

Task 7: Configure Load Balancing and Auto Scaling for a Web Application

Load Balancer dashboard showing backend instances

The screenshot shows the Compute Engine interface under the 'Virtual machines' section. The left sidebar has 'Compute Engine' selected. Under 'Virtual machines', 'Instance templates' is selected. The main area displays an instance template named 'web-template'. The table shows the following details:

Name	Machine type	Image	Disk type	Location	Placement policy	Actions
web-template	r2-medium	debian-12-bookworm-x20231014	Balanced persistent disk	us-central1	No policy	

* Auto Scaling configuration

The screenshot shows the Compute Engine interface under the 'Virtual machines' section. The left sidebar has 'Compute Engine' selected. Under 'Virtual machines', 'Instance groups' is selected. The main area displays an instance group named 'web-group'. The table shows the following details:

Status	Name	Instances	Template	Group type	Creation time	Recommendation	Autoscaling
	web-group	0 → 1	web-template (Regional)	Managed	Nov 4, 2023, 9:49:19 PM UTC+05:30		No config

Public load balancer URL (working app)

The screenshot shows the 'Create global external application load balancer' wizard. Step 1: Backend. The left sidebar has 'Load balancing' selected. The main area shows the configuration for a load balancer named 'load'. The 'Backend' section includes:

- Backend services:
 - 1. back
- Frontend configuration (selected)
- Backend configuration (selected)
- Routing rules

At the bottom are 'Create', 'Cancel', and 'Equivalent code' buttons.

Short note (4–5 lines) describing setup and what happens during scaling

I created an Instance Template that installs Apache and a simple HTML page on startup, then used that template to create a Managed Instance Group with autoscaling (min=1, max=3, target CPU=60%). I configured an HTTP Load Balancer with a health check that routes traffic to the instance group. Under load the group automatically scales out additional instances and the load balancer distributes requests across healthy instances, ensuring availability and responsive scaling.