a health check

⌵ Advanced configurations (Session affinity, connection draining timeout, security policies)

9. Set the following values, leave all other values at their defaults:

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

Health checks determine which instances receive new connections. This HTTP health check polls instances every 5 seconds, waits up to 5 seconds for a response and treats 2 successful or 2 failed attempts as healthy or unhealthy, respectively.

10. Click **Save and Continue**.
11. Check the **Enable Logging** box.
12. Set the **Sample Rate** to 1:

Logging ?

☒ Enable logging

Sample rate ?

1

13. Click **Create** to create the backend service.

### Create backend service

Name ?

⌵ Description

Protocol: HTTP   Named port: http   Timeout: 30 seconds

Backends

Regions: europe-west1, us-east1

us-east1-mig (Zone: us-east1, Port: 80) *Not saved*

europe-west1-mig (Zone: europe-west1, Port: 80) *Not saved*

[+ Add backend](#)

Cloud CDN ?

☐ Enable Cloud CDN

Health check ?

http-health-check (TCP)

port: 80, timeout: 5s, check interval: 5s, unhealthy threshold: 2 attempts

⌵ [Advanced configurations \(Session affinity, connection draining timeout, security policies\)](#)

Create
Cancel



# 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 static traffic to another backend. However, you are not configuring the Host and path rules in this lab.

1. Click on **Frontend configuration**.
2. Specify the following, leaving all other values at their defaults:

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

3. Click **Done**.
4. Click **Add Frontend IP and port**.
5. Specify the following, leaving all other values at their defaults:

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

6. 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, then proxied over IPv4 to your backends.

# Review and create the HTTP Load Balancer

## 1. Click on **Review and finalize**.

←

New HTTP(S) load balancer

Name ⓘ

lowercase, no spaces

✓

Backend configuration

You have configured 1 backend(s)

●

Host and path rules

You have created host and path rules

✓

Frontend configuration

Your frontend is configured →

ⓘ

Review and finalize

Optional

Create

Cancel

Frontend configuration

Specify an IP address, port and protocol. This IP address is the frontend IP for your clients requests. For SSL, a certificate must also be assigned.

Protocol:HTTP, IP: EPHEMERAL:80, Port:80

Not saved

✎

Protocol:HTTP, IP: [EPHEMERAL]:80, Port:80

Not saved

✎

+ Add Frontend IP and port

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

Review and finalize

Backend

Backend services

1. http-backend

Endpoint protocol: HTTP   Named port: http   Timeout: 30 seconds   Cloud CDN: disabled   Health check: http-health-check

⌵ Advanced configurations

Instance group	Zone	Autoscaling	Balancing mode	Capacity
europa-west1-mig	europa-west1	Target CPU usage 80%	Max CPU: 80%	100%
us-east1-mig	us-east1	Target CPU usage 80%	Max RPS: 50 (per instance)	100%

Host and path rules

Hosts	Paths	Backend
All unmatched (default)	All unmatched (default)	http-backend

Frontend

Protocol	IP:Port	Certificate	Network Tier ⓘ
HTTP	EPHEMERAL:80	—	Premium
HTTP	[EPHEMERAL]:80	—	Premium

## 3. Click on **Create**.

## 4. Wait for the load balancer to be created.

5. Click on the name of the load balancer (**http-lb**).
6. 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.

Click *Check my progress* to verify the objective.

# Test the HTTP Load Balancer

Now that you created the HTTP Load Balancer for your backends, verify that traffic is forwarded to the backend service.

The HTTP load balancer should forward traffic to the region that is closest to you.  
True


## Access the HTTP Load Balancer

To test IPv4 access to 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.

It might take up to 5 minutes to access the HTTP Load Balancer. In the meantime, you might get a 404 or 502 error. Keep trying until you see the page of one of the backends. Depending on your proximity to **us-east1** and **europa-west1**, you traffic is either forwarded to a **us-east1-mig** or **europa-west1-mig** instance. If you have a local IPv6 address, try the IPv6 address of the HTTP Load Balancer by navigating to `http://[LB_IP_v6]`. Make sure to replace `[LB_IP_v6]` with the IPv6 address of the load balancer.

## Stress test the HTTP Load Balancer

Create a new VM to simulate a load on the HTTP Load Balancer using `siege`. Then, determine if traffic is balanced across both backends when the load is high.

1. In the Console, navigate to **Navigation menu** () > **Compute Engine** > **VM instances**.
2. Click **Create instance**.
3. Set the following values, leave all other values at their defaults:

Property	Value (type value or select option as specified)
Name	siege-vm
Region	us-west1
Zone	us-west1-c
Series	N1

Given that **us-west1** is closer to **us-east1** than to **europa-west1**, traffic should be forwarded only to **us-east1-mig** (unless the load is too high).

4. Click **Create**.
5. Wait for the **siege-vm** instance to be created.
6. For **siege-vm**, click **SSH** to launch a terminal and connect.
7. Run the following command, to install siege:

```
sudo apt-get -y install siege
```

8. To store the IPv4 address of the HTTP Load Balancer in an environment variable, run the following command, replacing [LB\_IP\_v4] with the IPv4 address:


```
export LB_IP=[LB_IP_v4]
```

9. To simulate a load, run the following command:

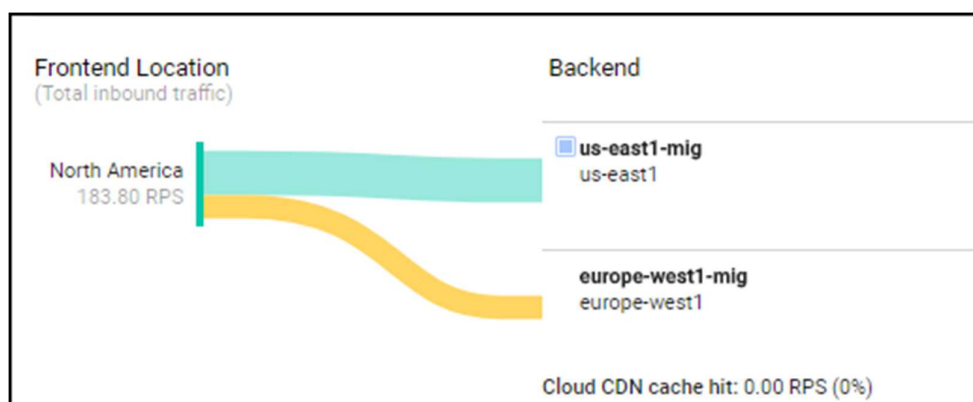
```
siege -c 250 http://$LB_IP
```

The output should look like this (**do not copy; this is example output**):

```
New configuration template added to /home/cloudcurriculumdeveloper/.siege
Run siege -C to view the current settings in that file
[alert] Zip encoding disabled; siege requires zlib support to enable it: No such file
or directory
** SIEGE 4.0.2
** Preparing 250 concurrent users for battle.
The server is now under siege...
```

10. In the Cloud Console, on the **Navigation menu** () click **Network Services > Load balancing**.
11. Click **Backends**.
12. Click **http-backend**.
13. Monitor the **Frontend Location (Total inbound traffic)** between North America and the two backends for 2 to 3 minutes.

At first, traffic should just be directed to **us-east1-mig** but as the RPS increases, traffic is also directed to **europa-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.

14. Return to the **SSH** terminal of **siege-vm**.
15. Press **CTRL+C** to stop siege.

# Denylist the siege-vm

Use Cloud Armor to denylist the **siege-vm** from accessing the HTTP Load Balancer.

## Create the security policy

Create a Cloud Armor security policy with a denylist rule for the **siege-vm**.

1. In the Console, navigate to **Navigation menu** () > **Compute Engine** > **VM instances**.

2. Note the **External IP** of the **siege-vm**. This will be referred to as `[SIEGE_IP]`.

There are ways to identify the external IP address of a client trying to access your HTTP Load Balancer. For example, you could examine traffic captured by [VPC Flow Logs in BigQuery](#) to determine a high volume of incoming requests.

3. In the Cloud Console, navigate to **Navigation menu** > **Network Security** > **Cloud Armor**.

4. Click **Create policy**.

5. Set the following values, leave all other values at their defaults:

Property	Value (type value or select option as specified)
Name	denylist-siege
Default rule action	Allow



6. Click **Next step**.

7. Click **Add rule**.

8. Set the following values, leave all other values at their defaults:

Property	Value (type value or select option as specified)
Condition	<i>Enter the SIEGE_IP</i>
Action	Deny
Deny status	403 (Forbidden)
Priority	1000

9. Click **Done**.

New rule



Description (Optional)

Condition ?

Mode



☒ Basic mode (IP addresses/ranges only) ?
☐ Advanced mode ?

Match ?

You can put up to 10 IP addresses or ranges per rule. Use comma to separate IP address/ranges.

34.83.131.143

Action ?

☐  Allow
☒  Deny

Deny status ?

403 (Forbidden) ▼

Preview only ?
☐ Enable

Priority ?

Priority is evaluated from 0 (highest) to 2,147,483,647 (lowest)

a positive integer (lower values take precedence)

Done

Cancel

10. Click **Next step**.
11. Click **Add Target**.
12. For **Type**, select **Load balancer backend service**.
13. For **Target**, select **http-backend**.
14. Click **Done**.
15. Click **Create policy**.

Alternatively, you could set the default rule to **Deny** and only allowlist/allow traffic from authorized users/IP addresses.

16. Wait for the policy to be created before moving to the next step. Click *Check my progress* to verify the objective.



# Verify the security policy

Verify that the **siege-vm** cannot access the HTTP Load Balancer.

1. Return to the **SSH** terminal of **siege-vm**.
2. To access the load balancer, run the following:

```
curl http://$LB_IP
```

The output should look like this (**do not copy; this is example output**):

```
<!doctype html><meta charset="utf-8"><meta name=viewport content="width=device-width,
initial-scale=1"><title>403</
title>403 Forbidden
```

It might take a couple of minutes for the security policy to take affect. If you are able to access the backends, keep trying until you get the **403 Forbidden error**.

3. 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.

You can access the HTTP Load Balancer from your browser because of the default rule to **allow** traffic; however, you cannot access it from the **siege-vm** because of the **deny** rule that you implemented.

4. Back in the SSH terminal of **siege-vm**, to simulate a load, run the following command:

```
siege -c 250 http://$LB_IP
```

The output should look like this (**do not copy; this is example output**):


```
[alert] Zip encoding disabled; siege requires zlib support to enable it
** SIEGE 4.0.2
** Preparing 250 concurrent users for battle.
The server is now under siege...
```

Explore the security policy logs to determine if this traffic is also blocked.

5. In the Console, navigate to **Navigation menu > Network Security > Cloud Armor**.
6. Click **denylist-siege**.
7. Click **Logs**.
8. Click **View policy logs**.
9. On the Logging page, make sure to clear all the text in the **Query preview**. Select resource to **Cloud HTTP Load Balancer > http-lb-forwarding-rule > http-lb** then click **Add**.
10. Now click **Run Query**.
11. Expand a log entry in **Query results**.
12. Expand **HttpRequest**.


The request should be from the **siege-vm** IP address. If not, expand another log entry.


13. Expand **jsonPayload**.
14. Expand **enforcedSecurityPolicy**.
15. Notice that the **configuredAction** is to **DENY** with the **name** `denylist-siege`.


**Query results** 


[Jump to Now](#) [Actions](#) [Configure](#)

SEVERITY	TIMESTAMP	IST	SUMMARY
			Mozilla/5.0 ... <a href="http://34.120.174.190/">http://34.120.174.190/</a>

 Hide log summary

 Expand nested fields

 Copy to clipboard

 Copy link

▼ {

insertId: "1q5y1snfnjuuxe"

▼ jsonPayload: {

▼ enforcedSecurityPolicy: {

outcome: "DENY"

configuredAction: "DENY"

priority: 1000

name: "denylist-siege"

}

}

}

Cloud Armor security policies create logs that can be explored to determine when traffic is denied and when it is allowed, along with the source of the traffic.

# Congratulations!

You configured an HTTP Load Balancer with backends in us-east1 and europe-west1. Then, you stress tested the Load Balancer with a VM and denylisted the IP address of that VM with Cloud Armor. You were able to explore the security policy logs to identify why the traffic was blocked.



## Finish Your Quest

This self-paced lab is part of the Qwiklabs [Networking in the Google Cloud](#) Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above 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 if you've taken this lab. [See other available Qwiklabs Quests](#).

## Take Your Next Lab

Continue your Quest with [Create an Internal Load Balancer](#), or check out these suggestions:

- [Using a Vault on Compute Engine for Secret Management](#)
- [Continuous Delivery with Jenkins in Kubernetes Engine](#)

## Next Steps / Learn More

For information on the basic concepts of Cloud Armor, see [Cloud Armor Documentation](#). For more information on Load Balancing, see [Load Balancing](#).

Manual Last Updated April 02, 2021

Lab Last Tested April 02, 2021

Copyright 2021 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.