Configuring an HTTP Load Balancer with Cloud Armor

2 hoursFree

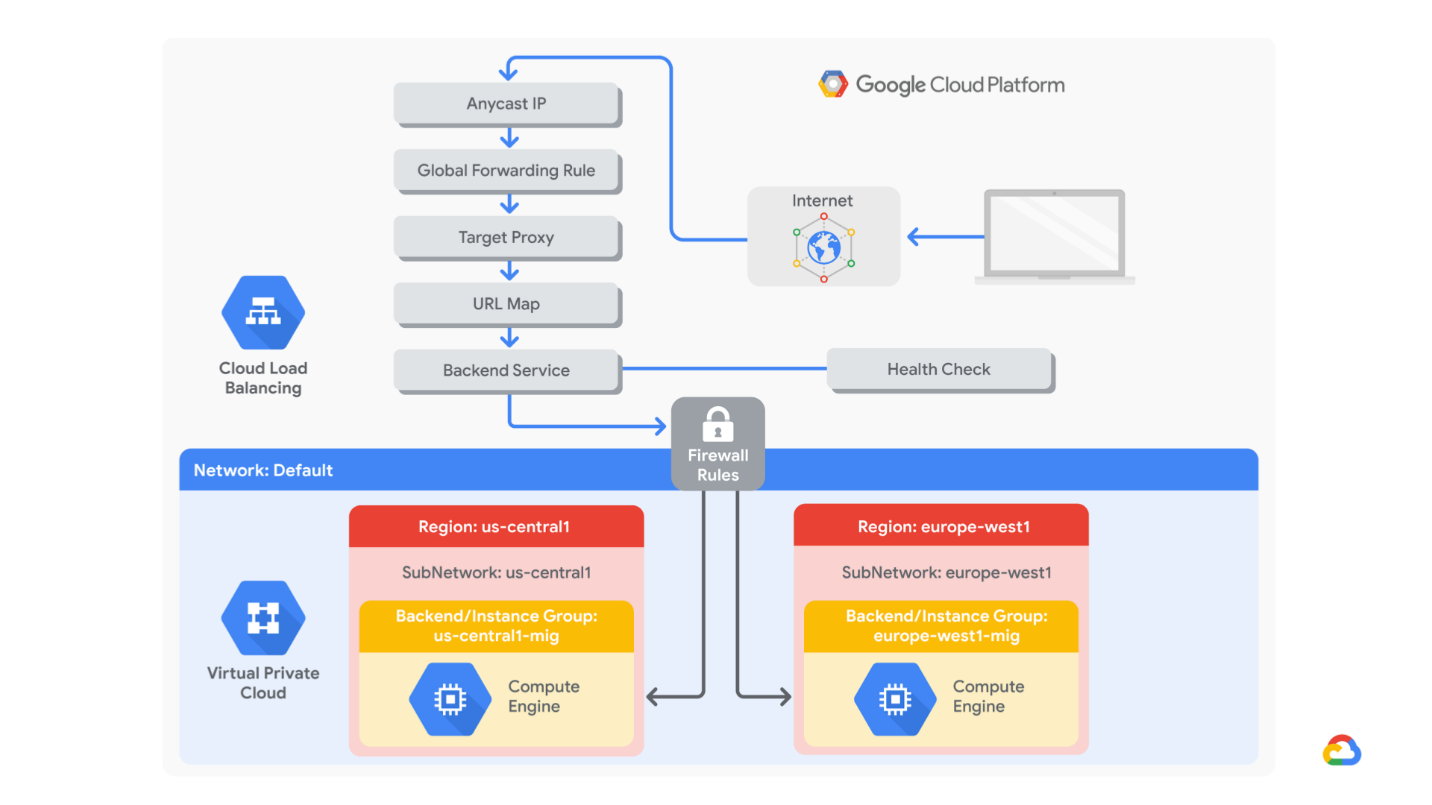
Rate Lab

**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 blacklists/whitelists 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 blacklist the stress test IP with Cloud Armor.



Objectives

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

* Create an HTTP and health check firewall rule
* Configure two instance templates
* Create two managed instance groups
* Configure an HTTP load balancer with IPv4 and IPv6
* Stress test an HTTP load balancer
* Blacklist an IP address to restrict access to an HTTP load balancer

**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 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 the Google Cloud Platform 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 GCP account or project, do not use it for this lab.

**Task 1. Configure an HTTP and health check firewall rule**

Configure a firewall rule 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, on the **Navigation menu** (Navigation menu), click **VPC network** > **Firewall rules**. Notice the existing **ICMP**, **internal**, **RDP**, and **SSH** firewall rules.

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

1. Click **Create Firewall Rule**.
2. Specify the following, and leave the remaining settings as 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 |

1. For **tcp**, specify port **80**.

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

1. 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.

As your **default-allow-http** firewall rule already allows connections from **0.0.0.0/0** for **tcp:80**, you don't have to create an additional health check firewall rule in this lab.

Click *Check my progress* to verify the objective.

Configure HTTP and health check firewall rule

Check my progress

**Task 2. 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 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. Create one instance template for **us-central1** and one for **europe-west1**.

1. In the Cloud Console, on the **Navigation menu** (Navigation menu), click **Compute Engine** > **Instance templates**.
2. Click **Create instance template**.
3. For **Name**, type **us-central1-template**.
4. Click **Management, security, disks, networking, sole tenancy**.
5. Click **Management**.
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 will be executed 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. You can explore this script [here](https://storage.googleapis.com/cloud-training/gcpnet/httplb/startup.sh).

1. Click **Networking**.
2. Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (type value or select option as specified)** |
| Network | default |
| Subnet | default (us-central1) |
| Network tags | http-server |

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

1. Click **Create**. Wait for the instance template to be created.

Create another instance template by copying **us-central1-template**:

1. Select the **us-central1-template**, and click **Copy**.
2. For **Name**, type **europe-west1-template**.
3. Click **Management, security, disks, networking, sole tenancy**.
4. Click **Networking**.
5. Select **default (europe-west1)** as the **Subnet**.
6. Click **Create**.

**Create the managed instance groups**

Create a managed instance group in **us-central1** and one in **europe-west1**.

1. On the **Navigation menu**, click **Compute Engine** > **Instance groups**.
2. Click **Create Instance group**.
3. Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (type value or select option as specified)** |
| Name | us-central1-mig |
| Location | Multiple zones |
| Region | us-central1 |
| Instance template | us-central1-template |
| Autoscaling > Autoscaling metrics | Click pencil icon, set Target CPU utilization 80 |
| Cool-down period | 45 |
| Minimum number of instances | 1 |
| Maximum number of instances | 5 |

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.

1. Click **Create**.

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

1. Click **Create Instance group**.
2. 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 | europe-west1-template |
| Autoscaling > Autoscaling metrics | Click pencil icon, set Target CPU utilization 80 |
| Cool-down period | 45 |
| Minimum number of instances | 1 |
| Maximum number of instances | 5 |

1. Click **Create**.

Click *Check my progress* to verify the objective.

Configure instance templates and create instance groups

Check my progress

**Verify the backends**

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

1. 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.

1. Click on the **External IP** of an instance of *us-central1-mig*.

You should see the **Client IP** (your IP address), the **Hostname** (starts with *us-central1-mig*), and the **Server Location** (a zone in us-central1).

1. Click on the **External IP** of an instance of *europe-west1-mig*.

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

Which of these fields identify the region of the backend?



Hostname



Client IP

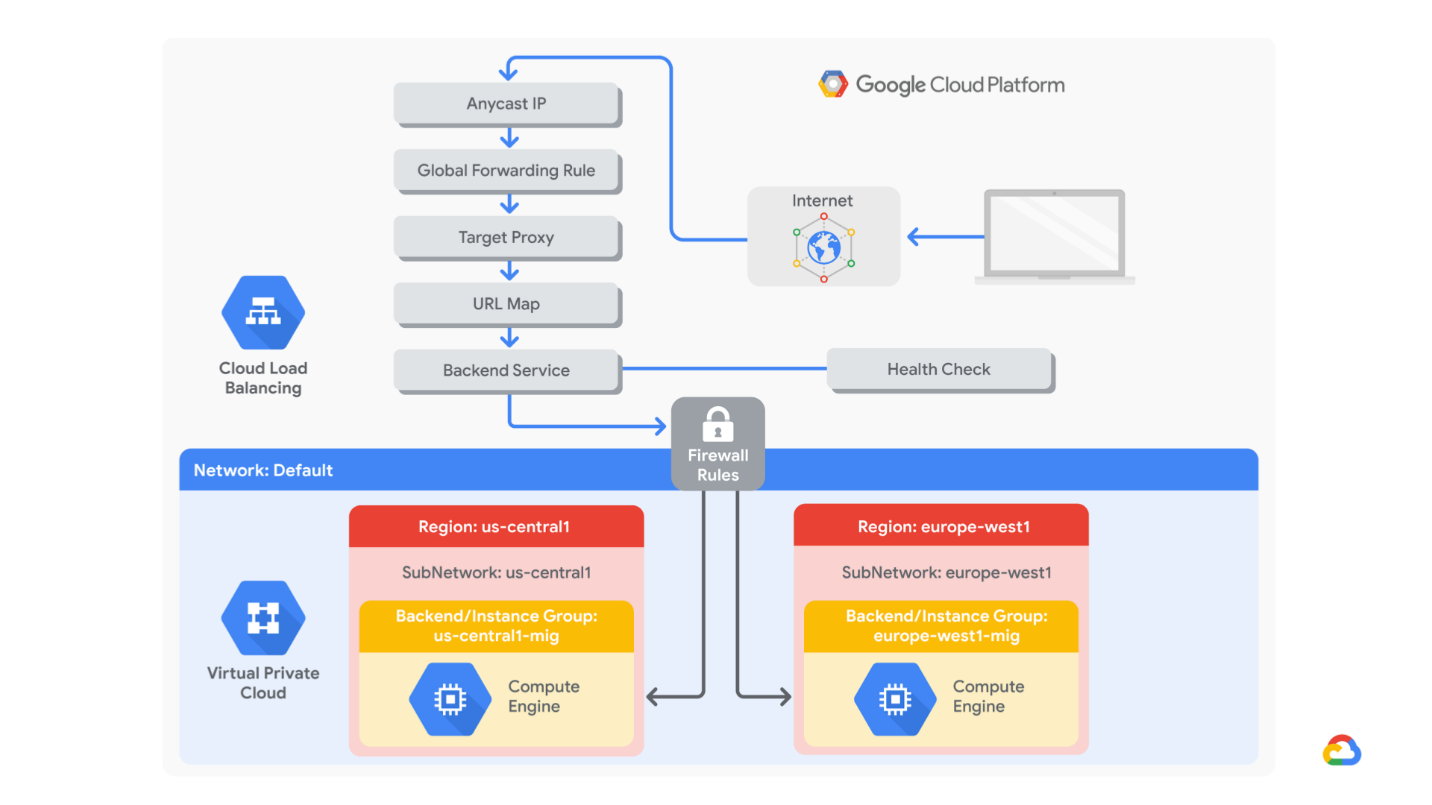


Server Location

Submit

**Task 3. 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**

1. On the **Navigation menu**, click **Network Services** > **Load balancing**.
2. Click **Create load balancer**.
3. Under **HTTP(S) Load Balancing**, click **Start configuration**.
4. Click **Continue**.
5. 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.

1. Click **Backend configuration**.
2. For **Backend services & backend buckets**, click **Create or select backend services & backend buckets** > **Backend services** > **Create a backend service**.
3. Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (select option as specified)** |
| Name | http-backend |
| Instance group | us-central1-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-central1-mig** at or below 50 requests per second (RPS).

1. Click **Done**.
2. Click **Add backend**.
3. Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (select option as specified)** |
| Instance group | europe-west1-mig |
| Port numbers | 80 |
| Balancing mode | Utilization |
| Maximum CPU 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.

1. Click **Done**.
2. For **Health Check**, select **Create a health check**.
3. 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 |

Health checks determine which instances can 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.

1. Click **Save and Continue**.
2. Click **Create**.

**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, 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 |

1. Click **Done**.
2. Click **Add Frontend IP and port**.
3. 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 |

1. 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**

1. Click **Review and finalize**.
2. Review the **Backend services** and **Frontend**.
3. Click **Create**. Wait for the load balancer to be created.
4. Click on the name of the load balancer (**http-lb**).
5. 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.

Configure the HTTP Load Balancer

Check my progress

**Task 4. Test the HTTP load balancer**

Now that you have created the HTTP load balancer for your backends, it is time to 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

False

**Access the HTTP load balancer**

1. 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 a couple of 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-central1** and **europe-west1**, you traffic is either forwarded to a **us-central1-mig** or **europe-west1-mig** instance.

1. 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 whether traffic is balanced across both backends when the load is high.

1. In the Cloud Console, on the **Navigation menu** (Navigation menu), click **Compute Engine** > **VM instances**.
2. Click **Create instance**.
3. Specify the following, and leave the remaining settings as their defaults:

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

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).

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

sudo apt-get -y install siege

1. 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]

1. 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...

1. In the Cloud Console, on the **Navigation menu** (Navigation menu), click **Network Services** > **Load balancing**.
2. Click **Backends**.
3. Click **http-backend**.
4. 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-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.

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

**Task 5. Blacklist the siege-vm**

Use Cloud Armor to blacklist the **siege-vm** from accessing the HTTP load balancer.

**Create the security policy**

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

1. In the Cloud Console, on the **Navigation menu**, click **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](https://cloud.google.com/vpc/docs/using-flow-logs#exporting_logs_to_bigquery_name_short_pubsub_name_short_and_custom_targets) to determine a high volume of incoming requests.

1. On the **Navigation menu**, click **Network Security** > **Cloud Armor**.
2. Click **Create policy**.
3. Specify the following, and leave the remaining settings as their defaults:

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

1. Click **Next step**.
2. Click **Add rule**.
3. Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (type value or select option as specified)** |
| Condition | *Enter the SIEGE\_IP* |
| Action | Deny |
| Deny status | 403 (Forbidden) |
| Priority | 1000 |

1. Click **Done**.
2. Click **Next step**.
3. Click **Add Target**.
4. For **Type**, select **Load balancer backend service**.
5. For **Target**, select **http-backend**.
6. Click **Done**.
7. Click **Create policy**. Wait for the policy to be created before moving to the next step.

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

Click *Check my progress* to verify the objective.

Blacklist the siege-vm

Check my progress

**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**.

1. 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.

1. 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 whether this traffic is also blocked.

1. In the Cloud Console, on the **Navigation menu**, click **Network Security** > **Cloud Armor**.
2. Click **blacklist-siege**.
3. Click **Logs**.
4. Click **View policy logs**.
5. Expand a log entry.
6. Expand **httpRequest**.

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

1. Expand **jsonPayload**.
2. Expand **enforcedSecurityPolicy**. Notice that the **configuredAction** is to DENY with the **name** blacklist-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.

**Task 6. 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 and blacklisted 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.