Using autohealing for highly available apps

This interactive tutorial shows how to use autohealing (/compute/docs/instance-groups#autohealing) to build highly available apps on Compute Engine.

Highly available apps are designed to serve clients with minimal latency and downtime. Availability is compromised when an app crashes or freezes. Clients of a compromised app can experience high latency or downtime.

Autohealing lets you automatically restart apps that are compromised. It promptly detects failed instances and recreates them automatically, so clients can be served again. With autohealing, you no longer need to manually bring an app back to service after a failure.

Objectives

- Configure a health check and an autohealing policy.
- Set up a demo web service on a managed instance group.
- Simulate health check failures and witness the autohealing recovery process.

Costs

This tutorial uses billable components of Google Cloud including:

- Compute Engine

Before you begin

Sign in (https://accounts.google.com/Login) to your Google Account.

If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).

In the Cloud Console, on the project selector page, select or create a Cloud project.

★ **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

Go to the project selector page (https://console.cloud.google.com/projectselector2/home/dashboar

Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project (/billing/docs/how-to/modify-project).

Enable the Compute Engine API.

Enable the API (https://console.cloud.google.com/flows/enableapi?apiid=compute_engine)

If you prefer to work from the command line, install the `gcloud` command-line tool.

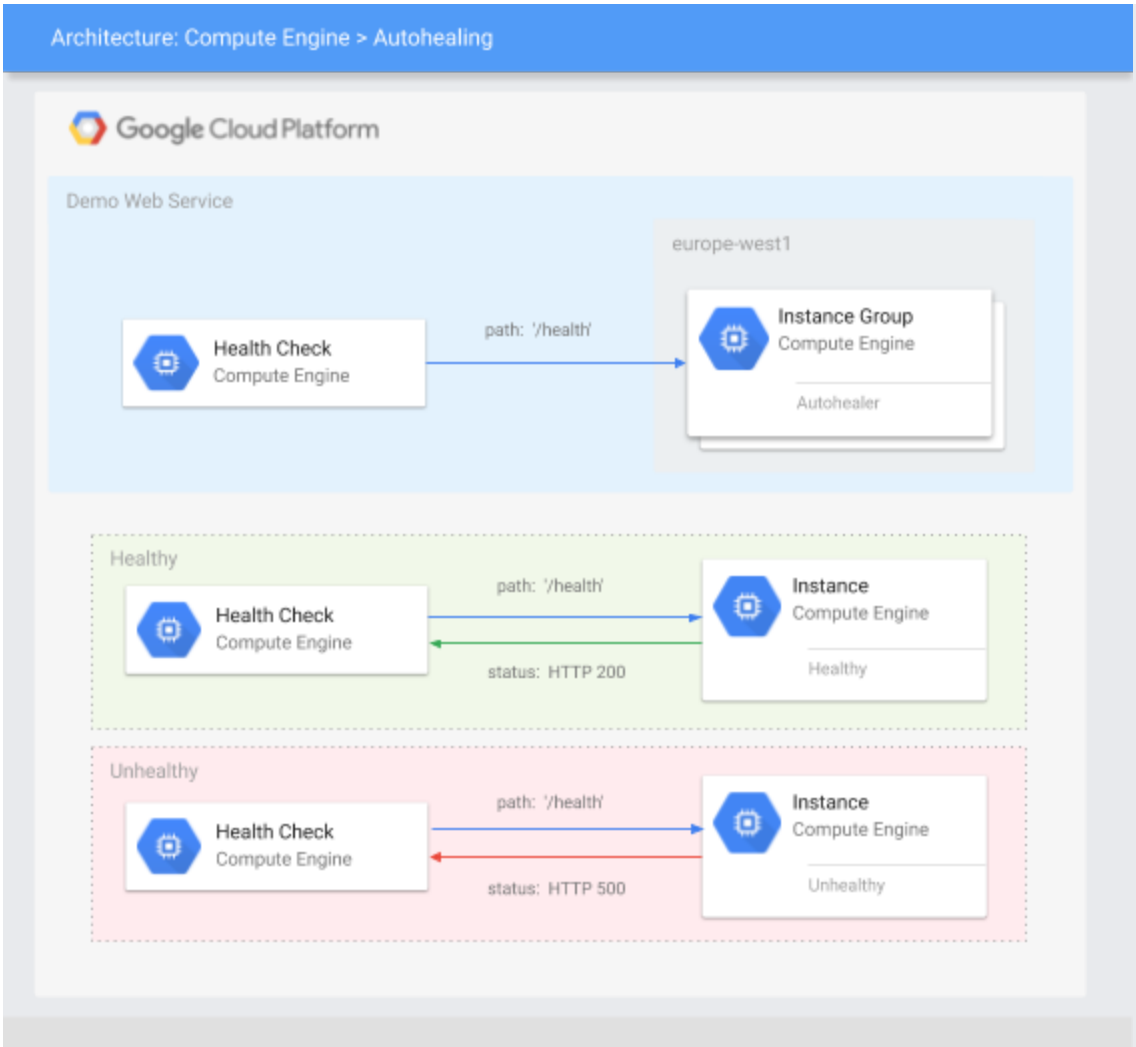
- Install and initialize the Cloud SDK (/sdk/docs).

★ **Note:** You can run the **`gcloud`** tool in the Cloud Console without installing the Cloud SDK. To run the **`gcloud`** tool in the Cloud Console, use Cloud Shell (https://console.cloud.google.com/home/dashboard?cloudshell=true).

App architecture

The app includes the following Compute Engine components:

- Health check (/load-balancing/docs/health-check-concepts): an HTTP health check policy used by the autohealer to detect failed VM instances.
- Firewall rules (/vpc/docs/firewalls): Google Cloud firewall rules let you allow or deny traffic to your instances.
- Managed instance group (/compute/docs/instance-groups#managed_instance_groups): A group of instances running the same demo web service.
- Instance template (/compute/docs/instance-templates): A template used to create each instance in the instance group.



How the health check probes the demo webservice

A health check sends probe requests to an instance using a specified protocol, such as HTTP(S), SSL, or TCP. For more information, see how health checks work (/load-balancing/docs/health-check-concepts#method) and health check categories, protocols, and ports (/load-balancing/docs/health-check-concepts#categories_protocols_ports).

The health check in this tutorial is an HTTP health check that probes the HTTP path `/health` on port 80. For an HTTP health check, the probe request passes only if the path returns an `HTTP 200 (OK)` response. For this tutorial, the demo web server defines the path `/health` to return an `HTTP 200 (OK)` response when healthy or an `HTTP 500 (Internal Server Error)` response when unhealthy. For more information, see success criteria for HTTP, HTTPS, and HTTP/2 (/load-balancing/docs/health-check-concepts#criteria-protocol-http).

Create the health check

To set up autohealing, create a custom health check and configure the network firewall to allow health check probes.

Pro Tip: Use separate health checks for load balancing and for autohealing (/compute/docs/instance-groups#autohealing). Health checks for load balancing detect unresponsive instances and direct traffic away from them. Health checks for autohealing detect and recreate failed instances, so they should be less aggressive than load balancing health checks. Using the same health check for these services would remove the distinction between unresponsive instances and failed instances, causing unnecessary latency and unavailability for your users.
For more information, see What makes a good autohealing health check (#what_makes_a_good_autohealing_health_check).

Consolegcloud (#gcloud)

1. Create a health check.
    a. Go to the **Health checks** page in the Cloud Console.
    Go to the Health checks page (https://console.cloud.google.com/compute/healthChecks)
    b. Click **Create health check**.
    c. Set **Name** to `autohealer-check`.
    d. For **Protocol** select `HTTP`.
    e. Set **Request path** to `/health`. This indicates what HTTP path the health check uses. For this tutorial, the demo web server defines the path `/health` to return either an `HTTP 200 (OK)` response when healthy or an `HTTP 500 (Internal Server Error)` response when unhealthy.
    f. Set the **Health criteria**:
        i. Set **Check interval** to `10`. This defines the amount of time from the start of one probe to the start of the next one.
        ii. Set **Timeout** to `5`. This defines the amount of time that Google Cloud waits for a response to a probe. This value must be less than or equal to the check interval.
        iii. Set **Healthy threshold** to `2`. This defines the number of sequential probes that must succeed for the instance to be considered healthy.

iv. Set **Unhealthy threshold** to `3`. This defines the number of sequential probes that must fail for the instance to be considered unhealthy.

g. Click **Create** at the bottom.

2. Create a firewall rule to allow health check probes to make HTTP requests.

a. Go to the **Create firewall rule** page in the Cloud Console.

Go to the Create firewall rule page (https://console.cloud.google.com/networking/firewall

b. For **Name**, enter `default-allow-http-health-check`.

c. For **Network**, select `default`.

d. For **Targets**, select `All instances in the network`.

e. For **Source filter**, select `IP ranges`.

f. For **Source IP ranges**, enter `130.211.0.0/22` and `35.191.0.0/16`.

g. In **Protocols and ports**, select **tcp** and enter `80`.

h. Click **Create**.

★ **Note:** Health check probes come from addresses in the ranges **`130.211.0.0/22`** and **`35.191.0.0/16`**. For this tutorial, your health check uses the **`HTTP`** protocol, so make sure the firewall rule allows connections to port 80. For more information, see setting up health checking and autohealing for managed instance groups (/compute/docs/instance-groups/autohealing-instances-in-migs).

What makes a good autohealing health check

Health checks used for autohealing should be conservative so they don't preemptively delete and recreate your instances. When an autohealer health check is too aggressive, the autohealer might mistake busy instances for failed instances and unnecessarily restart them, reducing availability.

- `unhealthy-threshold`. Should be more than `1`. Ideally, set this value to `3` or more. This protects against rare failures like a network packet loss.
- `healthy-threshold`. A value of `2` is sufficient for most apps.
- `timeout`. Set this time value to a generous amount (five times or more than the expected response time). This protects against unexpected delays like busy instances or a slow

network connection.

- `check-interval`. This value should be between 1 second and two times the timeout (not too long nor too short). When a value is too long, a failed instance is not caught soon enough. When a value is too short, the instances and the network can become measurably busy, given the high number of health check probes being sent every second.

Set up the web service

This tutorial uses a web app that is stored on GitHub. If you would like learn more about how the app was implemented, see the GoogleCloudPlatform/python-docs-samples GitHub repository (https://github.com/GoogleCloudPlatform/python-docs-samples/tree/master/compute/managed-instances/demo).

To set up the demo web service, create an instance template that launches the demo web server on startup. Then, use this instance template to deploy a managed instance group and enable autohealing.

Consolegcloud (#gcloud)

1. Create an instance template. Include a startup script that starts up the demo web server.
    a. Go to the **Instance templates** page in the Cloud Console.
    Go to the Instance templates page (https://console.cloud.google.com/compute/instanceT
    b. Click **Create instance template**.
    c. Set the **Name** to `webserver-template`.
    d. For **Machine configuration** select `micro` (f1-micro).
    e. Under **Firewall**, select the **Allow HTTP traffic** checkbox.
    f. Click **Management, security, disks, networking, sole tenancy** to reveal advanced settings. Several tabs appear.
    g. On the **Management** tab, find **Automation** and enter the following **Startup script**:

```
sudo apt update && sudo apt -y install git gunicorn3 python3-pip
git clone https://github.com/GoogleCloudPlatform/python-docs-samples
cd python-docs-samples/compute/managed-instances/demo
sudo pip3 install -r requirements.txt
sudo gunicorn3 --bind 0.0.0.0:80 app:app --daemon
```

h. Click **Create**.

2. Deploy the web server as a managed instance group.

a. Go to the **Instance groups** page in the Cloud Console.

Go to the Instance groups page (https://console.cloud.google.com/compute/instanceGrou

b. Click **Create instance group**.

c. Set the **Name** to `webserver-group`.

d. For **Region**, select `europe-west1`.

e. For **Zone**, select `europe-west1-b`.

f. For **Instance template**, select `webserver-template`.

g. For **Autoscaling**, select **Off**.

h. Set **Number of instances** to `3`.

i. For **Health check**, select `autohealer-check`.

j. Set **Initial delay** to `90`.

k. Click **Create**.

★ **Note: Initial delay** indicates how long to wait before sending the first health check. A new instance needs to be fully running and ready to respond with a 'healthy' response. For this sample webserver in this tutorial, 90 seconds should be enough time.

3. Create a firewall rule that allows HTTP requests to the web servers.

a. Go to the **Create firewall rule** page in the Cloud Console.

Go to the Create firewall rule page (https://console.cloud.google.com/networking/firewall

b. For **Name**, enter `default-allow-http`.

c. For **Network**, select `default`.

d. For **Targets**, select `Specified target tags`.

e. For **Target Tags**, enter `http-server`.

f. For **Source filter**, select `IP ranges`.

g. For **Source IP ranges**, enter `0.0.0.0/0`.

h. In **Protocols and ports**, select **tcp** and enter `80`.

i. Click **Create**.

Simulate health check failures

To simulate health check failures, the demo web server provides ways for you to force a health check failure.

Consolegcloud (#gcloud)

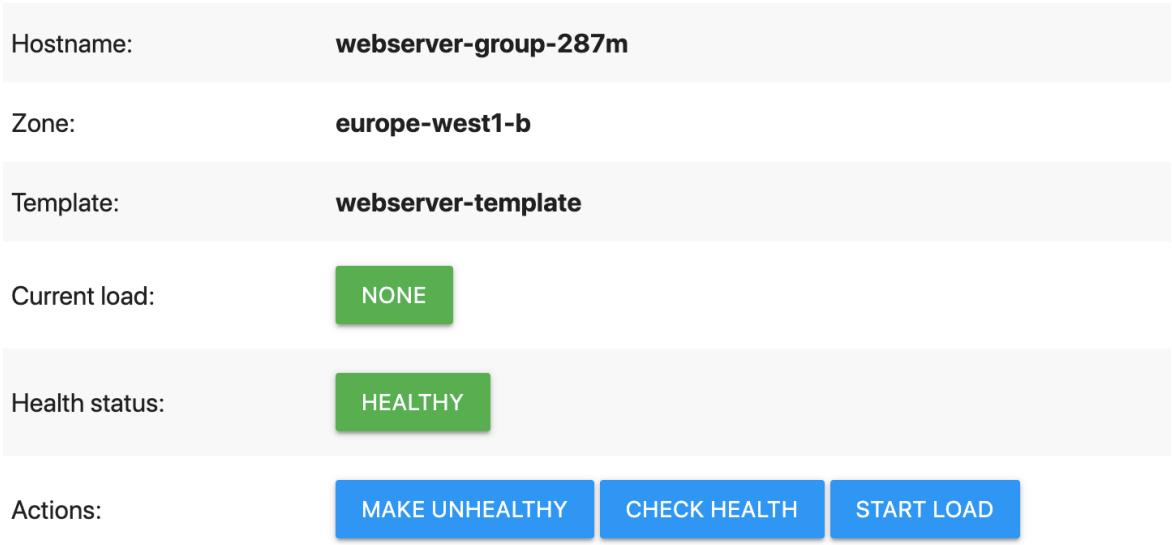
1. Navigate to a web server instance.

   a. Go to the **VM instances** page in the Cloud Console.

   Go to the VM instances page (https://console.cloud.google.com/compute/instances)

   b. Under the **External IP** column, click the IP address for any `webserver-group` instance. A new tab opens in your web browser. If the request times out or the web page is not available, wait a minute to let the server finish setting up and try again.

The demo web server displays a page similar to the following:



2. On the demo web page, click **Make unhealthy**.

   This causes the web server to fail the health check. Specfically, the web server makes the `/health` path return an `HTTP 500 (Internal Server Error)`. You can verify this yourself by quickly clicking the **Check health** button (this stops working after the autohealer has started rebooting the instance).

3. Wait for the autohealer to take action.

   a. Go to the **VM instances** page in the Cloud Console.

   Go to the VM instances page (https://console.cloud.google.com/compute/instances)

   b. Wait for the status of the web server instance to change. The green checkmark next to the instance name should change to a grey square, indicating that the autohealer has started rebooting the unhealthy instance.

   c. Click **Refresh** at the top of the page periodically to get the most recent status.

   d. The autohealing process is finished when the grey square changes back to a green checkmark, indicating the instance is healthy again.

Feel free to repeat this exercise. Here are some ideas:

- What happens if you make all instances unhealthy at one time? For more information about autohealing behavior during concurrent failures, see autohealing behavior (/compute/docs/instance-groups/autohealing-instances-in-migs#autohealing_behavior).
- Can you update the health check configuration to heal instances as fast as possible? (In practice, you should set the health check parameters to use conservative values as

explained in this tutorial. Otherwise, you may risk instances being mistakenly deleted and restarted when there is no real problem.)

- The instance group has an `initial delay` configuration setting. Can you determine the minimum delay needed for this demo web server? (In practice, you should set the delay to somewhat longer (10%–20%) than it takes for an instance to boot and start serving app requests. Otherwise, you risk the instance getting stuck in an autohealing boot loop.)

View autohealer history (optional)

To view a history of autohealer operations use the following `gcloud` command:

```
gcloud compute operations list --filter='operationType~compute.instances.repair.*'
```

For more information, see viewing historical autohealing operations (/compute/docs/instance-groups/autohealing-instances-in-migs#viewing_historical_autohealing_operations)

Cleaning up

After you've finished the autohealing tutorial, you can clean up the resources that you created on Google Cloud so they won't take up quota and you won't be billed for them in the future. The following sections describe how to delete or turn off these resources.

If you created a separate project for this tutorial, delete the entire project. Otherwise, if the project has resources that you want to keep, only delete the specific resources created in this tutorial.

Deleting the project

Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.

- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an `appspot.com` URL, delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

   Go to the Manage resources page (https://console.cloud.google.com/iam-admin/projects)

2. In the project list, select the project that you want to delete and then click **Delete** .
3. In the dialog, type the project ID and then click **Shut down** to delete the project.

Deleting specific resources

If you can't delete the project used for this tutorial, delete the tutorial resources individually.

Deleting the instance group

consolegcloud (#gcloud)

1. In the Cloud Console, go to the **Instance groups** page.

   Go to the Instance groups page (https://console.cloud.google.com/compute/instanceGroups/lis

2. Click the checkbox for your `webserver-group` instance group.
3. Click **Delete** to delete the instance group.

Deleting the instance template

Note: You must delete the instance group before deleting the instance template. You can't delete an instance template if a managed instance group uses it.

consolegcloud (#gcloud)

1. Go to the **Instance templates** page in the Cloud Console.

   Go to the Instance templates page (https://console.cloud.google.com/compute/instanceTemplat

2. Click the checkbox next to the instance template.
3. Click **Delete** at the top of the page. In the new window, click **Delete** to confirm the deletion.

Deleting the health check

Note: You must delete the instance group before deleting the health check. You can't delete a health check if other resources use it.

consolegcloud (#gcloud)

1. Go to the **Health checks** page in the Cloud Console.

   Go to the Health checks page (https://console.cloud.google.com/compute/healthChecks)

2. Click the checkbox next to the health check.
3. Click **Delete** at the top of the page. In the new window, click **Delete** to confirm the deletion.

Deleting the firewall rules

consolegcloud (#gcloud)

1. Go to the **Firewall rules** page in the Cloud Console.

   Go to the Firewall page (https://console.cloud.google.com/networking/firewalls)

2. Click the checkboxes next to the firewall rules named `default-allow-http` and `default-allow-http-health-check`.
3. Click **Delete** at the top of the page. In the new window, click **Delete** to confirm the deletion.

What's next

- Try another tutorial:
  - Using load balancing for highly available applications (/compute/docs/tutorials/high-availability-load-balancing).
  - Using autoscaling for highly scalable applications (/compute/docs/tutorials/high-scalability-autoscaling).
- Learn more about managed instance groups (/compute/docs/instance-groups).
- Learn more about designing robust systems (/compute/docs/tutorials/robustsystems).
- Learn more about building scalable and resilient web apps on Google Cloud (/solutions/scalable-and-resilient-apps).

Last updated 2020-07-22 UTC.