# Set Up Network and HTTP Load Balancers

Google Cloud Self-Paced Labs

## Overview

In this hands-on lab, you'll learn the differences betw een a netw ork load balancer and a HTTP load balancer, and how to set them up for your applications running on Google Compute Engine virtual machines.

There are several w ays you can load balance in Google Cloud Platform. This lab takes you through the setup of the follow ing load balancers.:

- L3 Netw ork Load Balancer
- L7 HTTP(s) Load Balancer

Students are encouraged to type the commands themselves, w hich helps in learning 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 do

- Setup a netw ork load balancer.

- Setup a HTTP(s) load balancer.

- Get hands-on experience learning the differences betw een netw ork load balancers and HTTP load balancers.

## Prerequisites

Familiarity w ith standard Linux text editors such as `vim`, `emacs`, or `nano` is helpful.

### What you need

To complete this lab, you need:

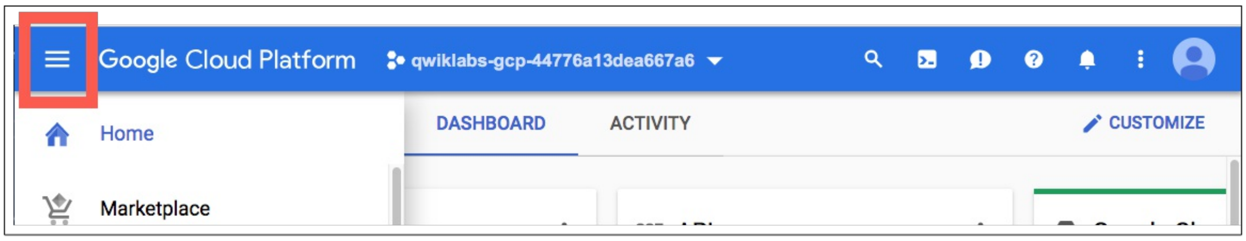
- Access to a standard internet brow ser (Chrome brow ser recommended).
- Time to complete the lab.

### How to start your lab and sign in to the Console

- Open https://console.cloud.google.com/
- Enter login credentials

After a few moments, the GCP console opens in this tab.

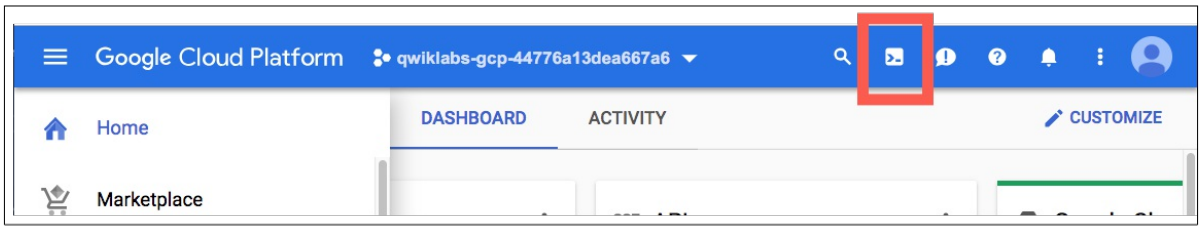
**Note:** You can view the menu w ith a list of GCP Products and Services by clicking the **Navigation menu** at the top-left, next to “Google Cloud Platform”.



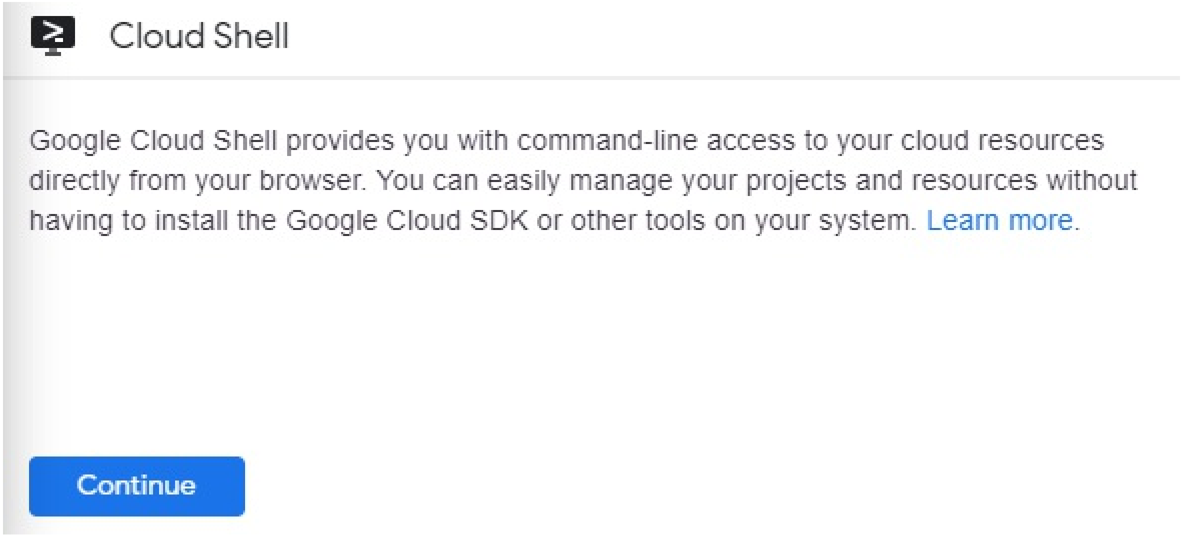
## Activate Google Cloud Shell

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

1. In GCP console, on the top right toolbar, click the Open Cloud Shell button.

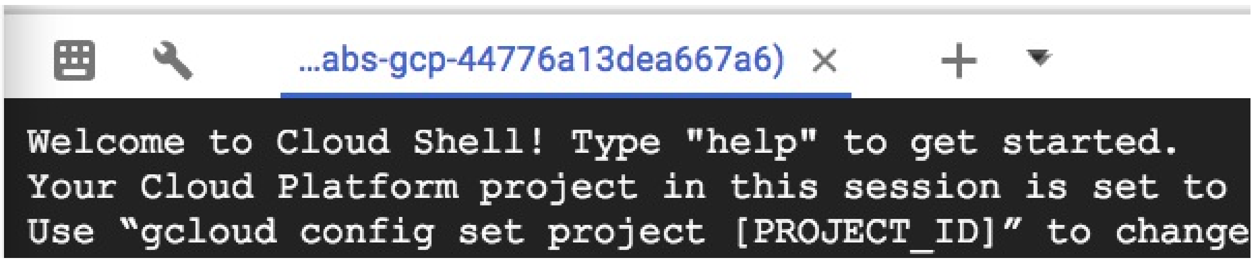


2. Click **Continue**.



Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. Learn more.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to
Use "gcloud config set project [PROJECT_ID]" to change
```

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

You can list the active account name w ith this command:

```
gcloud auth list
```

Output:

```
Credentialed accounts:
 - <myaccount>@<mydomain>.com (active)
```

Example output:

```
Credentialed accounts:
 - google1623327_student@testlabs.net
```

You can list the project ID w ith this command:

```
gcloud config list project
```

Output:

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

Example output:

```
[core]
project = testlabs-gcp-44776a13dea667a6
```

Full documentation of **gcloud** is available on Google Cloud gcloud Overview .

## Set the default region and zone for all resources

In Cloud Shell, set the default zone:

```
gcloud config set compute/zone us-central1-a
```

Set the default region:

```
gcloud config set compute/region us-central1
```

Learn more about choosing zones and regions here: Regions & Zones documentation.

**Note:** When you run `gcloud` on your own machine, the `config` settings persist across sessions. In Cloud Shell you need to set this for every new session or reconnection.

## Create multiple web server instances

To simulate serving from a cluster of machines, create a simple cluster of Nginx web servers to serve static content using Instance Templates and Managed Instance Groups.
Instance Templates define the look of every virtual machine in the cluster (disk, CPUs, memory, etc). Managed Instance Groups instantiate a number of virtual machine instances using the Instance Template.

To create the Nginx web server clusters, create the following:

- A startup script to be used by every virtual machine instance to setup Nginx server upon startup
- An instance template to use the startup script
- A target pool
- A managed instance group using the instance template

Still in Cloud Shell, create a startup script to be used by every virtual machine instance. This script sets up the Nginx server upon startup:

```
cat << EOF > startup.sh
#! /bin/bash
apt-get update
apt-get install -y nginx
service nginx start
sed -i -- 's/nginx/Google Cloud Platform - '"\$HOSTNAME"'/' /var/www/html/index.nginx-debian.html
EOF
```

Create an instance template, which uses the startup script:

```
gcloud compute instance-templates create nginx-template \
         --metadata-from-file startup-script=startup.sh
```

(Output)

```
Created [...].
NAME           MACHINE_TYPE  PREEMPTIBLE CREATION_TIMESTAMP
nginx-template n1-standard-1             2015-11-09T08:44:59.007-08:00
```

Create a target pool. A target pool allows a single access point to all the instances in a group and is necessary for load balancing in the future steps.

```
gcloud compute target-pools create nginx-pool
```

(Output)

```
Created [...].
NAME       REGION       SESSION_AFFINITY BACKUP HEALTH_CHECKS
nginx-pool us-central1
```

Create a managed instance group using the instance template:

```
gcloud compute instance-groups managed create nginx-group \
         --base-instance-name nginx \
         --size 2 \
         --template nginx-template \
         --target-pool nginx-pool
```

(Output)

```
Created [...].
NAME         LOCATION       SCOPE  BASE_INSTANCE_NAME  SIZE  TARGET_SIZE  INSTANCE_TEMPLATE  AUTOSCALED
nginx-group  us-central1-a  zone   nginx               0     2            nginx-template     no
```

This creates 2 virtual machine instances w ith names that are prefixed w ith `nginx-`. This may take a couple of minutes.

List the compute engine instances and you should see all of the instances created:

```
gcloud compute instances list
```

(Output)

```
NAME       ZONE           MACHINE_TYPE  PREEMPTIBLE INTERNAL_IP EXTERNAL_IP    STATUS
nginx-7wvi us-central1-a n1-standard-1             10.240.X.X  X.X.X.X           RUNNING
nginx-9mwd us-central1-a n1-standard-1             10.240.X.X  X.X.X.X           RUNNING
```

Now configure a firew all so that you can connect to the machines on port 80 via the `EXTERNAL_IP` addresses:

```
gcloud compute firewall-rules create www-firewall --allow tcp:80
```

You should be able to connect to each of the instances via their external IP addresses via `http://EXTERNAL_IP/` show n as the result of running the previous command.

Check your lab progress. Click **Check my progress** below to verify that you've created a group of webservers.

Create a group of webservers

## Create a Network Load Balancer

Network load balancing allows you to balance the load of your systems based on incoming IP protocol data, such as address, port, and protocol type. You also get some options that are not available, with HTTP(S) load balancing. For example, you can load balance additional TCP/UDP-based protocols such as SMTP traffic. And if your application is interested in TCP-connection-related characteristics, network load balancing allows your app to inspect the packets, where HTTP(S) load balancing does not.

For more information, see Setting Up Network Load Balancing.

Create an L3 network load balancer targeting your instance group:

```
gcloud compute forwarding-rules create nginx-lb \
         --region us-central1 \
         --ports=80 \
         --target-pool nginx-pool
```

(Output)

```
Created [https://www.googleapis.com/compute/v1/projects/...].
```

List all Google Compute Engine forwarding rules in your project.

```
gcloud compute forwarding-rules list
```

(Output)

```
NAME     REGION      IP_ADDRESS     IP_PROTOCOL TARGET
nginx-lb us-central1 X.X.X.X        TCP         us-central1/targetPools/nginx-pool
```

You can then visit the load balancer from the browser `http://IP_ADDRESS/` where `IP_ADDRESS` is the address shown as the result of running the previous command.

Check your lab progress. Click **Check my progress** below to verify that you've created an L3 Network Load Balancer that points to the webservers.

Create an L3 Network Load Balancer that points to the webservers

## Create a HTTP(s) Load Balancer

HTTP(S) load balancing provides global load balancing for HTTP(S) requests destined for your instances. You can configure URL rules that 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, provided 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.

Learn more about the HTTP(s) Load Balancer in the documentation.

First, create a health check.
Health checks verify that the instance is responding to HTTP or HTTPS traffic:

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

(Output)

```
Created [https://www.googleapis.com/compute/v1/projects/...].
NAME             HOST PORT REQUEST_PATH
http-basic-check      80   /
```

Define an HTTP service and map a port name to the relevant port for the instance group. Now the load balancing service can forward traffic to the named port:

```
gcloud compute instance-groups managed \
       set-named-ports nginx-group \
       --named-ports http:80
```

(Output)

```
Updated [https://www.googleapis.com/compute/v1/projects/...].
```

Create a backend service:

```
gcloud compute backend-services create nginx-backend \
      --protocol HTTP --http-health-checks http-basic-check --global
```

(Output)

```
Created [https://www.googleapis.com/compute/v1/projects/...].
NAME          BACKENDS PROTOCOL
nginx-backend          HTTP
```

Add the instance group into the backend service:

```
gcloud compute backend-services add-backend nginx-backend \
    --instance-group nginx-group \
    --instance-group-zone us-central1-a \
    --global
```

(Output)

```
Updated [https://www.googleapis.com/compute/v1/projects/...].
```

Create a default URL map that directs all incoming requests to all your instances:

```
gcloud compute url-maps create web-map \
    --default-service nginx-backend
```

(Output)

```
Created [https://www.googleapis.com/compute/v1/projects/...].
NAME    DEFAULT_SERVICE
Web-map nginx-backend
```

To direct traffic to different instances based on the URL being requested, see content-based routing.

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
```

(Output)

```
Created [https://www.googleapis.com/compute/v1/projects/...].
NAME          URL_MAP
http-lb-proxy web-map
```

Create a global forw arding rule to handle and route incoming requests. A forw arding rule sends traffic to a specific target HTTP or HTTPS proxy depending on the IP address, IP protocol, and port specified. The global forw arding rule does not support multiple ports.

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

(Output)

```
Created [https://www.googleapis.com/compute/v1/projects/...].
```

After creating the global forw arding rule, it can take several minutes
for your configuration to propagate.

```
gcloud compute forwarding-rules list
```

(Output)

```
NAME              REGION IP_ADDRESS    IP_PROTOCOL TARGET
http-content-rule        X.X.X.X       TCP         http-lb-proxy
nginx-lb   us-central1  X.X.X.X       TCP         us-central1/....
```

Take note of the http-content-rule IP_ADDRESS for the forw arding rule.

From the brow ser, you should be able to connect to `http://IP_ADDRESS/`.
It may take three to five minutes. If you do not connect, w ait a minute
then reload the brow ser.

Check your lab progress. Click **Check my progress** below to verify
that you've created an L7 HTTP(S) Load Balancer.

Create an L7 HTTP(S) Load Balancer

## Test your knowledge

Test your know ledge about Google cloud Platform by taking our quiz.
(Please select multiple correct options if necessary.)

## Congratulations!

You built a netw ork load balancer and a HTTP(s) load balancer. You
practiced w ith Instance templates and Managed Instance Groups. You are
w ell on your w ay to having your Google Cloud Platform project monitored
w ith Cloud Monitoring.

### Next Steps / Learn More

- Deploy more resources to your project, and see them get monitored
- Add your shiny new GCP Essentials badge to your resume!