# Reparación automática (Autohealing)


Elaborada por: M. en C. Ukranio Coronilla

**Nota:** Para esta práctica es necesario un nuevo proyecto y en el mismo crear un bucket conteniendo el primer servidor HTTP en Java visto en clase en formato JAR (`WebServer.jar`). Adjunte los ítems solicitados en un archivo pdf y envíelos para acreditar la presente práctica.

## Autohealing

Una de las razones de implementar un sistema distribuido es mantener una alta disponibilidad. Para lograr este objetivo GCP ofrece un sistema de recuperación automática de fallos (autohealing), el cual detecta si una instancia deja de funcionar correctamente. Al ser detectada GCP la reinicia o la reemplaza sin intervención manual optimizando así el mantenimiento. Veremos a continuación como lo podemos implementar.

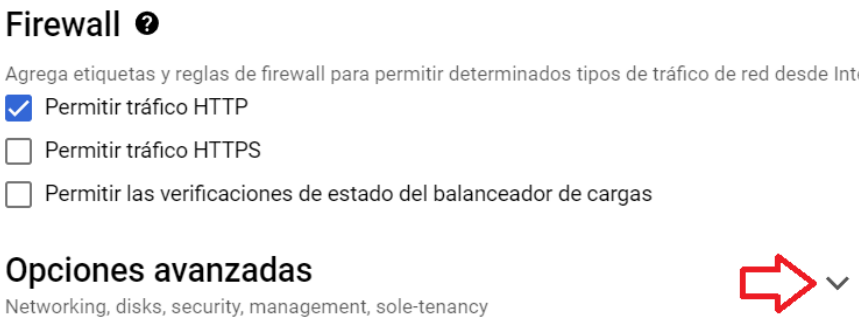
## Plantilla de instancias

En GCP una plantilla de instancias (instance template) es un recurso que define una configuración específica para la creación de una instancia. Esta configuración especifica el tipo de máquina (e2-medium, n2-standard-4, etc.), sistema operativo (Ubuntu, Debian, etc.), disco de almacenamiento (tamaño, HDD o SDD), si requiere ejecutarse algún código adicional al iniciar el sistema (startup scripts), la red VPC a la cual se conectará la instancia, reglas de firewall, entre otras opciones disponibles.

Las ventajas son que al utilizar una plantilla de instancias podemos automatizar la creación de n instancias y se garantiza que todas las instancias tendrán la misma configuración. También se puede actualizar la plantilla y aplicar los cambios de manera controlada a un grupo de instancias, y finalmente puedo reutilizarlas para distintos proyectos.

Vamos a crear una plantilla de instancias usando la consola GCP, para lo cual accedemos a la consola de Google, creamos un nuevo proyecto, almacenamos el archivo jar del servidor HTTP en un bucket cuyo nombre anotamos y posteriormente le damos click a la hamburguesa ☰ en la esquina superior izquierda para ver el panel (dashboard) y seleccionamos: **Compute Engine-> Plantillas de Instancia**.

Posteriormente damos click en CREAR PLANTILLA DE INSTANCIAS y en el apartado nombre le ponemos como nombre **plantilla-instancias-prueba** a nuestra plantilla, seleccionamos la región, por ejemplo, us-central1, el tipo de máquina que queramos, en este caso seleccionamos de la serie N1 la **f1-micro** de **Núcleo compartido**. En el disco de arranque dejamos seleccionado el sistema operativo Debian (Debian GNU/Linux 12 (bookworm)), en el firewall permitimos el tráfico HTTP y le damos click a la flecha que abre el menú de opciones avanzadas como se muestra:




Posteriormente damos click en el menú de **Gerenciamiento**, con lo que se muestra:

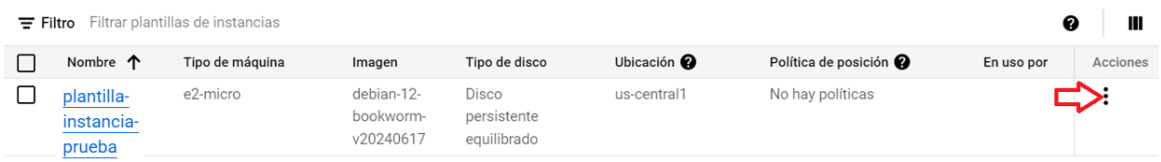
Dentro del campo **Automatización** vamos a colocar la siguiente secuencia de comandos donde especifica que se utilizará el intérprete de comando bash, el cual ejecutará una actualización de la lista de paquetes de software disponibles en los repositorios oficiales, instala el JRE de Java en su versión 17 y sin hacer preguntas, crea el directorio destino si no existe, copia el archivo WebServer.jar en formato JAR del bucket a la instancia (cambie el nombre del bucket en amarillo por el suyo y en verde escriba la ruta home de su instancia ), espera hasta que se haga la copia del archivo antes de continuar, da permisos de ejecución al archivo por si es necesario y finalmente ejecuta el archivo JAR en el puerto 80, enviando la salida del comando al archivo webserver.log por si se requiere conocerla para depuración. Esta secuencia de comandos se va a ejecutar automáticamente cada que se inicie o reinicie nuestra instancia:

```
#!/bin/bash

# Actualiza la lista de paquetes
apt update

# Instala OpenJDK 17
apt -y install openjdk-17-jre-headless

# Crea el directorio destino si no existe
mkdir -p /home/ukraniocc

# Copia el archivo JAR desde el bucket
gsutil cp gs://bucket-reparacion/WebServer.jar /home/ukraniocc/

# Espera hasta que el archivo exista
while [ ! -f /home/ukraniocc/WebServer.jar ]; do
  echo "Esperando a que WebServer.jar esté disponible..."
  sleep 1
done

# Da permisos de ejecución por si se requiere
chmod +x /home/ukraniocc/WebServer.jar

# Ejecuta el archivo JAR
nohup java -jar /home/ukraniocc/WebServer.jar 80 > /home/ukraniocc/webserver.log 2>&1 &
```

Por último, damos click en 

, y entonces nos aparece la plantilla creada en la lista de plantillas disponibles.

Vamos a probar nuestra plantilla de instancias creando una instancia para verificar que funciona correctamente. Al final del renglón de la plantilla creada damos click en los tres puntos que marca la flecha y seleccionamos **Crear VM**:



No modificamos nada más sólo le damos click en 

en la parte inferior de la página, con lo cual se crea la instancia y se ejecuta la secuencia de comandos, lo cual tarda aproximadamente un par de minutos (depende de la carga de trabajo, los recursos disponibles y el tráfico de red dentro de GCP) por lo que tendrá que esperar a que eso suceda. Pasado el tiempo podemos copiar la dirección IP de la instancia creada, y usarla con curl en su LAP para acceder al endpoint **/status**, pruébelo (A veces no se levanta el servicio aun después de los dos minutos, por lo que se recomienda entrar con SSH para revisar si se han ejecutado los comandos comenzando con la instalación de java, la copia del archivo jar y si ya se ha ejecutado apareciendo en la tabla de procesos. Si no es así tendremos que esperar aún más tiempo mientras seguimos intentando con curl). Es posible que el sitio no se ejecute correctamente dentro de la red Wifi de la ESCOM, pero en su celular usando los datos debería funcionar.

Envíe la captura de pantalla donde se observe la plantilla creada, así como la terminal de su computadora ejecutando curl como prueba. (**Item 1**)

Al terminar de hacer la prueba es importante borrar la instancia que acabamos de crear. También podemos borrar la plantilla de instancias, pero en este caso no lo haremos porque la vamos a utilizar posteriormente además de que no tiene ningún costo asociado su existencia.

## Grupo de instancias administrado MIG

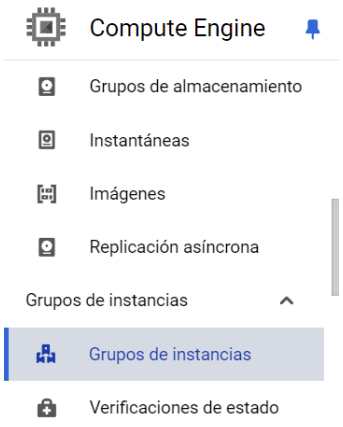
Un grupo de instancias administrado o MIG (Managed Instance Group) es una colección de instancias que se administran como si fuera un solo recurso. Este grupo se genera con una plantilla de instancias y permite incorporar diversas características deseables en un sistema distribuido, por ejemplo:

- Escalabilidad automática (autoscaling) con lo cual se aumenta o disminuye el número de instancias en función de la demanda de CPU, tráfico de red o métricas personalizadas.
- Reparación automática (autohealing) con el cual se hacen verificaciones periódicas del estado de las instancias. Si alguna no pasa la verificación se le elimina y se crea una nueva instancia de forma automática.
- Balanceo de carga HTTP(S)
- Distribución de las instancias en diversas zonas de la misma región mejorando la resiliencia y la tolerancia a fallos.

- Actualizaciones automáticas con lo cual se puede actualizar la plantilla de instancias y aplicar esta nueva configuración a todo el grupo de manera automática.

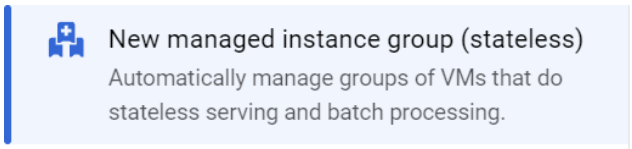
Ahora vamos a crear un grupo de instancias administrado (se requiere haber creado antes una plantilla de instancias) y también vamos a probar la auto reparación (autohealing).

En el recurso Compute Engine de GCP damos click en Grupos de instancias:

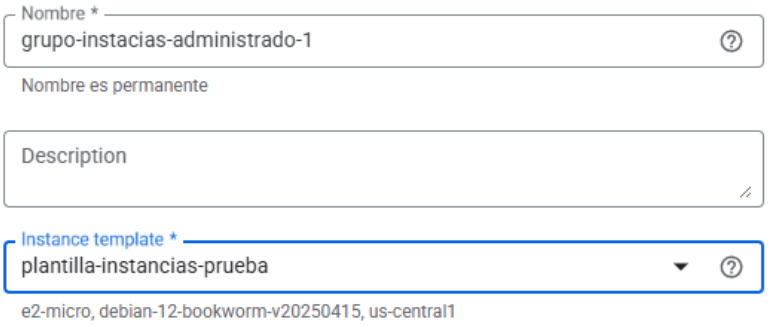



Posteriormente damos click en 

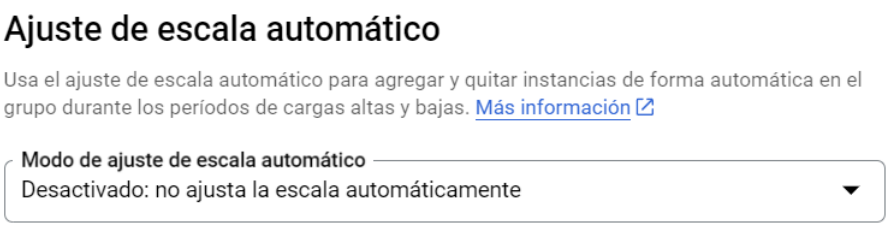
donde por default queda seleccionada la creación de un nuevo grupo de instancias administrado sin estado:




Le ponemos un nombre y seleccionamos la plantilla que hemos creado con anterioridad:



En **Cantidad de instancias** vamos a especificar que se conserven dos instancias en el grupo, pero para ello será necesario antes desactivar el **Ajuste de escala automático** que aparece más abajo en la página como se muestra:

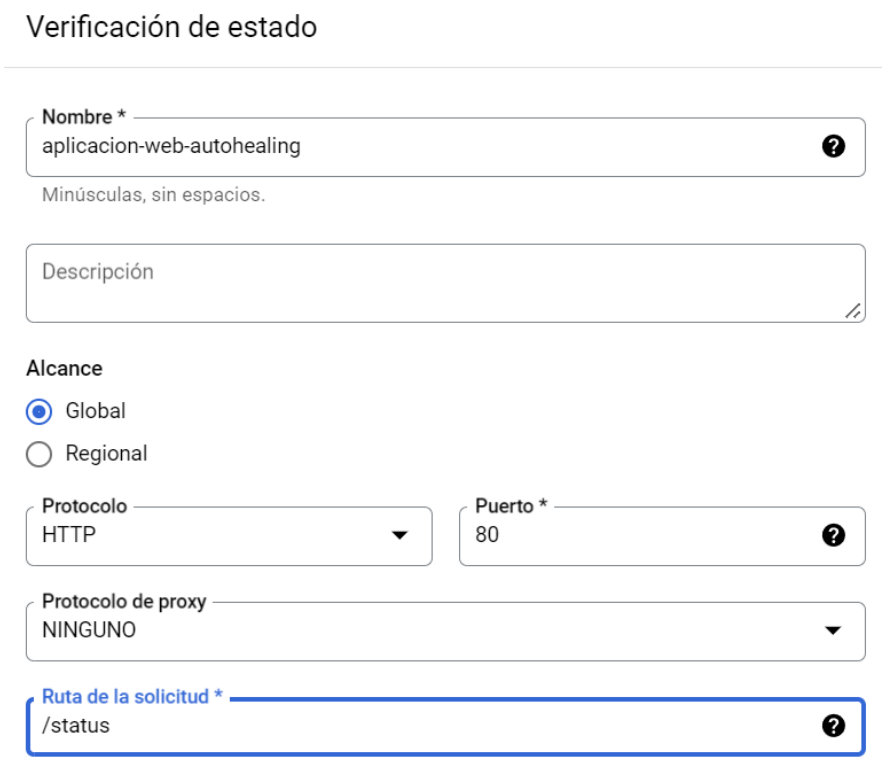



Para mejorar la disponibilidad podemos solicitar que utilice varias zonas de la región configurada en la plantilla, aunque en este caso le dejaremos en **Zona única**:

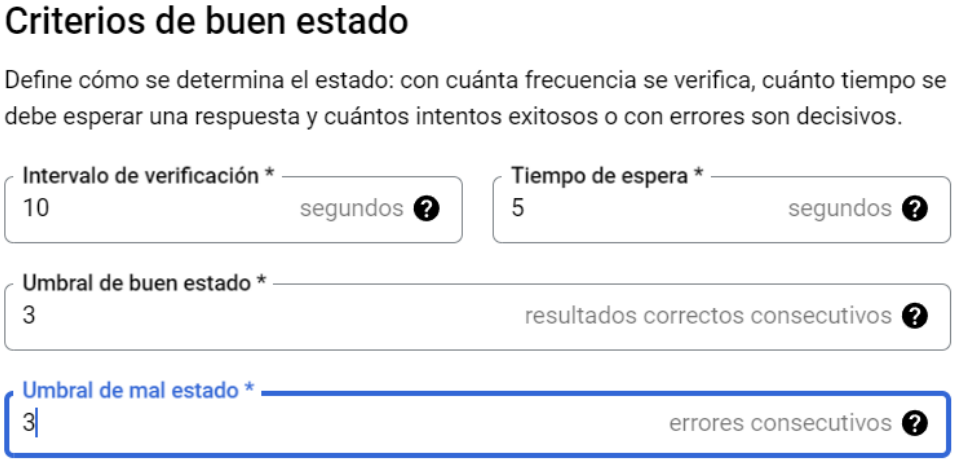

Ahora nos deslizamos en la página hasta llegar a **Reparación automática**, damos click en


y seleccionamos Crear una comprobación de estado :

Aparece entonces una ventana de **Verificación de estado** donde vamos a indicarle la manera de saber si una instancia está dañada, le damos un nombre, seleccionamos el protocolo como HTTP, el puerto 80 y el endpoint **/status** que ya tenemos programado en nuestro servidor HTTP:

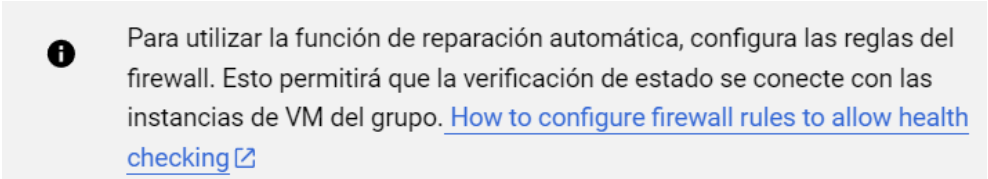



Abajo en los **Criterios de buen estado** ponemos que haga la verificación cada 10 segundos, que espere 5 segundos por una respuesta del endpoint **/status**, que haga 3 chequeos consecutivos a partir de los cuales se considera saludable la instancia o 3 chequeos consecutivos a partir de los cuales se considera no saludable la instancia:




Al final damos click en 

para mantener esta configuración, aunque nos va a aparecer el mensaje:

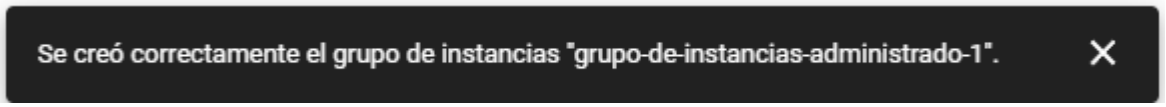



en nuestro caso cuando creamos la plantilla ya habíamos configurado la regla de firewall que permite el tráfico HTTP por lo que no necesitamos hacer otro cambio.

Posteriormente en **Retraso inicial** que es el tiempo de espera en segundos antes de iniciar las pruebas de autohealing dejaremos el valor de 300 segundos (5 minutos) por default, pero si tardan mucho en levantarse los servicios tendremos que incrementarlo posteriormente.

Finalmente, para crear nuestro grupo de instancias administrado al final de la página damos click en 

y esperamos el mensaje que nos indica que se ha creado el grupo de instancias administrado:




Esto podemos verificarlo dando click en  Instancias de VM.

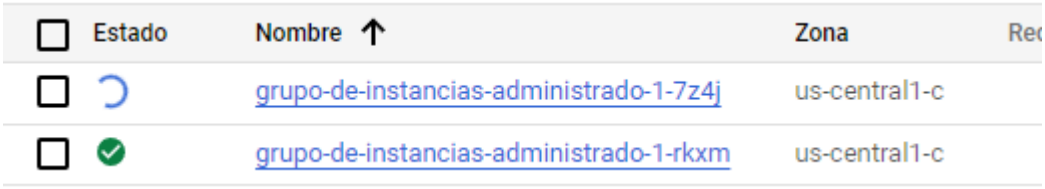
A partir de este momento intentamos acceder al endpoint **/status** con curl en las dos instancias creadas, las cuales tendrían que estar disponibles antes de que transcurran los cinco minutos en los que se va a iniciar el proceso de autohealing.

Para verificar la reparación automática primero accedemos al endpoint **/status** de una de las instancias para ver si el servidor está vivo. Posteriormente vamos a matar al servidor HTTP simulando un fallo, para lo cual abrimos la terminal SSH de dicha instancia y matamos el proceso asociado con el programa java que ejecuta nuestra aplicación como se muestra (adjuntar captura **item 2**):

```
ukraniocc@instance-group-ukranio-nf20:~$ ps -A | grep java
   2023 ?        00:00:13 java
ukraniocc@instance-group-ukranio-nf20:~$ sudo kill -9 2023
```

salimos de la terminal SSH y verificamos con curl que nuestro servidor ya no está corriendo, aunque aparece la instancia como activa. Sólo esperamos un tiempo para verificar que el reparador automático detecta la inactividad de la instancia e inicia el proceso de dar de baja la instancia considerada defectuosa y la reemplaza con una nueva con una IP distinta. Podrá observar que en 

el estado de la instancia no se estará ejecutando como se muestra:




Mande una captura como la anterior donde el **Estado** de una de las instancias se muestra con el semicírculo azul y que incluya la IP inicial de dicha instancia, así como otra captura donde la IP ha cambiado para esta instancia pero la IP de la otra instancia permanece sin cambios para comprobar este paso (**Item 3**).

GCP no cobra por los grupos de instancias administrados como entidades, pero si hay costos asociados a los recursos utilizados, por ejemplo, se cobra por el uso de las instancias que fueron creadas. En el caso del autohealing no hay un costo directo por habilitar esta característica.