

Google cloud platform project

Apache Web Server on Google Cloud

by Abdulaziz alqahtani

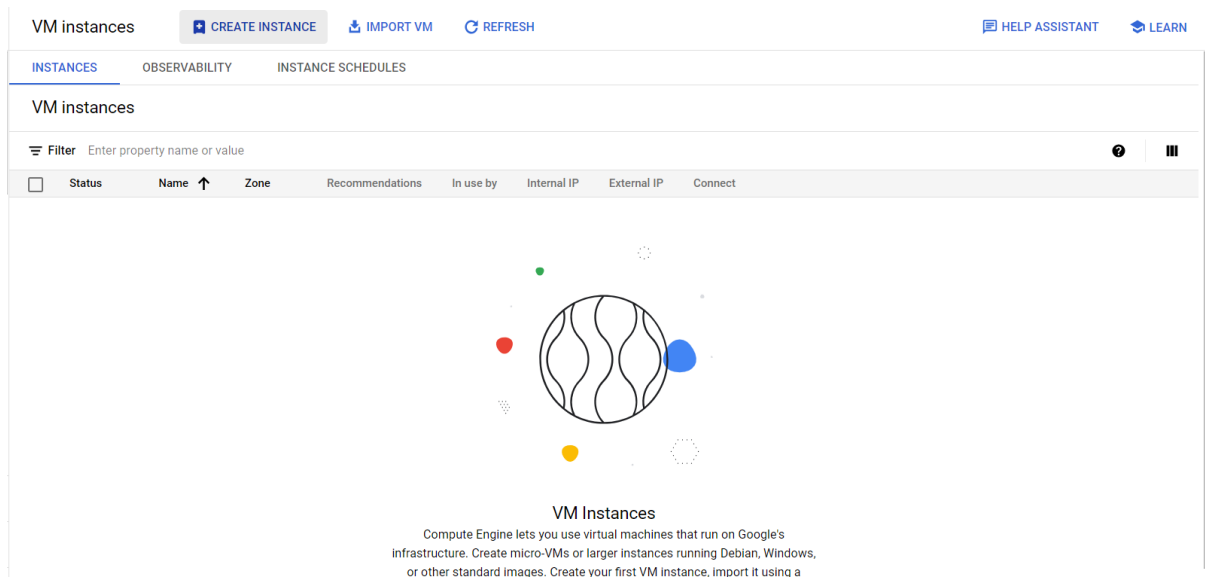
Guidelines:

- First make your own GCP account.
- second after you make your account enable compute engine
- third create your first virtual machine

Hands-on-deck \$:~


VM settings:

here you make your first VM.



The boot disk settings.

Boot disk ?

Name	instance-1
Type	New balanced persistent disk
Size	10 GB
License type ?	Free
Image	 Debian GNU/Linux 11 (bullseye)

Allow HTTP and HTTPS traffic.

Firewall ?

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

- ☒ Allow HTTP traffic
- ☒ Allow HTTPS traffic

VM SSH:

After we create our first VM, we'll run SSH from clicking the SSH button.


VM instances

Filter


Enter property name or value


?


☰


<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	✓	instance-1	me-central1-a			10.212.0.2 (nic0)	34.18.32.248 (nic0)	 SSH ⌵ ⋮


Related actions


 **Explore Backup and DR** NEW
Back up your VMs and set up disaster recovery


 **View billing report**
View and manage your Compute Engine billing

 **Monitor VMs**
View outlier VMs across metrics like CPU and network

 **Explore VM logs**
View, search, analyze, and download VM instance logs

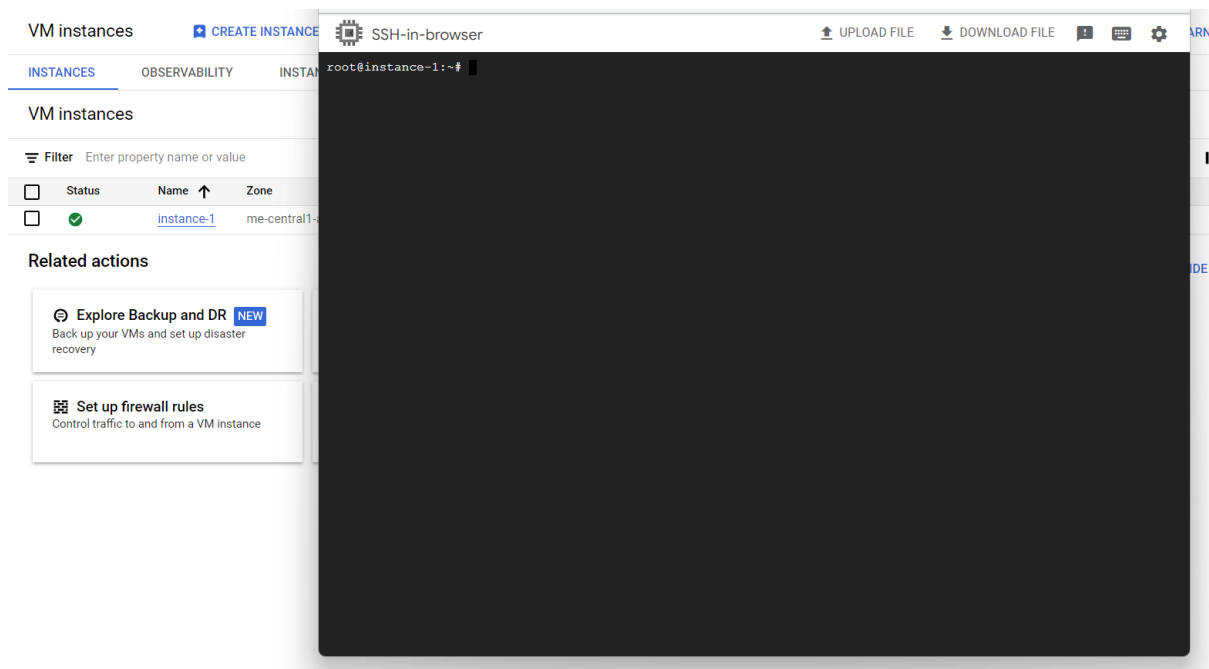
 **Set up firewall rules**
Control traffic to and from a VM Instance

 **Patch management**
Schedule patch updates and view patch compliance on VM Instances

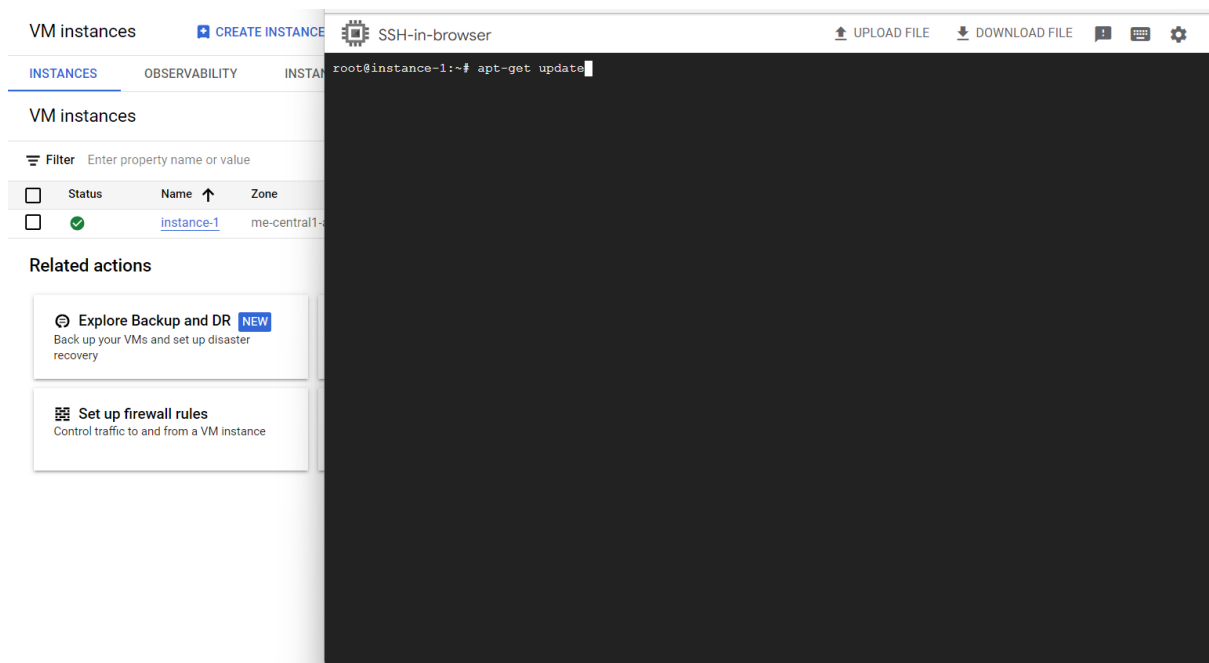
 **Load balance between VMs** [↗](#)
Set up Load Balancing for your applications as your traffic and users grow

[HIDE](#)

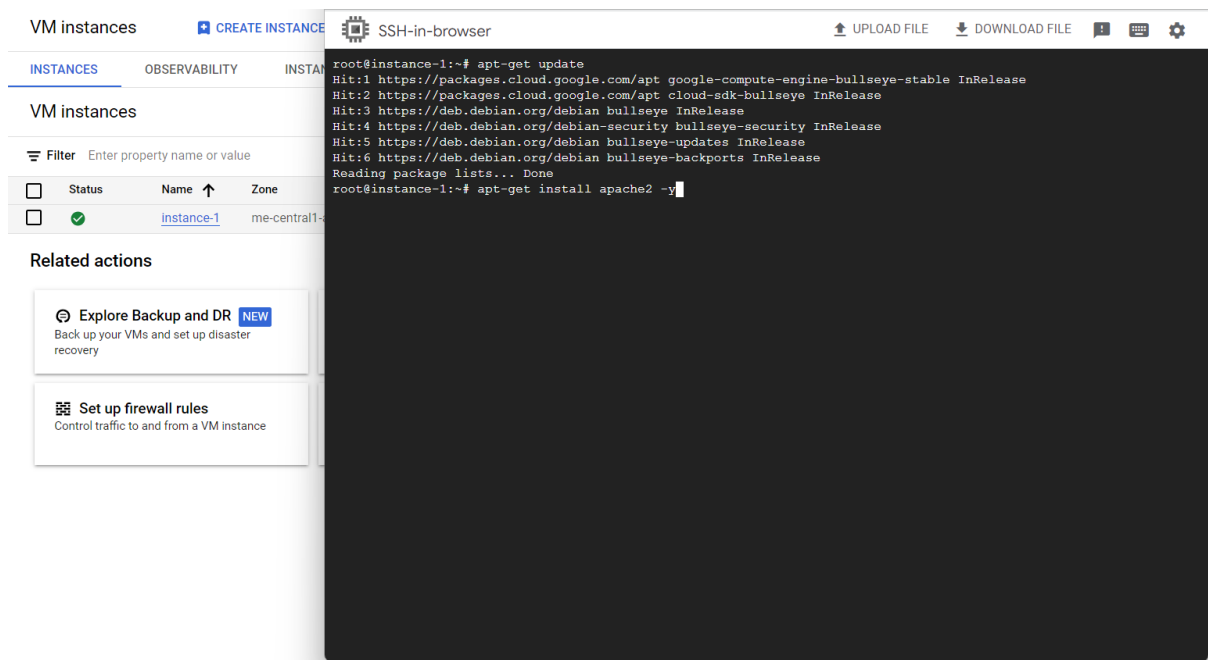
Now we open the SSH window.



First thing is to update packages by the cmd: **apt-get update**



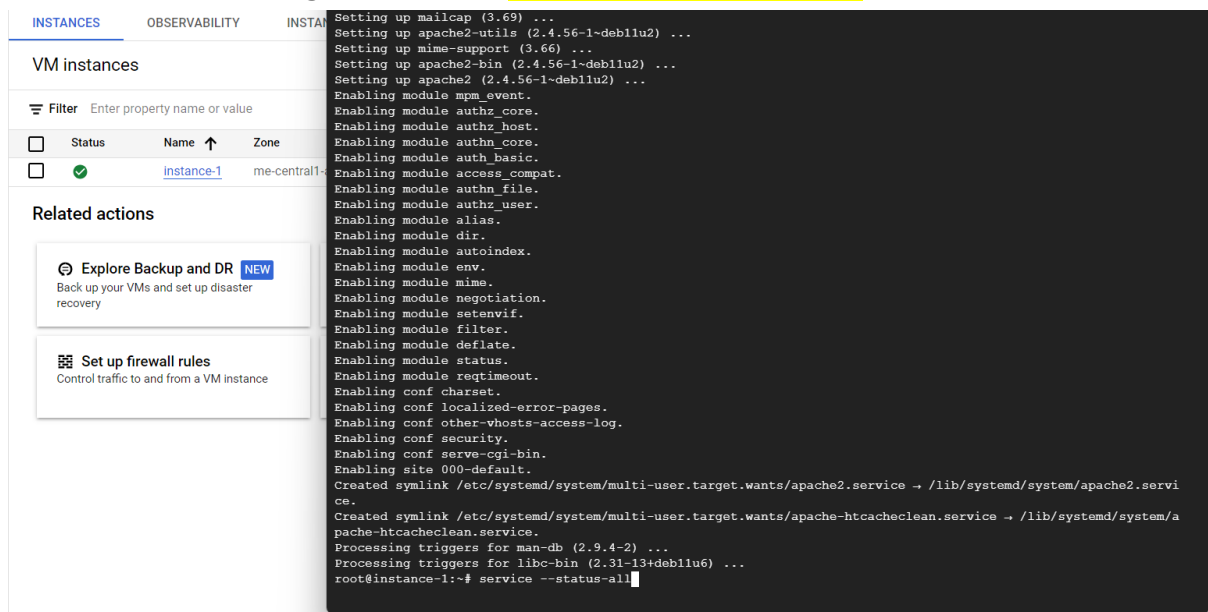
After that install apache2 server the cmd: `apt-get install apache2 -y`



The screenshot shows the Google Cloud Platform VM instances page. On the left, the 'VM instances' table lists 'instance-1' with a status of 'Running'. Below the table, there are 'Related actions' like 'Explore Backup and DR' and 'Set up firewall rules'. An 'SSH-in-browser' terminal window is open, showing the command `apt-get update` and `apt-get install apache2 -y` being executed. The terminal output shows the package lists being updated and the installation of apache2 being successful.

```
root@instance-1:~# apt-get update
Hit:1 https://packages.cloud.google.com/apt google-compute-engine-bullseye-stable InRelease
Hit:2 https://packages.cloud.google.com/apt cloud-sdk-bullseye InRelease
Hit:3 https://deb.debian.org/debian bullseye InRelease
Hit:4 https://deb.debian.org/debian-security bullseye-security InRelease
Hit:5 https://deb.debian.org/debian bullseye-updates InRelease
Hit:6 https://deb.debian.org/debian bullseye-backports InRelease
Reading package lists... Done
root@instance-1:~# apt-get install apache2 -y
```

we install apache2 server successfully but how to check if the server is working the cmd: `service --status-all`



The screenshot shows the Google Cloud Platform VM instances page with the same 'instance-1' listed. The 'SSH-in-browser' terminal window is open, showing the command `service --status-all` being executed. The terminal output shows the status of various services, including 'apache2' which is 'running'.

```
Setting up mailcap (3.69) ...
Setting up apache2-utils (2.4.56-1-deb11u2) ...
Setting up mime-support (3.66) ...
Setting up apache2-bin (2.4.56-1-deb11u2) ...
Setting up apache2 (2.4.56-1-deb11u2) ...
Enabling module mpm_event.
Enabling module authz_core.
Enabling module authz_host.
Enabling module authn_core.
Enabling module auth_basic.
Enabling module access_compat.
Enabling module authn_file.
Enabling module authz_user.
Enabling module alias.
Enabling module dir.
Enabling module autoindex.
Enabling module env.
Enabling module mime.
Enabling module negotiation.
Enabling module setenvif.
Enabling module filter.
Enabling module deflate.
Enabling module status.
Enabling module reqtimeout.
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vmhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /lib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service → /lib/systemd/system/apache-htcacheclean.service.
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for libc-bin (2.31-13+deb11u6) ...
root@instance-1:~# service --status-all
```

The check is completed.

The screenshot shows the AWS Management Console interface. On the left, the 'INSTANCES' tab is selected, displaying a table of VM instances. The table has columns for 'Status', 'Name', and 'Zone'. One instance, 'instance-1', is shown with a green status icon and is located in the 'me-central-1' zone. Below the table, there are 'Related actions' such as 'Explore Backup and DR' and 'Set up firewall rules'. On the right, a terminal window displays the output of the 'service --status-all' command, showing the status of various system services. The output indicates that several services are enabled and running, including 'apache-htcacheclean', 'apache2', 'apparmor', 'chrony', 'cron', 'dbus', 'exim4', 'haveged', 'hwclock.sh', 'kmod', 'networking', 'procps', 'rsyslog', 'screen-cleanup', 'ssh', 'sudo', 'udev', 'unattended-upgrades', and 'uuid'. The terminal prompt is 'root@instance-1:~#'.

Status	Name	Zone
✓	instance-1	me-central-1

```
Enabling module filter.
Enabling module deflate.
Enabling module status.
Enabling module reqtimeout.
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /lib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service → /lib/systemd/system/apache-htcacheclean.service.
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for libc-bin (2.31-13+deb11u6) ...
root@instance-1:~# service --status-all
[ + ] apache-htcacheclean
[ + ] apache2
[ + ] apparmor
[ + ] chrony
[ + ] cron
[ + ] dbus
[ + ] exim4
[ + ] haveged
[ - ] hwclock.sh
[ + ] kmod
[ + ] networking
[ + ] procps
[ + ] rsyslog
[ - ] screen-cleanup
[ + ] ssh
[ - ] sudo
[ + ] udev
[ + ] unattended-upgrades
[ - ] uuid
root@instance-1:~# ls
root@instance-1:~#
```

Note:

if you have index.html upload it and move it to /var/www/html

Finally after you upload your website now go to your external IP and click on it.

The screenshot shows the external IP address '34.18.32.248' (nic0) and a browser window displaying the Codegrid website. The website has a dark theme with a blue sidebar and a yellow footer. The main content area features the Codegrid logo, a navigation menu with links to 'Work', 'About us', and 'Contact us', and a large blue button labeled 'See our services'. The text 'ate about Web Design.' is partially visible on the left side of the page.

External IP: 34.18.32.248 (nic0)

Codegrid

ate about Web Design.

See our services

Work

About us

Contact us

I hope that the project will help you.