Cloud & Kubernetes

Workshop

Pablo González Troyano

Seguridad y Alta Disponibilidad

2º ASIR

Diciembre 2021

ASIR.SYAD.K8S.2.KeyDoc.PabloGonzález

Foto de portada: Kubernetes logo

Índice

Cloud, ¿una moda? ☁️ **5**
Mitos sobre la nube 5
A la larga es más caro 5
La nube no es segura 5
La nube es limitante 5
La nube sirve para todo 6
¿Dónde queda el personal de IT? 6
Casos de uso y clientes 6
Principios de escalabilidad y alta disponibilidad 7
Diferencias entre generaciones de infraestructuras 8
Es lo que se conoce como “Infrastructure as Code”: 9
Para continuar la investigación: 9
Despliegues **10**
Apache balanceado (antes: GCP Access & VM) 10
Crear grupo de instancias 10
Ajustes de escalado automático 11
Prueba de estado 12
Crear una plantilla de instancias 14
Regla de Firewall (no siempre necesaria) 14
Comprobaciones de las máquinas 15
Creación de un bucket de GCS 16
Balanceador de carga 18
Definición y tipos 18
Creación del balanceador de carga 20
Comprobaciones del balanceador de carga y pruebas de estrés 23
Nextcloud Docker compose 29
Kuber... ¿qué? **31**
¿Contenedores? ¿De reciclaje dices? 31
Así funciona kubernetes 33
Componentes / Capas de abstracción 33
Deployment 33
ReplicaSet 34
Pod 34
Cluster Level 34
Cluster 34
Worker Node 35
Cluster Master 35
Node Processes 35
Master Components 35
Worker Node Components 36
Docker Container Level 36
SatefulSet 37
DaemonSet 37
Servicios 37
Volum-cosas 37
Volumes 37
PersistentVolumes and PersistentVolumeClaims 37
Resumen de Kubernetes 38
Despliegues con Kubernetes **39**
hello-app en GKE 39
Libro de visitas con Redis y PHP 43
Wordpress con PersistentDisk y CloudSQL 46
Otras opciones 47
Bibliografía **49**

Cloud, ¿una moda? ☁️

Algunos expertos lo definen como “un modelo de entrega y consumo de servicios”. Realmente la definición de este concepto variará según a quién preguntes, a qué empresa pertenece, qué puesto ocupa y la generación a la que pertenezca.

Debemos diferenciar entre el prestador de servicios y el consumidor de estos. Como prestadores encontramos grandes tecnológicas: Amazon (Web Services), Microsoft (Azure), IBM (Cloud), Alibaba (Cloud), OVH (Cloud), Google (Cloud Platform)...

En esencia, nos permite acceder a servicios **bajo demanda**, sin tener que soportar el coste del equipamiento físico, configuración y despliegue. La nube es *el Netflix de la informática*.

La gran mayoría de empresas utilizan, de una forma u otra, aplicaciones y sistemas en la nube. Bien como infraestructura de computación y servicios, o herramientas de colaboración y comunicación.

Mitos sobre la nube

A la larga es más caro

¿Tienes capacidad para comprarte un avión? No, pero puedes pagar un asiento en uno y viajar a casi cualquier parte del mundo. Un Boeing 767-300ER tiene un coste de 171 millones de euros[1]. Si adoptamos como precio medio de un vuelo 50€, podríamos realizar 3.420.000 vuelos hasta amortizar el coste del avión, sin tener en cuenta combustible, costes de mantenimiento, personal, burocracia, etc.

Lo mismo ocurre con la nube. ¿Puede una empresa permitirse un despliegue si necesita comprar todo el equipamiento, cuando ni siquiera sabe si su negocio es viable?

La nube no es segura

Los grandes de la computación en la nube invierten miles de millones en mantener sus redes y equipamiento seguro. La seguridad dependerá en gran medida de la confianza que tengamos en la empresa que nos presta el servicio. Estas empresas dependen en gran medida de su reputación, no se pueden permitir escándalos de seguridad pues perderían el atractivo y sus clientes.

La nube es limitante

Si algo no es la nube es limitante. Permite tener una escalabilidad casi ilimitada. Pone a nuestra disposición todos los recursos que podamos necesitar (y pagar, claro). Permite tener hoy 1 servidor y mañana 100, pasado mañana 1.000 y la semana que

[1] ¿Cuánto cuesta comprar un avión? - Periódico La Vanguardia - Consultado el 13 nov 2021

viene tener desplegar todo en mediante microservicios y *server-less computing* en 20 localizaciones globales distintas. Sin tocar ni un solo servidor.

Respecto a la posibilidad de cambiar de proveedor, dejando a un lado la complejidad técnica de una migración, siempre es posible. En cada proveedor tendrá un nombre *catchy* distinto: EC2[2] vs Compute Engine[3], Amazon Simple Storage Service (Amazon S3)[4] vs Google Cloud Storage[5]. Pero en esencia es el mismo servicio, distinto panel de administración, funciones similares.

La nube sirve para todo

La nube es realmente útil para muchas cargas de trabajo y casuísticas, pero no para todas. Antes de migrar a la nube es necesario hacer un estudio de impacto y costes.

¿Dónde queda el personal de IT?

Las informáticas e informáticos no sobramos a la hora de implantar los servicios de la nube. De hecho, es una de las ramas con mayor crecimiento. Hay especializaciones para cada proveedor. Los especialistas deben conocer con exactitud las posibilidades y limitaciones de los servicios, la manera de securizarlos y asegurar la disponibilidad.

Los *Cloud Architects*[6] permiten a las organizaciones aprovechar las tecnologías en la nube. Con un conocimiento profundo de la arquitectura de la nube diseñan, desarrollan y administran soluciones sólidas, seguras, escalables, de alta disponibilidad y dinámicas para impulsar los objetivos comerciales.

Casos de uso y clientes

Entre los muchos grandes clientes[7] de la infraestructura Cloud encontramos:

- PayPal aprovecha la nube para acelerar su proceso de desarrollo con cargas de trabajo basadas en contenedores. Además, es *regulatory compliant* gracias a la localización de los centros de datos.
- Airbus entrena modelos de inteligencia artificial con Tensorflow y Cloud Vision para identificar las imágenes de sus 14 satélites. Utiliza Kubernetes para el proceso de cargas de trabajo de forma distribuida, Cloud Datastore para el almacenamiento de datos y Cloud Storage para el almacenamiento y distribución de sus imágenes.
- Carrefour migró su plataforma monolítica a una arquitectura de microservicios basada en la nube y contenedores. Además, migró sus 700 Tb de datos a la nube.

[2] AWS | Elastic compute cloud (EC2) de capacidad modificable en la nube
[3] Compute Engine: Virtual Machines (VMs)
[4] Almacenamiento de datos seguro en la nube (S3)
[5] Cloud Storage
[6] Professional Cloud Architect Certification
[7] Google Cloud Customer Voices

Almacenarlo en sus propios servidores no le permitía escalar. A día 8 de abril de 2021, crecía a ritmo de 2 Tb diarios.

Principios de escalabilidad y alta disponibilidad

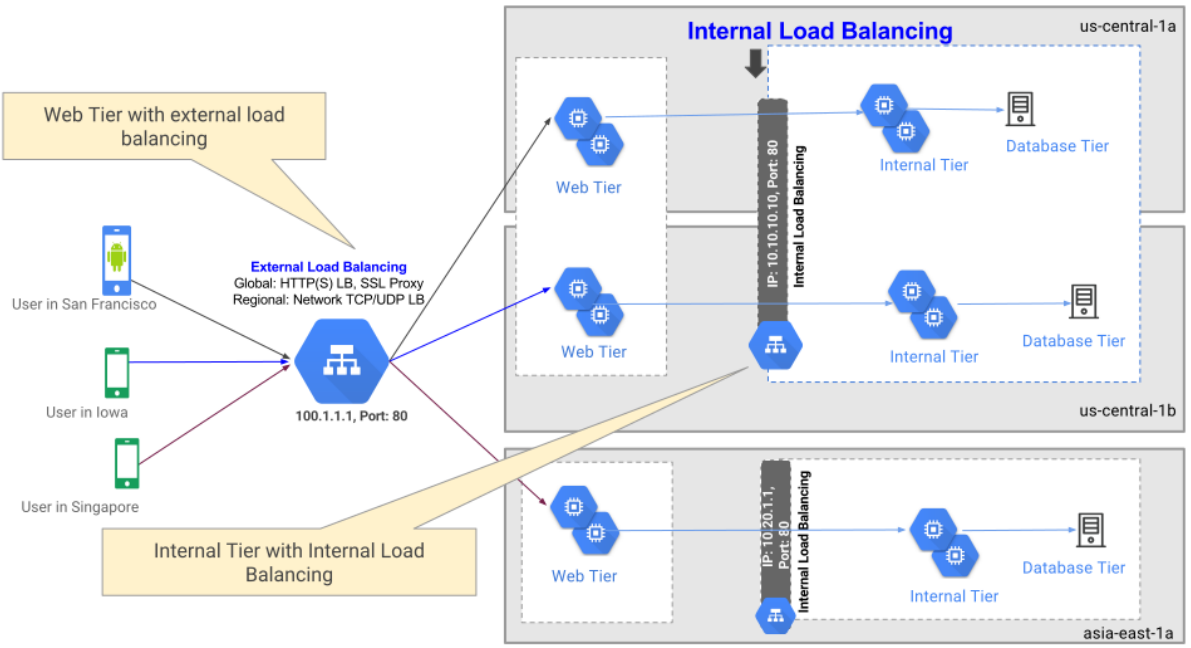
Gracias a las tecnologías *Cloud*, disponer de diferentes capacidades de computación es sencillo.

Es posible ampliar un cluster en cuestión de minutos, distribuir el tráfico entre máquinas virtuales, en distintas regiones geográficas con un par de clics o llamadas a API. Incluso es posible hacerlo de forma automática en base a la demanda del servicio.

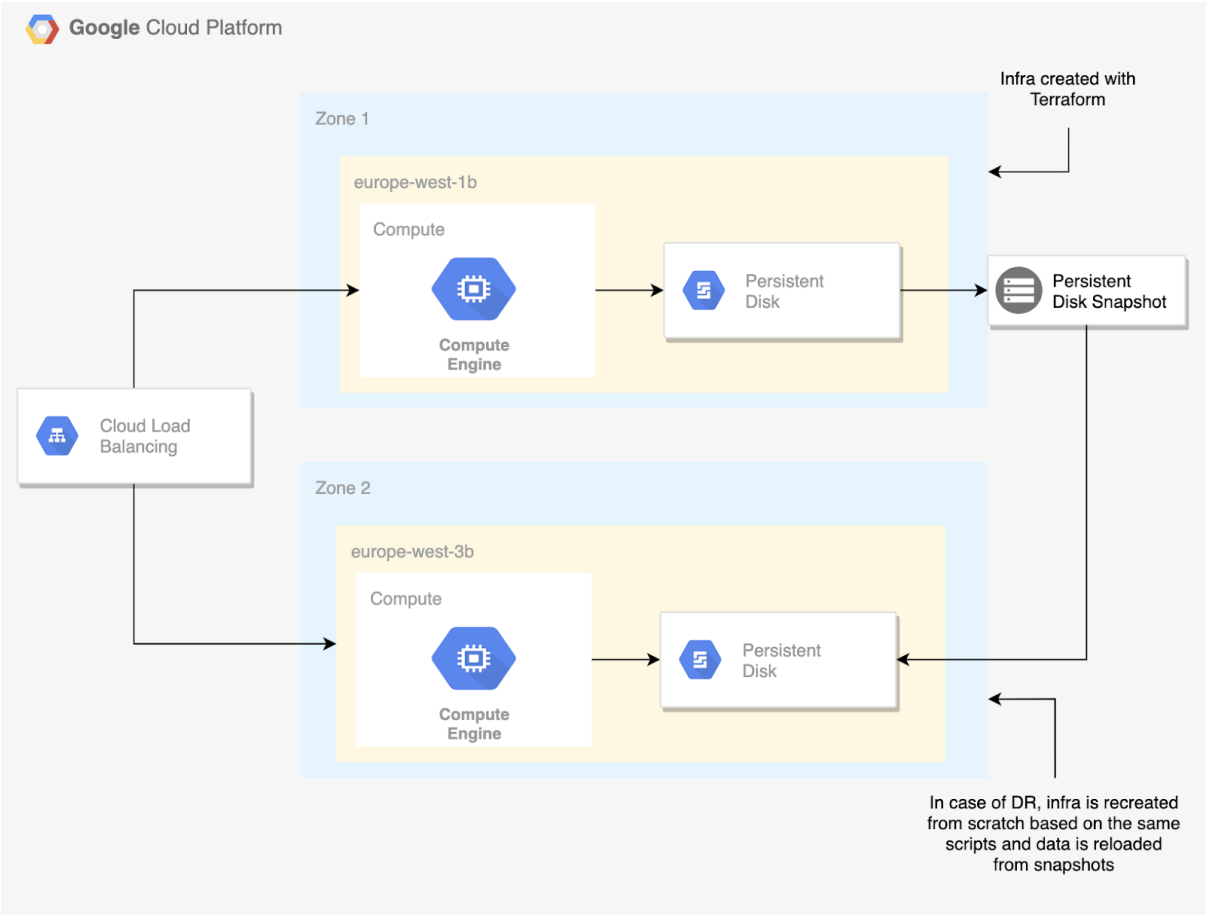
De hecho, es posible no administrar servidores ni contenedores con las tecnologías conocidas como "Server-less".

Cloud Run permite disponer de la capacidad necesaria (sin importar cuán alta sea) en cuestión de *ms* para acciones preprogramadas, Cloud Spanner y Big Query ofrecen una capacidad de ingesta y tratamiento de datos sin igual. A nivel de PB para el almacenamiento y TB para el proceso de datos en cuestión de minutos. Si bien es posible lograr rendimientos *similares* con Hadoop, la escalabilidad y continuidad es más difícil de lograr.

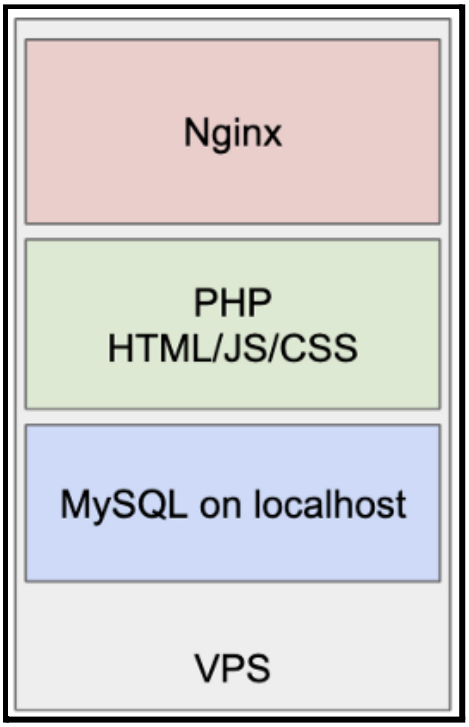
Veamos una posible infraestructura sencilla que permite la Alta Disponibilidad y la escalabilidad:

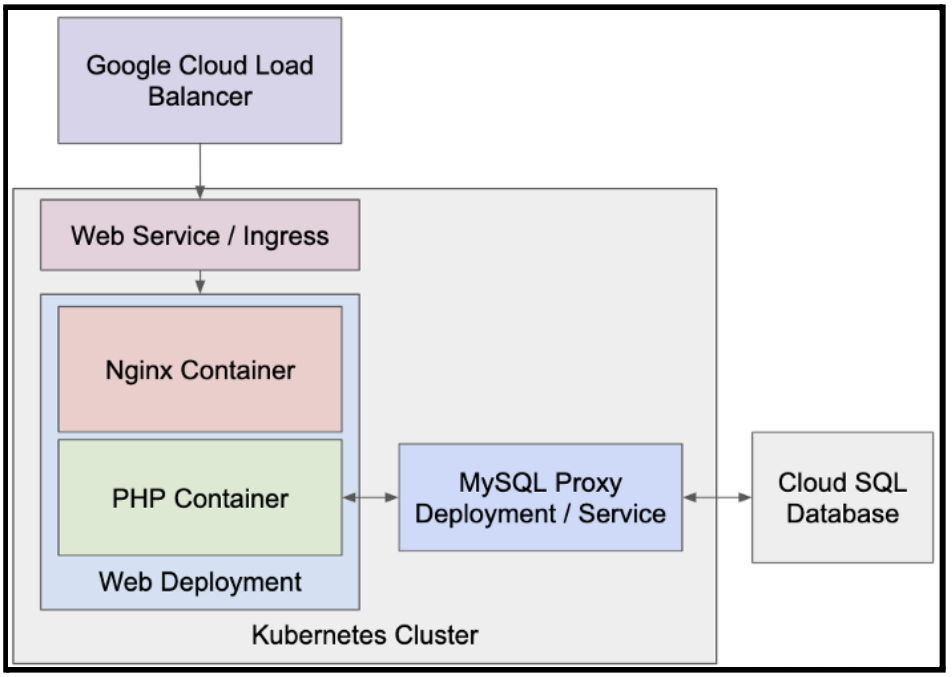


Veamos otra, a nivel regional. **Muy importante** la coletilla de *Infra created with Terraform*.



Diferencias entre generaciones de infraestructuras





Es lo que se conoce como “Infrastructure as Code”:

- ¿Qué es la infraestructura como código (IaC)?
- Forma en la que la infraestructura como código (IaC) gestiona infraestructuras complejas

Para continuar la investigación:

- 5 principles for cloud-native architecture—what it is and how to master it
    - Principle 1: Design for automation
        - Automatizar la creación de infraestructura.
        - CI/CD. Automatizar la implementación de nuevas versiones de las infraestructura.
        - Escalado automática en base a métricas de demanda.
        - Monitorización y recuperación automática ante errores.
    - Principle 2: Be smart with state
        - ***Scale****: To scale up, just add more copies. To scale down, instruct instances to terminate once they have completed their current task.*
        - ***Repair****: To 'repair' a failed instance of a component, simply terminate it as gracefully as possible and spin up a replacement.*
        - ***Roll-back****: If you have a bad deployment, stateless components are much easier to roll back, since you can terminate them and launch instances of the old version instead.*
        - ***Load-Balance across****: When components are stateless, load balancing is much simpler since any instance can handle any request. Load balancing across stateful components is much harder, since the state of the user's session typically resides on the instance, forcing that instance to handle all requests from a given user.*
    - Principle 3: Favor managed services
    - Principle 4: Practice defense in depth
    - Principle 5: Always be architecting
- Want repeatable scale? Adopt infrastructure as code on GCP
- Site Reliability Engineering (Libros y metodología)
- Cloud Run: What no one tells you about Serverless (and how it's done)
- Open data lakehouse on Google Cloud
- How Lowe's improved incident response processes with SRE
- Where should I run my stuff? Choosing a Google Cloud compute option
- What is Compute Engine? Use cases, security, pricing and more
- Google Cloud: Cloud Computing Services
- Cloud Services - Amazon Web Services (AWS)
- Microsoft Azure: Servicios de informática en la nube
- Web hosting, cloud y servidores dedicados
- Reliable Secure Cloud Solutions to Empower Your Global Business
- ¿Qué es IBM Cloud?

Despliegues

Apache balanceado *(antes: GCP Access & VM)*

Crearemos un grupo de instancias, en las que instalaremos automáticamente Apache, modificando el index de la web para indicar el hostname de la máquina que sirve el contenido.

Creamos un bucket para el alojamiento estático de recursos, donde subiremos algunas imágenes.

Definiremos un balanceador de carga de nivel OSI 7 delante de las máquinas virtuales. Crearemos reglas que permitan redirigir el tráfico a las distintas máquinas virtuales y al bucket creado.

Crear grupo de instancias[8]

Antes de nada debemos definir qué entendemos por instancia. Una instancia es una máquina virtual *corriendo* en Google Cloud Platform.

Agrupar las instancias nos permite escalarlas, asegurar la disponibilidad y actualizar las instancias que la confirman como si fueran un *ente* único.

Hacemos clic en el menú de GCP > Compute Engine > Grupos de Instancias

[8] https://cloud.google.com/compute/docs/instance-groups

Tenemos 3 posibles grupos de instancias. Viendo las características de cada una de ellas, decidimos que la que más se ajusta a nuestras necesidades es el *grupo de instancias administrado (sin estado)*.[9]

Para la creación del grupo de instancias, debemos definir al menos los siguientes valores:

- **Nombre**, el nombre es permanente e identificará inequívocamente al grupo de instancias en todo el proyecto. Será el valor que debamos utilizar para llamar a las APIs. Debe ser lo más descriptivo posible. Como ejemplo, podemos utilizar *mig-apache-balanceado*.
- Si bien la **descripción** no es necesaria puede ser interesante definirla si vamos a trabajar con más personas.
- **Ubicación**, para asegurar la resiliencia de nuestro despliegue, seleccionamos varias zonas como destino. Elegimos la región más cercana a nuestros usuarios.
- **Plantilla de instancias**, con los grupos de instancias, definimos que queremos mantenerlas y escalarlas, entre otros. Pero debemos indicar también cómo queremos crear las instancias. Si no tenemos ninguna definida, debemos crearla. Las instrucciones detalladas se encuentran en la sección crear una plantilla de instancias.

Ajustes de escalado automático

Marcamos, por supuesto, la opción de *ajustar escala automática*. En la *política de ajuste de escala automático* configuramos los umbrales para el escalado.

Podemos escalar el grupo en base a:

- Uso de CPU
- Capacidad de entrega de un balanceador de cargas de HTTP(S) externo
- Los datos almacenados en Cloud Monitoring

Lo más sencillo es elegir el escalado en base al uso de CPU. Si queremos verlo en directo, podemos definir un uso bajo de procesador.

Para el periodo de enfriamiento, debemos definir suficiente como para que a la máquina le dé tiempo a crearse y configurarse. De ser muy reducido, corremos el riesgo de enviar peticiones de los clientes a endpoints que todavía no pueden procesarlas. 60-120 segundos puede ser suficiente, aunque dependerá de la aplicación.

En tanto al número de instancias que formarán parte del grupo, definiremos 2 como mínimo y 4 como máximo, o lo que sea conveniente para nuestra aplicación. La recomendación para los grupos de instancias multizonales (también llamados regionales) es definir al menos 2 instancias para que se distribuyan en las distintas zonas (mejorando así la disponibilidad y la alta disponibilidad de los servicios, en caso de posibles caídas zonales).

[9] https://www.redhat.com/es/topics/cloud-native-apps/stateful-vs-stateless

Podemos activar los *controles de escalamiento*, que permitirán limitar el escalado automática a un porcentaje de crecimiento o cantidad por periodo de tiempo. Esto evitará gastos imprevistos en caso de errores en la configuración (las instancias se terminan y se crean indefinidamente, por ejemplo). Debemos tener en cuenta que limitar el escalado del grupo puede suponer problemas de disponibilidad en el caso de que la demanda sufra un aumento repentino no contemplado.

Prueba de estado

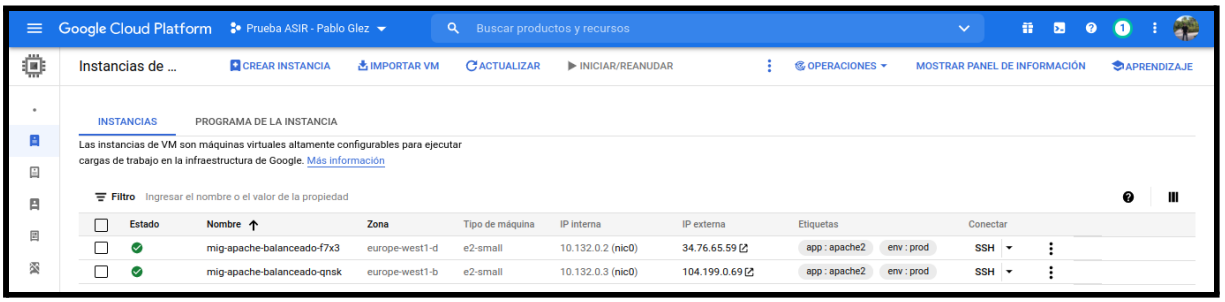
Un grupo que tenga muchas instancias, pero sin ninguna funcionar es un grupo no funcional. Para asegurarnos de que las instancias funcionan correctamente, es altamente recomendable crear comprobaciones de estado.
En base a estas comprobaciones de estado, el sistema podrá "matar" las instancias con problemas y crear otras para sustituirlas.

Definiremos, como para todos los recursos en Google Cloud Platform, un nombre para la comprobación de estado.

Como ejemplo, podemos utilizar *statecheck-mig-apache-balanceado*. Teniendo en cuenta que en los servidores estamos ejecutando un servidor web, tiene sentido utilizar peticiones TCP/HTTP el puerto 80 o el 443.

Podemos definir una respuesta esperada, encabezados personalizados, etc. En tanto a los criterios, definimos los tiempos de verificación y espera; así como los umbrales de estado correcto e incorrecto.

Al hacer clic en crear el grupo, situado en la parte inferior, el sistema procederá a crearlo:



Google Cloud Platform | Prueba ASIR - Pablo Glez | Buscar productos y recursos

Crear un grupo de instancias

Para crear un grupo de instancias, selecciona una de las siguientes opciones:

Nuevo grupo de instancias administrado (sin estado)
Para cargas de trabajo de entrega sin estado y cargas de trabajo por lotes.
Admite:
- ajuste de escala automático, reparación automática, actualización automática
- implementación en varias zonas
- balanceo de cargas

Nuevo grupo de instancias administrado (con estado)
Para cargas de trabajo con estado, como bases de datos.
Admite:
- conservación de discos y metadatos
- reparación y actualización automáticas
- implementación de varias zonas
- balanceo de cargas

Crea un nuevo grupo de instancias no administrado
Un grupo de VM que administras tú mismo.
Admite:
- balanceo de cargas

Organiza instancias de VM en grupos para administrarlas conjuntamente.
Grupos de instancias

Nombre
El nombre es permanente
mig-apache-balanceado

Descripción (Opcional)

Ubicación
Para asegurar una mayor disponibilidad, selecciona una ubicación de varias zonas para un grupo de instancias. Más información

- Zona única
- Varias zonas
  Solo los grupos de instancias administradas pueden existir en varias zonas.

Región
La región es permanente
europe-west1

Zonas
- europe-west1-b
- europe-west1-c
- europe-west1-d

Menos

Forma de distribución objetivo
Solo puedes establecer la forma de distribución objetivo durante la creación del grupo de instancias. No podrás cambiarla después.
- Uniforme
  Las instancias administradas se distribuyen de manera uniforme entre las zonas
- Equilibrada
  Distribuye las instancias administradas de la forma más uniforme posible en las zonas según la disponibilidad de recursos en cada zona
- Cualquiera
  Las instancias administradas se implementan en una o varias zonas en función de la disponibilidad de recursos en cada zona.

Redistribución de instancias
- Habilitar redistribución de instancias

Especificar la asignación del nombre de puerto (Opcional)

Plantilla de instancias
plantilla-apache-balanceado

Un grupo de instancias creado desde una plantilla con una VM interrumpible no se comportará como otros grupos de instancias.
Más información

Cantidad de instancias
Según la configuración del ajuste de escala automático

Ajuste de escala automático
Usa el ajuste de escala automático para permitir el cambio de tamaño automático de este grupo de instancias en los periodos de cargas altas y bajas.
Ajuste de escala automático de grupos de instancias

Modo de ajuste de escala automático
Ajustar escala automáticamente

Política de ajuste de escala automático
Usa las métricas y los programas para determinar cuándo realizar el ajuste de escala automático en el grupo. El ajuste de escala automático siempre necesita al menos un indicador, ya sea una métrica o un programa.
Política del ajuste de escala automático y uso objetivo

Uso de CPU: 60 % (predeterminado)

+ Agregar métrica nueva

+ Add new scaling schedule

Ajuste de escala automático predictivo
Usa el ajuste de escala automático predictivo para predecir futuras necesidades de capacidad en función de las cargas anteriores de CPU. Más información
- Desactivado
  No se usa un método predictivo. El escalador automático ajustará la escala del grupo para cumplir la demanda actual en función de las métricas en tiempo real.
- Optimizar para lograr mayor disponibilidad
  El ajuste de escala automático predictivo mejora la disponibilidad, ya que supervisa los patrones de carga de CPU diarios y semanales, y realiza un escalamiento horizontal en función de la demanda prevista.

Para habilitar el escalador automático predictivo, tu grupo debe haber tenido una configuración del ajuste de escala automático de al menos 3 días.

Periodo de enfriamiento
Especifica cuánto tarda tu app en inicializarse desde el inicio hasta que está lista para la entrega. Periodo de enfriamiento
60 segundos

Número mínimo de instancias
2

Número máximo de instancias
4

Para maximizar la disponibilidad, la cantidad mínima de instancias debe ser al menos igual a la cantidad de zonas. Se colocarán instancias adicionales en diferentes zonas.
Distribución de instancias mediante grupos de instancias administradas regionales

Controles de escalamiento
Evita una reducción inesperada en la cantidad de instancias de VM en el grupo. Para ello, controla el proceso de escalamiento. Más información

- Habilitar controles de escalamiento

Borrar configuración del ajuste de escala automático

Menos

Reparación automática
Verificación de estado
statecheck-mig-apache-balanceado (HTTP)
puerto: 80, tiempo de espera: 5 s, intervalo de verificación: 20 s, umbral de mal estado: 3 intentos

Retraso inicial
300 segundos

Para utilizar la función de reparación automática, configura las reglas del firewall. Esto permitirá que la verificación de estado se conecte con las instancias de VM del grupo.
Cómo configurar las reglas del firewall para permitir las verificaciones de estado

Se usará tu crédito de la prueba gratuita para las instancias de VM de este grupo.
Nivel gratuito de GCP

Crear | Cancelar

REST o línea de comandos equivalente

Crear una plantilla de instancias[10]

La plantilla nos permite replicar configuraciones de forma sencilla. No importa si son 1 o 1.000 máquinas virtuales. Para crearla, definimos algunos datos:

- Como **nombre**, indicamos por ejemplo, *plantilla-apache-balanceado*,
- Debemos también indicar de qué **tipo** serán las máquinas que creemos. Pensando en nuestra carga, elegimos *e2-small*.
- Escogemos un **disco de arranque** de 10 GB en modo persistente equilibrado (podríamos elegir SSD, aunque no es estrictamente necesario). En este instalamos Ubuntu 18.4 LTS.
- Permitimos el tráfico HTTP y HTTPS.
- Definimos las etiquetas *env:prod* y *app:apache2*, que nos ayudarán a organizar los recursos.
- Activamos la *preemptibilidad* de la máquina.
- Como startup-script indicamos:

```
#! /bin/bash
MAQUINITA=$(curl
http://metadata.google.internal/computeMetadata/v1/instance/hostname -H
"Metadata-Flavor: Google")
ZONA=$(curl
http://metadata.google.internal/computeMetadata/v1/instance/zone -H
"Metadata-Flavor: Google")
apt update
apt -y install apache2
cat <<EOF > /var/www/html/index.html
<html><body><h1>Hello World</h1>
<p>Instancia: $MAQUINITA</p>
<p>Zona: $ZONA </p>
</body></html>
EOF
```

Regla de Firewall (no siempre necesaria)

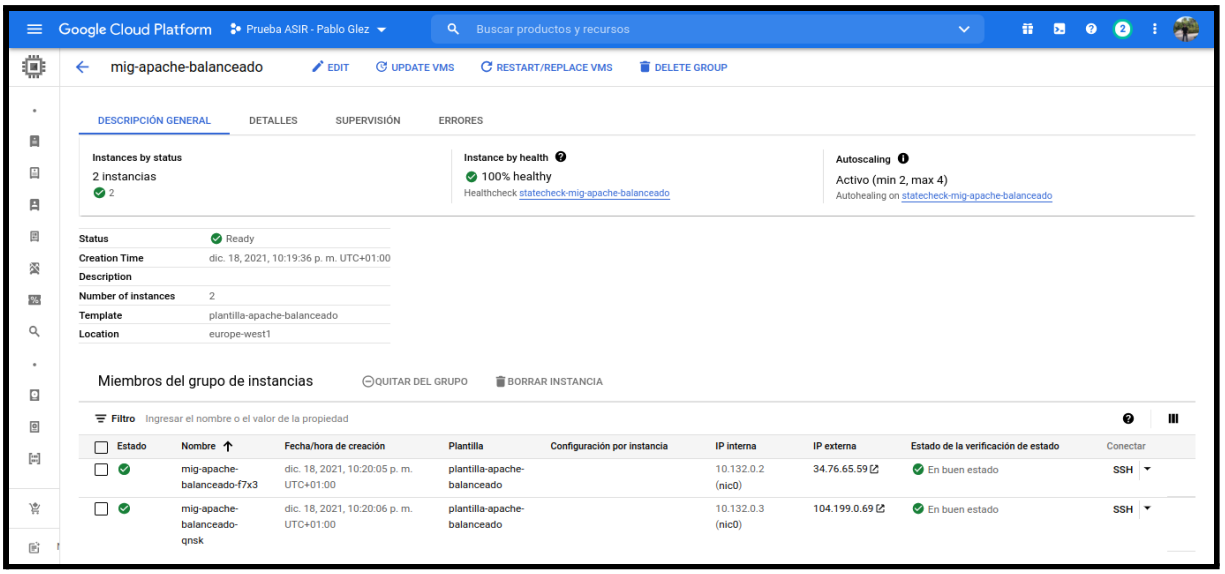
Iniciaremos Cloud Shell y ejecutaremos el siguiente comando:

```
gcloud compute --project=gcp-test-pablo-glez-asir2 firewall-rules create
allow-apache2 --direction=INGRESS --priority=1000 --network=default
--action=ALLOW --rules=tcp:80,tcp:443 --source-ranges=0.0.0.0/0
--target-tags=http-server,https-server --enable-logging
--logging-metadata=exclude-all
```

[10] https://cloud.google.com/compute/docs/instance-templates

Comprobaciones de las máquinas

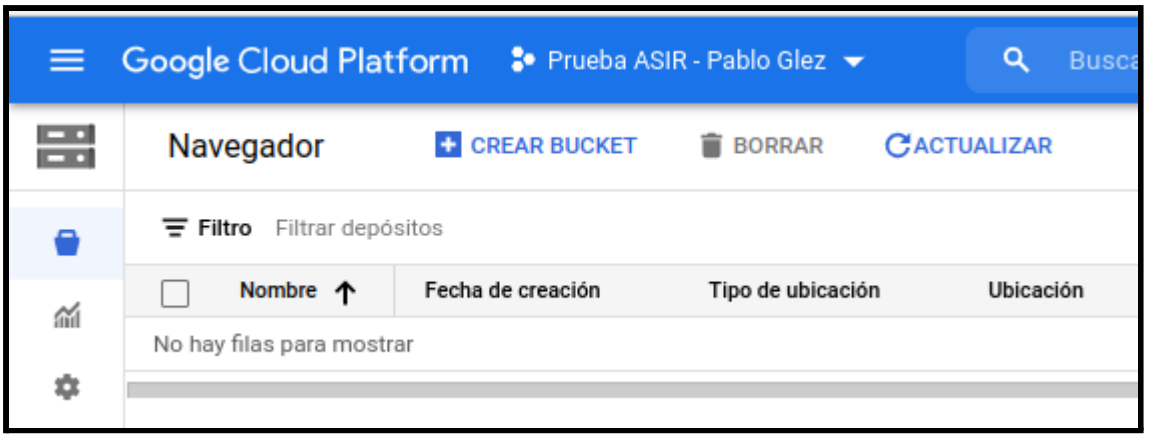
En la *overview* del grupo podemos ver que está funcionando:



Si accedemos a las IPs veremos como ha funcionado correctamente:

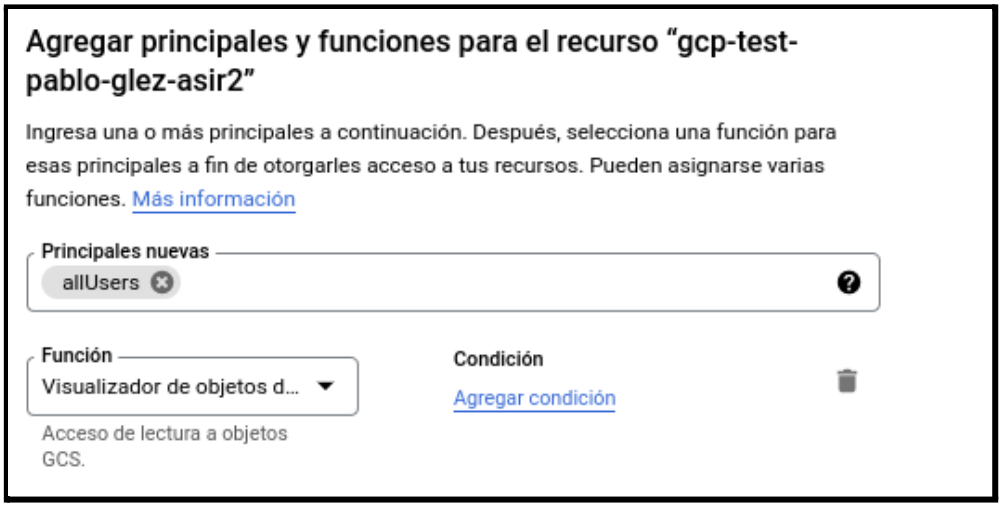
Creación de un *bucket* de GCS

Navegamos hasta el gestor de Google Cloud Storage y hacemos clic en *Crear Bucket*.



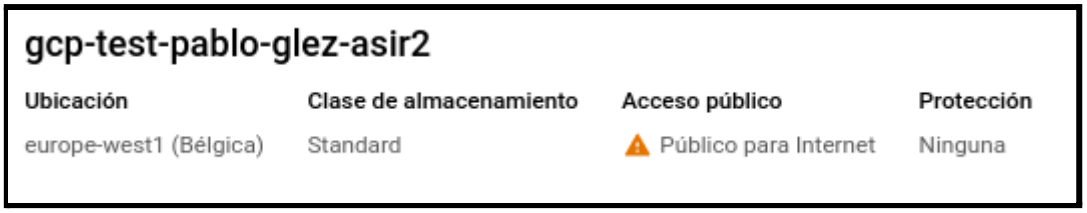
1. Como nombre podemos utilizar el nombre del proyecto
2. Para la ubicación, continuando con el caso práctico lo mejor sería multiregional, pero podemos elegir la opción que más nos convenga.
3. En la clase de almacenamiento[11] seleccionamos *Standard*. Esto sí es importante. En la nota al pie de página 11 hay más información sobre las opciones de "temperatura" para los datos.
4. Para hacerlo más sencillo, seleccionamos el control de acceso como uniforme y hacemos clic en continuar.
5. Podemos aplicar políticas de retención y ciclos de vida (guarda los datos por cumplimiento al menos 2 años, a los 100 días sin acceder a datos muévelos a una clase de almacenamiento más fría, etc).

Una vez creado el *bucket*, debemos añadir la personalidad *allUsers* para dotarlo de acceso público. Nos aparecerá un aviso para reconfirmar, por lo peligrosa que puede ser la acción[12]



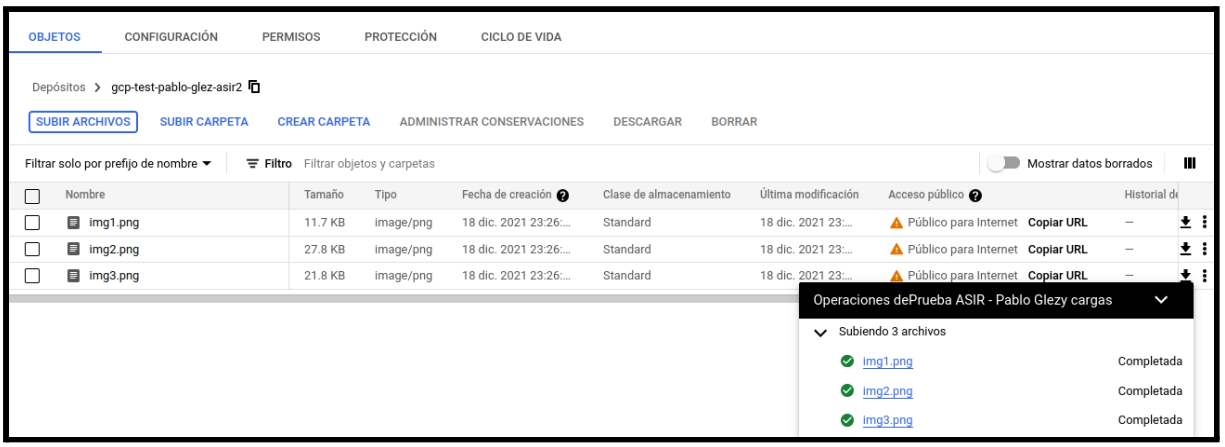
[11] https://cloud.google.com/storage/docs/storage-classes
[12] De ahí que haya políticas para prevenirlo a nivel de bucket.

En la parte superior de la página ya veremos la advertencia del acceso público:



En la sección *Objetos*, crearemos una carpeta llamada *static*.

Dentro de esta carpeta, hacemos clic en la opción *Subir objetos* y subimos algunos archivos/objetos de muestra. Los veremos en la interfaz web (no está creada la carpeta, úsese de ejemplo):



Si hacemos clic en *Copiar URL* y la copiamos en una ventana nueva, accederemos al objeto subido:

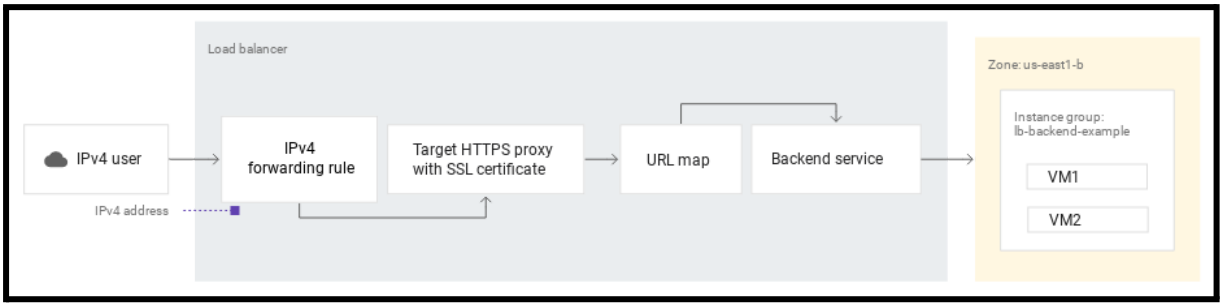
Las siguientes URLs son ambas válidas. También cabe la posibilidad de configurar un bucket con nombre de dominio y añadir un CNAME para que sea este dominio el endpoint de acceso.

https://storage.googleapis.com/gcp-test-pablo-glez-asir2/img1.png
https://gcp-test-pablo-glez-asir2.storage.googleapis.com/img1.png

Balanceador de carga[13] [14]

Definición y tipos

Un balanceador de cargas distribuye el tráfico del usuario en varias instancias de tus aplicaciones. Si se distribuye la carga, el balanceo de cargas reduce el riesgo de que la aplicación experimente problemas de rendimiento.[15]



Google Cloud ofrece las siguientes funciones de balanceo de cargas:

- Dirección IP Anycast única. Con Cloud Load Balancing, una sola dirección IP Anycast es el frontend de todas las instancias de backend en regiones de todo el mundo. Proporciona un balanceo de cargas entre regiones con conmutación automática por error multirregional, que traslada el tráfico a los backends de conmutación por error si los backends principales están en mal estado. Cloud Load Balancing reacciona de inmediato ante los cambios en los usuarios, el tráfico, la red, el estado del backend y otras condiciones relacionadas.

- Balanceo de cargas definido por software. Cloud Load Balancing es un servicio completamente distribuido, definido por software y administrado para todo tu tráfico. No es una solución basada en instancias ni en dispositivos, por lo que no estarás encerrado en una infraestructura de balanceo de cargas física ni tendrás que enfrentar los desafíos de alta disponibilidad (HA), escala y administración inherentes a los balanceadores de carga basados en instancias.

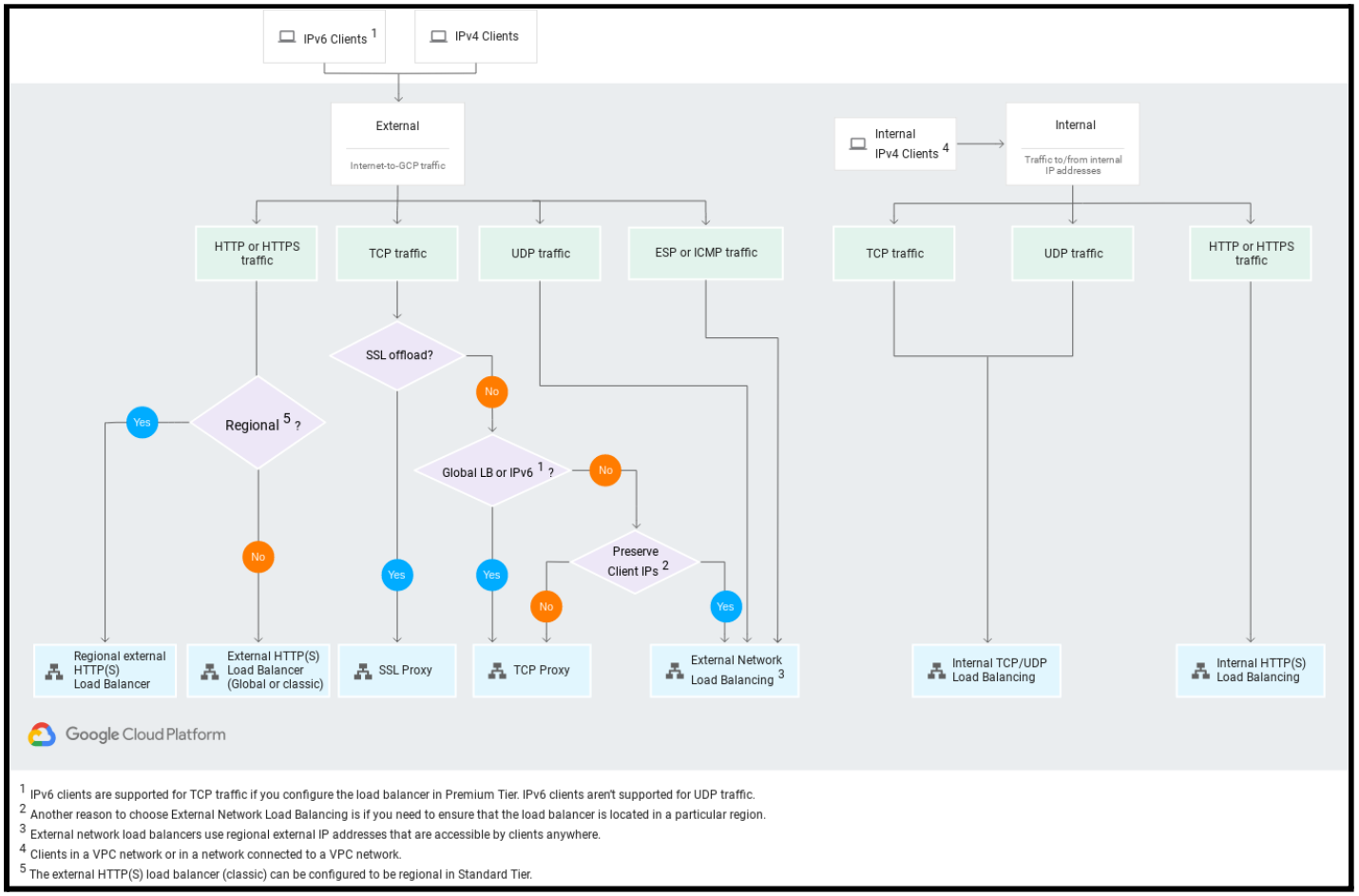
- Ajuste de escala automático sin interrupciones. Cloud Load Balancing puede escalarse a medida que aumentan la cantidad de usuarios y el tráfico; esto incluye la administración con facilidad de aumentos de tráfico inesperados, inmediatos y enormes, ya que los desvía a otras regiones del mundo que puedan aceptarlos. El ajuste de escala automático no requiere preparación previa: puedes pasar de cero a un funcionamiento pleno en cuestión de segundos.

- Balanceo de cargas de la capa 4 y la capa 7. Balanceo de cargas basado en la capa 4 para dirigir el tráfico según los datos de protocolos de red y capa de transporte, como TCP, UDP, ESP o ICMP. Usa balanceo de cargas basado en la capa 7 para

[13] Google Cloud networking: How Andromeda enables high-throughput VMs | Google Cloud Blog
[14] Maglev: A Fast and Reliable Software Network Load Balancer – Google Research
[15] https://cloud.google.com/load-balancing/docs/load-balancing-overview

agregar decisiones de enrutamiento de solicitudes en función de atributos, como el encabezado HTTP y el identificador uniforme de recursos

- Balanceo de cargas interno y externo. Puedes usar el balanceo de cargas externo cuando tus usuarios lleguen a tus aplicaciones desde Internet y el balanceo de cargas interno cuando tus clientes estén dentro de Google Cloud.
- Balanceo de cargas global y regional. Distribuye tus recursos con balanceo de cargas en una o varias regiones, para finalizar las conexiones cerca de tus usuarios y cumplir con tus requisitos de alta disponibilidad.
- Compatibilidad con funciones avanzadas. Cloud Load Balancing admite características como el balanceo de cargas global IPv6, WebSockets, encabezados de solicitudes definidos por el usuario y reenvío de protocolos para VIP privadas. También incluye las siguientes integraciones para el balanceo de cargas de HTTP(S) externo:
    - Integración con Cloud CDN para la publicación de contenido en caché
    - Integración con Google Cloud Armor para proteger tu infraestructura de ataques de denegación de servicio distribuido (DDoS) y otros ataques a aplicaciones dirigidas a aplicaciones.

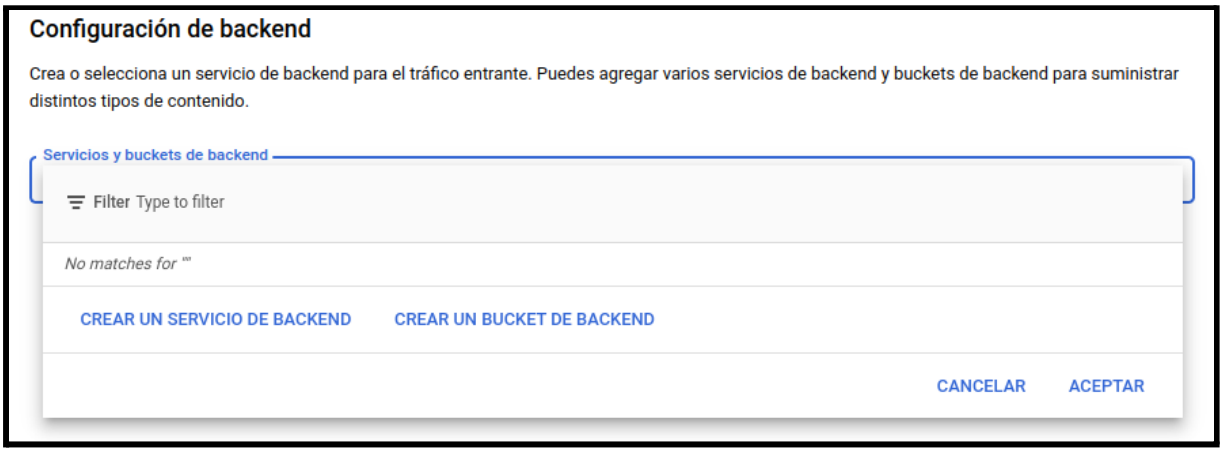


Creación del balanceador de carga

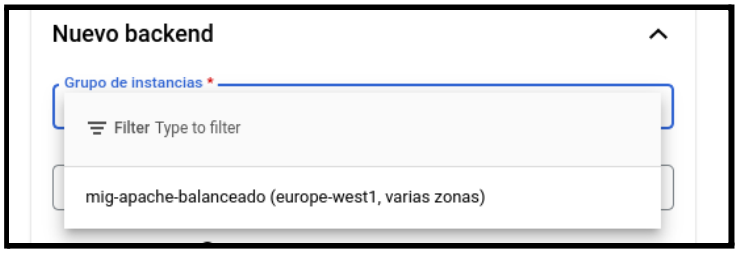
En el panel de administración web de Google Cloud Platform, navegamos hasta el apartado *Servicios de Red*, aquí, entre todas las opciones, seleccionamos *Balanceo de cargas*. En la siguiente pantalla que nos aparece, hacemos clic en Crear LB.

Veremos 3 opciones: balanceador de cargas HTTPS(S), balanceador de cargas TCP y balanceador de cargas UDP.

1. Seleccionamos la primera opción, HTTP(S)
2. Será un balanceador de carga entre internet y nuestras máquinas virtuales, por lo que seleccionamos esta opción en el apartado *Orientado a Internet o solo interno*.
3. Seleccionamos la Administración avanzada del tráfico[16], puesto que nos ofrecerá muchas más opciones para personalizar las rutas hacia nuestro *backend*. Hacemos clic en Continuar.
4. Introducimos un nombre para nuestro balanceador, puede ser *lb-web*, por ejemplo.
5. Debemos configurar 2 servicios de backend: el grupo de instancias de Compute Engine y el Bucket de Storage.



6. Comenzaremos haciendo clic en *Crear un servicio de Backend:*
   6.1. Definimos un nombre, *lb-backend-mig-apache2*, por ejemplo.
   6.2. Como tipo de backend, indicamos grupo de instancias.
   6.3. Dejamos los valores de destino como se encuentran (HTTP).
   6.4. Seleccionamos nuestro grupo de instancias en el desplegable.



[16] Traffic management overview for global external HTTP(S) load balancers

6.5. Introducimos el puerto 80, seleccionamos tasa como modo de balanceo. Para poder ver cómo las solicitudes se dirigen a instancias distintas, vamos a definir un máximo de 100 RPS (*request per second*, solicitudes por segundo), a un 80% de capacidad. El envío de tráfico se alternará entre las máquinas del grupo, pero nunca se dejará de enviar a este. El grupo está preparado para escalar si se supera el 60% de uso de CPU, por lo que es complicado que bloqueemos el servicio.

6.6. Para este backend no activaremos Cloud CDN. Queremos saturar las máquinas, de activarlo muchas solicitudes no llegarían a estas.

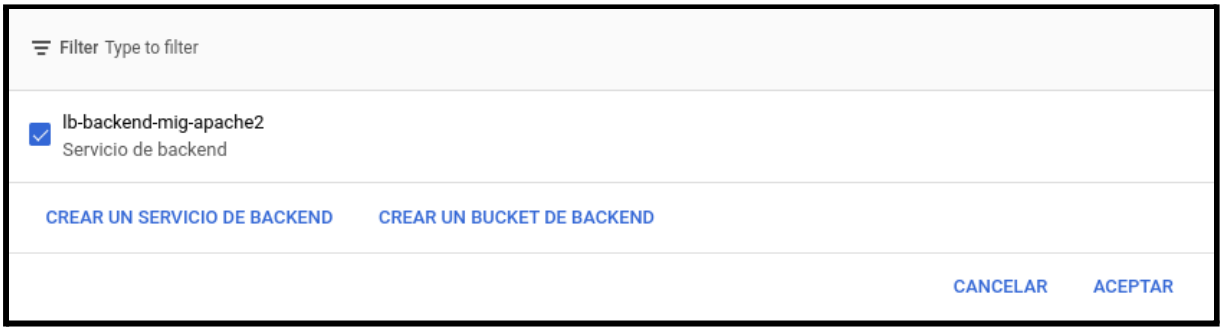
6.7. Asignamos la comprobación de estado que hemos creado en la creación del grupo de instancias.

6.8. Podemos habilitar el registro. Vamos a insertar "1" como tasa de muestreo, lo que significa que registraremos todas las peticiones, pero podemos reducirlo a nuestro gusto.

6.9. Para los encabezados de respuesta añadiremos los que se muestran en la tabla siguiente. Sí es cierto que sería más interesante añadirlos como encabezados de petición, pero así es más sencillo verlo.

Nombre del encabezado	Valor del encabezado
X-gcp-lb	lb-web
X-Ubicacion-Cliente	{client_region},{client_city}

6.10. Hacemos clic en Crear.



7. Ya tenemos disponible el grupo de instancias como backend. Ahora vamos a configurar el *bucket* como *backend*.

7.1. Al igual que hemos definido el nombre para el grupo de instancias, lo debemos definir para el backend del bucket. Indicaremos, por ejemplo, *lb-backend-asir2-bucket*.

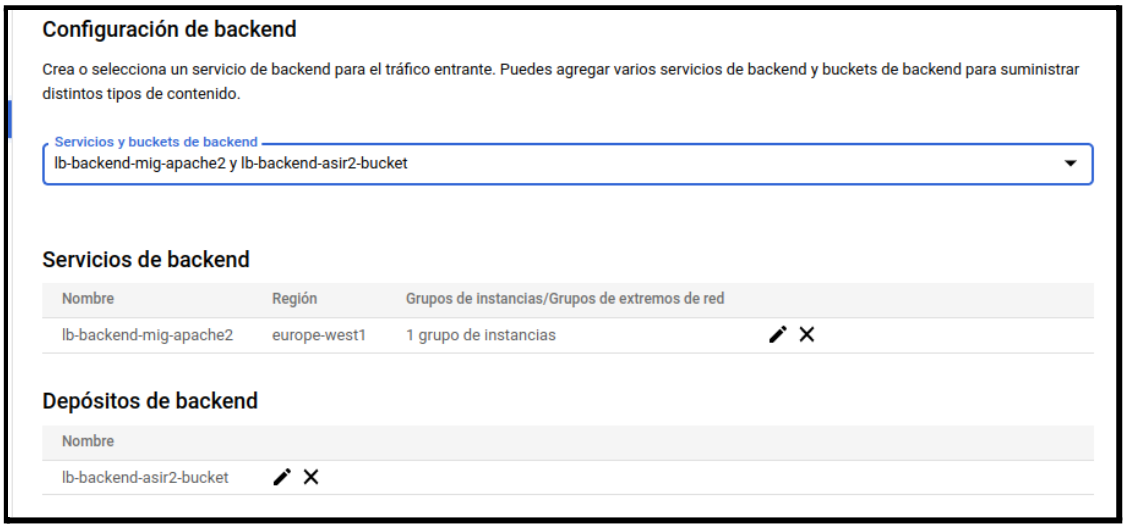
7.2. Haremos clic en *browse* para seleccionar qué contenedor/*bucket* de GCS queremos utilizar. Seleccionaremos el que hemos creado en el paso anterior.

7.3. Puesto que los recursos servidos desde este backend son recursos estáticos, conviene activar Cloud CDN para estos. Los costes de recuperación de Cloud CDN son menores que para Cloud Storage y aumentamos la velocidad.

7.4. Los valores por defecto de Cloud CDN son correctos, aunque conviene verlos detenidamente para entenderlos. Si los recursos pueden ser públicos, pero no divulgados (por ejemplo, las historias de Instagram) podemos activar una clave de firma. Debemos, como siempre, darle un nombre a este recurso. Puede ser, por ejemplo, *sign-key-lb-backend-asir2-bucket*. No lo activamos, pues para crear las URLs son necesarios pasos adicionales.

7.5. Hacemos clic en *Crear*.

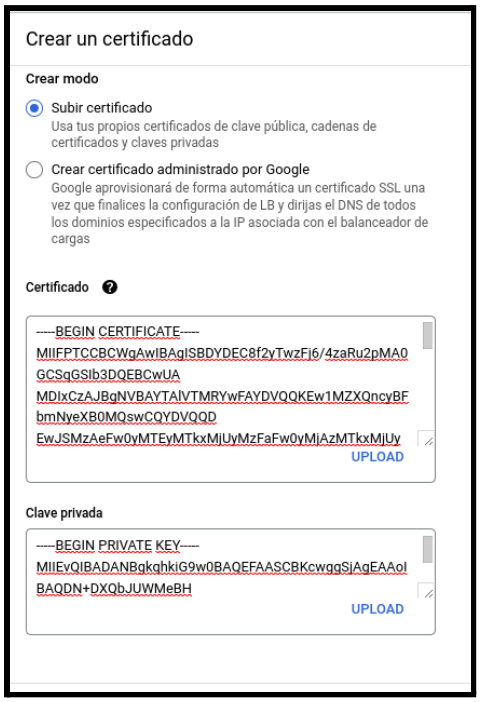
8. Ya veremos en la pantalla, ambos backends que acabamos de crear.



9. En el menú de la izquierda, seleccionamos la opción *Reglas de host y ruta*. Como ejemplo:

10. Con las reglas anteriores, indicamos que todo lo gestionarán las instancias de VM, excepto si se solicita un recurso en el host *gcp-test.gonzaleztroyano.es* y la ruta */static/**, que en este caso, se enviarán al contenedor.

11. Configuraremos a continuación el *Frontend* del balanceador.



11.1. Indicamos un nombre para este, puede ser *frontend-lb-web*.

11.2. En el protocolo, elegimos HTTPS. Seleccionamos nivel de red [17]Premium.

11.3. La configuración se puede mantener como está por defecto, o crear una reserva de IP.

11.4. En tanto al certificado, hacemos clic en crear un certificado. Tenemos dos opciones para generarlo: o bien subir el par de clave, o que Google genere un par de claves cuando apuntemos el registro DNS al balanceador de carga. Si decidimos subir el par de claves, debemos pegarlas en los cuadros de texto destinados a ello, como se puede ver el la imagen de la izquierda.

11.5. En cualquier caso, deberemos indicar el nombre para el recurso que estamos creando (el certificado). Un ejemplo puede ser *cert-lb-web*.

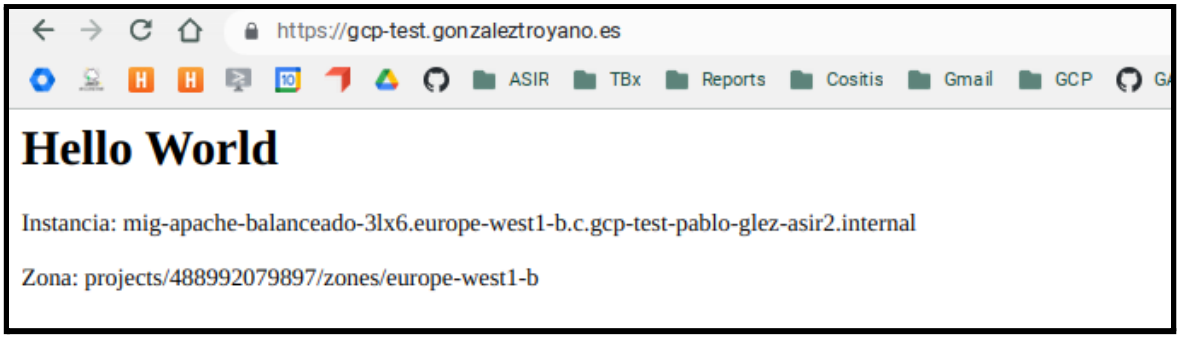
12. Hacemos clic en *Revisión y finalización*, comprobamos que esté correcto y le damos a *Crear*.

Comprobaciones del balanceador de carga

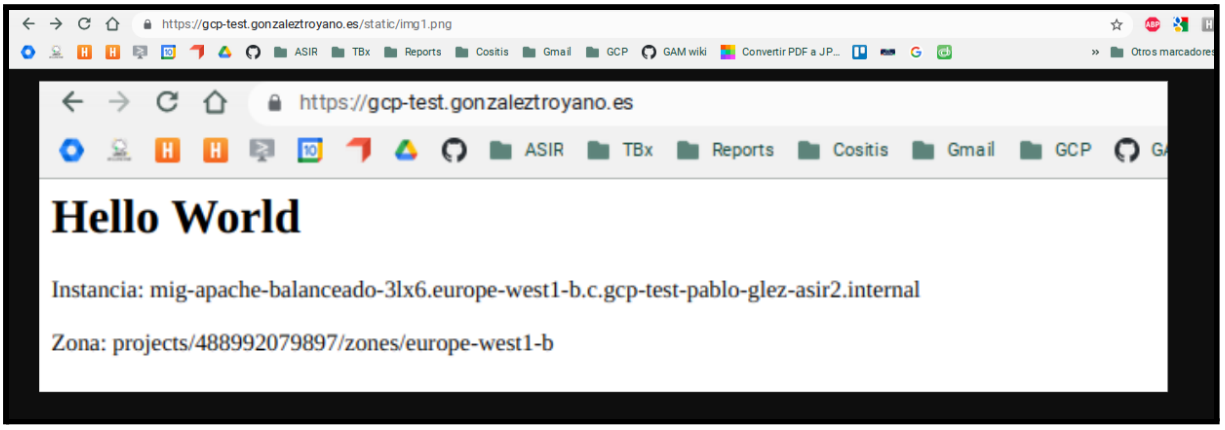
El endpoint de la red para nuestro balanceador de carga es gcp-test.gonzaleztroyano.es. Todas las peticiones que enviemos las tratará este en base a las reglas que hemos definido en la creación.

Si hacemos una petición, se enviará a una máquina virtual o a otra en base a su disponibilidad, así como al número de peticiones atendidas.

[17] https://cloud.google.com/network-tiers/



Para el contenido estático (se está sirviendo la imagen anterior):



Pruebas de estrés

Accedemos a Cloud Shell o creamos una instancia para realizar conexiones de forma simultánea.

```
apt update
apt install apache2-utils
```

```
ab -n 1000 https://gcp-test.gonzaleztroyano.es/
```

```
Server Software:        Apache/2.4.29
Server Hostname:        gcp-test.gonzaleztroyano.es
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-CHACHA20-POLY1305,2048,256
Server Temp Key:        X25519 253 bits
TLS Server Name:        gcp-test.gonzaleztroyano.es

Document Path:          /
Document Length:        202 bytes

Concurrency Level:      1
Time taken for tests:   6.514 seconds
Complete requests:      1000
```

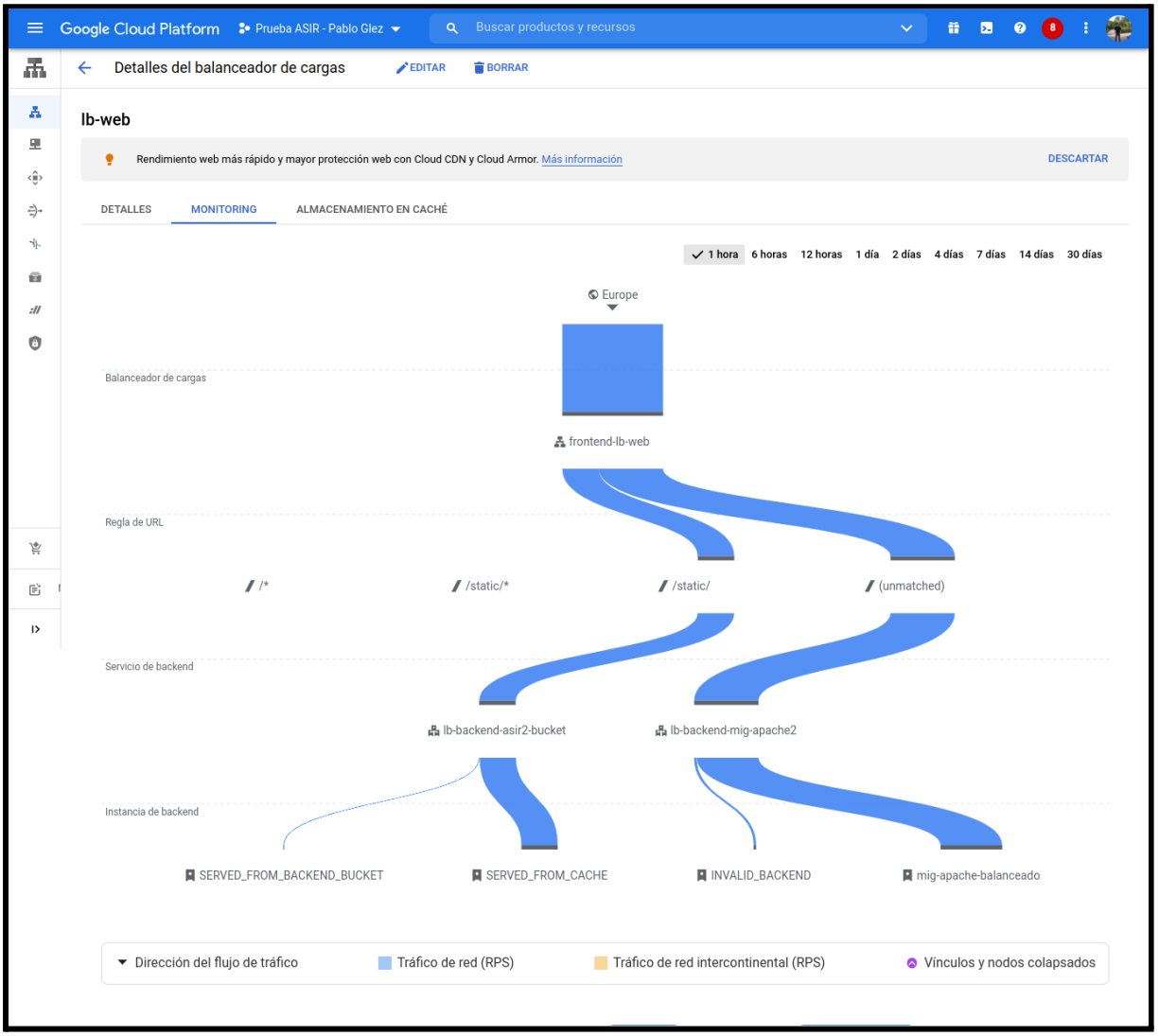
```
Failed requests:        0
Total transferred:      530000 bytes
HTML transferred:       202000 bytes
Requests per second:    153.53 [#/sec] (mean)
Time per request:       6.514 [ms] (mean)
Time per request:       6.514 [ms] (mean, across all concurrent requests)
Transfer rate:          79.46 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        3    4   6.7      3     213
Processing:     1    3   9.6      2     305
Waiting:        1    3   9.6      2     305
Total:          5    6  11.7      6     308

Percentage of the requests served within a certain time (ms)
  50%      6
  66%      6
  75%      6
  80%      7
  90%      7
  95%      8
  98%      9
  99%     10
 100%    308 (longest request)
```

ab -n 1000 https://gcp-test.gonzaleztroyano.es/static/img1.png

```
Server Software:        UploadServer
Server Hostname:        gcp-test.gonzaleztroyano.es
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-CHACHA20-POLY1305,2048,256
Server Temp Key:        X25519 253 bits
TLS Server Name:        gcp-test.gonzaleztroyano.es

Document Path:          /static/img1.png
Document Length:        29521 bytes

Concurrency Level:      1
Time taken for tests:   5.644 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      30178862 bytes
HTML transferred:       29521000 bytes
Requests per second:    177.19 [#/sec] (mean)
Time per request:       5.644 [ms] (mean)
Time per request:       5.644 [ms] (mean, across all concurrent requests)
Transfer rate:          5222.10 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        3    4   0.6      3       8
Processing:     1    2   2.7      2      53
```

```
Waiting:        1    1   2.7      1      53
Total:          4    6   2.8      5      57
WARNING: The median and mean for the initial connection time are not within a
normal deviation
        These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
  50%      5
  66%      6
  75%      6
  80%      6
  90%      6
  95%      7
  98%      8
  99%      9
 100%     57 (longest request)
```

Fijémonos en el contenido recuperado de caché:



ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/

```
Server Software:        Apache/2.4.29
Server Hostname:        gcp-test.gonzaleztroyano.es
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-CHACHA20-POLY1305,2048,256
Server Temp Key:        X25519 253 bits
TLS Server Name:        gcp-test.gonzaleztroyano.es

Document Path:          /
Document Length:        202 bytes

Concurrency Level:      1000
Time taken for tests:   30.432 seconds
Complete requests:      50000
Failed requests:        0
Total transferred:      26500000 bytes
HTML transferred:       10100000 bytes
Requests per second:    1643.01 [#/sec] (mean)
Time per request:       608.638 [ms] (mean)
Time per request:       0.609 [ms] (mean, across all concurrent
requests)
Transfer rate:          850.39 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        5  438  60.6    460     647
Processing:    63  161  59.9    139     408
Waiting:        3  101  53.7     88     325
Total:         68  600  23.3    596    1018

Percentage of the requests served within a certain time (ms)
  50%    596
  66%    601
  75%    604
  80%    607
  90%    622
  95%    628
  98%    640
  99%    649
 100%   1018 (longest request)
```

El sistema, después de aproximadamente un cuarto de millón de solicitudes, generó automáticamente una máquina nueva para afrontar la demanda:

Miembros del grupo de instancias QUITAR DEL GRUPO BORRAR INSTANCIA

Filtro Ingresar el nombre o el valor de la propiedad

Estado	Nombre	Fecha/hora de creación	Plantilla	Configuración por instancia	IP interna	IP externa	Estado de la verificación de estado	Conectar
✓	mig-apache-balanceado-2sml	dic. 19, 2021, 5:50:35 p. m. UTC+01:00	plantilla-apache-balanceado		10.132.0.7 (nic0)	35.195.134.94	En buen estado	SSH
✓	mig-apache-balanceado-3lx6	dic. 19, 2021, 1:30:35 p. m. UTC+01:00	plantilla-apache-balanceado		10.132.0.5 (nic0)	35.233.50.62	En buen estado	SSH
✓	mig-apache-balanceado-dkrk	dic. 19, 2021, 1:30:31 p. m. UTC+01:00	plantilla-apache-balanceado		10.132.0.4 (nic0)	34.77.198.114	En buen estado	SSH
✓	mig-apache-balanceado-pltd	dic. 19, 2021, 5:42:03 p. m. UTC+01:00	plantilla-apache-balanceado		10.132.0.6 (nic0)	35.187.23.122	En buen estado	SSH

```
pablo_gonzalez@cloudshell:~$ nohup: ignoring input and appending output to 'nohup.out'

pablo_gonzalez@cloudshell:~$ jobs
[2]   Running                 nohup ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/ &
[3]   Running                 nohup ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/ &
[4]-  Running                 nohup ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/ &
[5]+  Running                 nohup ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/ &
pablo_gonzalez@cloudshell:~$
```

cloudshell

```
  1 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]   Tasks: 42, 96 thr; 4 running
  2 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]   Load average: 7.23 2.93 1.38
  3 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]   Uptime: 00:36:21
  4 [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]
  Mem[||||||||||||||                                                      1.69G/15.6G]
  Swp[                                                                       0K/0K]

    PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
    995 pablo_gon  10 -10 73436 65684  5364 R 54.0  0.4  0:23.32 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
    990 pablo_gon  10 -10 73348 65980  5448 R 48.8  0.4  0:27.35 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
   1007 pablo_gon  10 -10 73468 65200  5380 R 34.9  0.4  0:12.95 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
   1027 pablo_gon  10 -10 73304 64944  5256 R 34.3  0.4  0:07.81 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
   1021 pablo_gon  10 -10 73356 65092  5396 R 32.9  0.4  0:09.03 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
   1024 pablo_gon  10 -10 73424 65024  5244 R 32.3  0.4  0:07.74 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
   1013 pablo_gon  10 -10 73412 65072  5328 R 29.7  0.4  0:09.65 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
   1000 pablo_gon  10 -10 73456 65448  5408 R 29.0  0.4  0:19.70 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
   1010 pablo_gon  10 -10 73240 65036  5448 R 28.3  0.4  0:12.32 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
   1018 pablo_gon  10 -10 73324 65128  5448 R 22.4  0.4  0:07.70 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
   1030 pablo_gon  10 -10 73196 64624  5356 R 21.7  0.4  0:04.59 ab -n 50000 -c 1000 https://gcp-test.gonzaleztroyano.es/
```

mig-apache-balanceado EDIT UPDATE VMS RESTART/REPLACE VMS DELETE GROUP

DESCRIPCIÓN GENERAL DETALLES SUPERVISIÓN ERRORES

Gráfico
Tamaño del grupo
Current autoscaler state: Activo

Crear política d Haz clic para seleccionar un intervalo de tiempo diferente

Current max
Current min

5 4 3 2 1 0

17:05 17:10 17:15 17:20 17:25 17:30 17:35 17:40 17:45 17:50 17:55 18:00

Instancias: 3

Utilización del escalador automático (CPU) Explora en Monitoring

150% 75% 0

17:05 17:10 17:15 17:20 17:25 17:30 17:35 17:40 17:45 17:50 17:55 18:00

Capacidad: 60.00% Utilización: 3.34%

Nextcloud Docker compose

Creación de máquina virtual, con el siguiente startup-script:

```
#!/bin/bash

apt-get update

apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -

add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"

apt-get update

apt-get install -y docker-ce

curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-compo
se-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose
```

Este script instalará docker y docker compose (se supone):

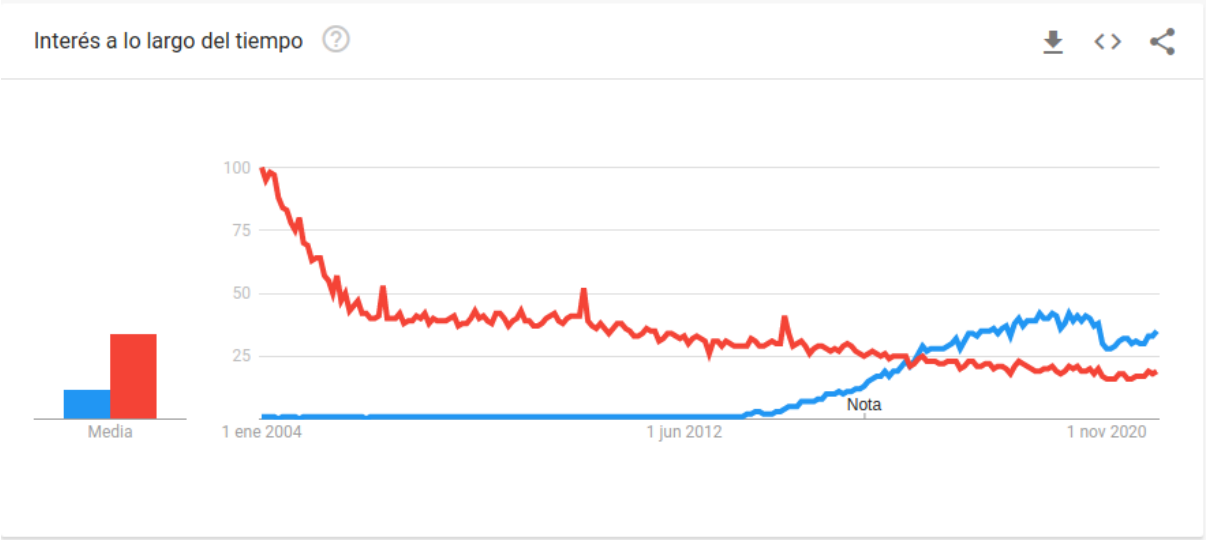
Utilizaremos el [siguiente]() docker-compose para configurar

```
services:
  nc:
    image: nextcloud:apache
    restart: always
    ports:
      - 80:80
    volumes:
      - nc_data:/var/www/html
    networks:
```

```
      - redisnet
      - dbnet
    environment:
      - REDIS_HOST=redis
      - MYSQL_HOST=db
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
      - MYSQL_PASSWORD=nextcloud
  redis:
    image: redis:alpine
    restart: always
    networks:
      - redisnet
    expose:
      - 6379
  db:
    image: mariadb:10.5
    command: --transaction-isolation=READ-COMMITTED --binlog-format=ROW
    restart: always
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - dbnet
    environment:
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
      - MYSQL_ROOT_PASSWORD=nextcloud
      - MYSQL_PASSWORD=nextcloud
    expose:
      - 3306
volumes:
  db_data:
  nc_data:
networks:
  dbnet:
  redisnet:
```

```
docker-compose up -d
```

Kuber... ¿qué?



[18]

- https://cloud.google.com/blog/topics/developers-practitioners/container-story-google-kubernetes-engine
- https://towardsdatascience.com/key-kubernetes-concepts-62939f4bc08e
- https://kubernetes.io/es/
- https://kubernetes.io/es/case-studies/
- https://www.cncf.io/
- Implementa WordPress en GKE con Persistent Disk y Cloud SQL
- https://cloud.google.com/blog/topics/developers-practitioners/how-pok%25C3%25A9mon-go-scales-millions-requests
- https://cloud.google.com/kubernetes-engine?hl=es

¿Contenedores? ¿De reciclaje dices?

La evolución en informática ha ido avanzando hacia la virtualización, huyendo de las máquinas "físicas". A mediados de la década de los 2.000 se inició la virtualización. Con la virtualización se instalaban grandes máquinas que emulaban ser muchas más.

Este sistema, la virtualización, fue una gran mejora. Permitió mucha más agilidad que la preexistente. Un "kernel error" no era sinónimo de parar un entorno de producción o tener la empresa parada, pues en cuestión de minutos/horas es posible volver a lanzar otra máquina.

En el ejemplo anterior, suponiendo que un SO necesita 5 Gb de almacenamiento, 2 Gb de RAM y bastante potencia de proceso, escalado a 16 máquinas, estamos utilizando de forma

[18] Comparación entre máquinas virtuales y Docker

poco eficiente los recursos de computación. Esto, a escala de una gran empresa supone una empresa y unos costes muchos más elevados.

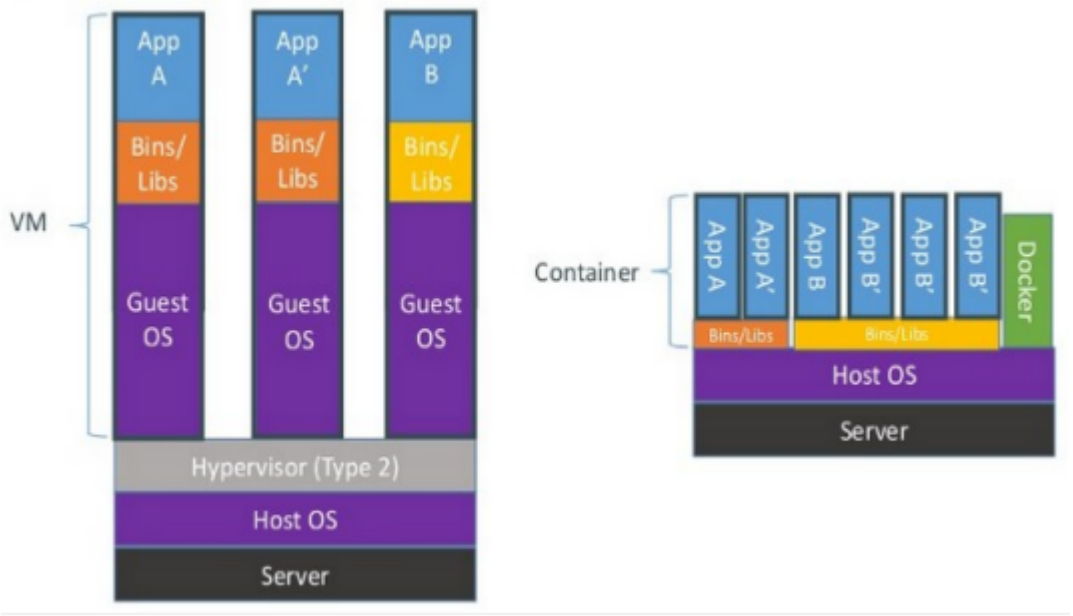
Si existían 8 subsistemas para un desarrollo, necesitaba virtualizar 8 sistemas operativos completos. Si además era necesaria redundancia geográfica, con solo 2 zonas, se virtualizaban 16 máquinas completas.[19]

Los informáticos e informáticas inteligentes de inicios de los 2010, se dieron cuenta de que no era nada eficiente virtualizar todo el sistema operativo.

La parte importante de la virtualización no era el Sistema Operativo Virtualizado (que era necesario mantener y actualizar), sino los servicios que ofrecían.

Por ello, se desarrolló el concepto de los contenedores, que permiten tener una abstracción similar a las máquinas virtuales, pero virtualizando solo el servicio en cuestión.[20]

Los contenedores permiten virtualizar únicamente lo estrictamente necesario para el funcionamiento del servicio y nada más. Sin depender del SO anfitrión; aunque sí utilizando sus librerías y kernels.



El proceso de utilizar contenedores "a pelo" es simple cuando el desarrollo y el mantenimiento se lleva a cabo por pocas personas. y la demanda no es variable.

Kubernetes intenta dar respuesta a algunas preguntas, como:

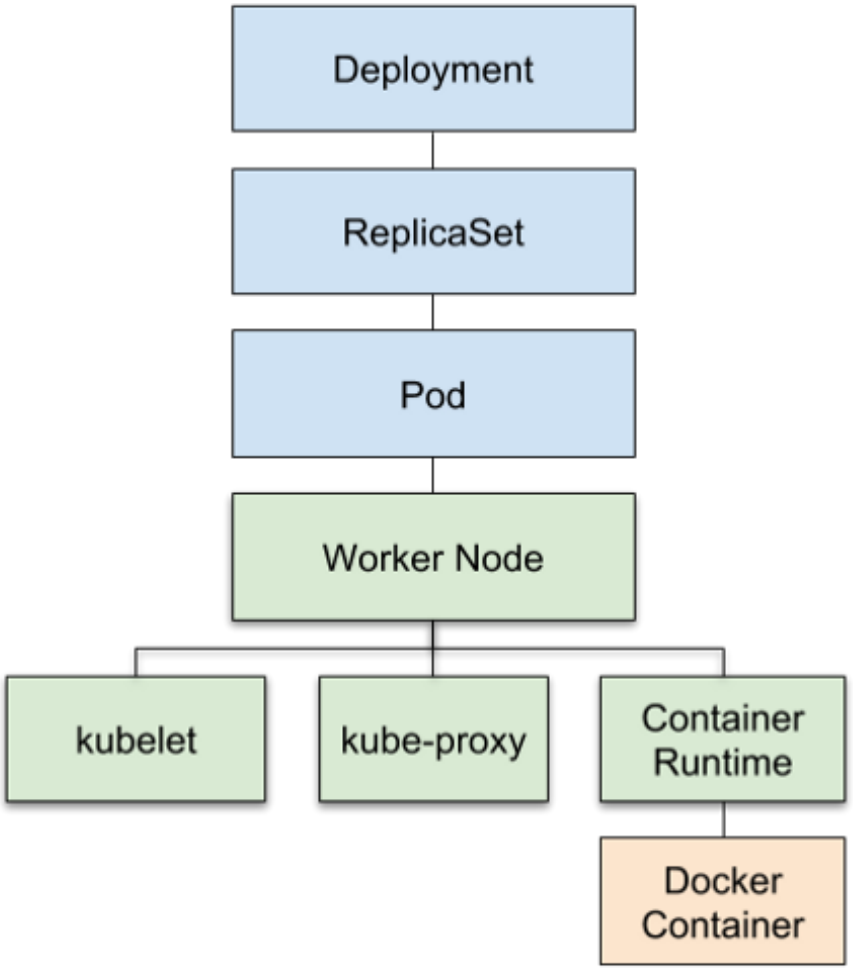
[19] Es una buena práctica, sobre todo hoy en día con los contenedores, separar servicios para evitar fallos en cascada y facilitar la recuperación ante errores.
[20] Los contenedores en comparación con las máquinas virtuales

- ¿Funcionará en todas las máquinas?
- ¿Qué ocurre si no funciona?
- ¿Cómo se absorben los picos de tráfico?

Así funciona kubernetes

Perdón por el lío al lector/oyente. Es una estructura compleja. No veremos todos los detalles ni los procesos de los masters o workers.

Componentes / Capas de abstracción



[21]

Deployment

Los despliegues sirven para poder actualizar aplicaciones *stateless*, sin estado, de forma transparente al usuario. En los despliegues también definimos políticas de reinicio y accicón en caso de fallo de los *Pods*[22].

[21] © Jeff Hale en towardsdatascience.com. Consultado en noviembre de 2021
[22] ¿Mayúscula o minúscula? Ni en Kubernetes lo saben.

ReplicaSet

Los *ReplicaSets* son una especie de sucesores de los "*Replication Controllers*". Con este otro nombre es mucho más sencillo entender su función.

Los *ReplicaSets* son los encargados de que los *Deployments* tengan el número deseado de *Pods*, de acuerdo a las directrices de replicación que asignamos en el despliegue.

>> Check: ¿Cuántos pods tengo disponibles y funcionales?
>> Answer: ¿5? ¡Necesito 8! Master, dile al worker que me cree Pods a cascoporro.
>> Check: ¿Cuántos Pods tengo disponibles y funcionales?
>> Answer: 8. Volveré a comprobar en un tiempo.

Pod

Los Pods son los elementos básicos de Kubernetes. Un Pod contiene uno o más contenedores. Normalmente, solamente contendrá 1, pero depende de la carga de trabajo esperada y la configuración.

Con los Pods gestionamos: Volúmenes, secretos y su propia configuración.

Son efímeros. No tienen persistencia (sin los volúmenes) y están pensados para reiniciarse en caso de fallo. Para un pico de tráfico, se aumenta el número de estos. Para un descenso, se eliminarán.

Los Pods son replicados cuando el ReplicaSet ordena el escalado horizontal del sistema.

Los Pods están alojados en los *Worker nodes*.

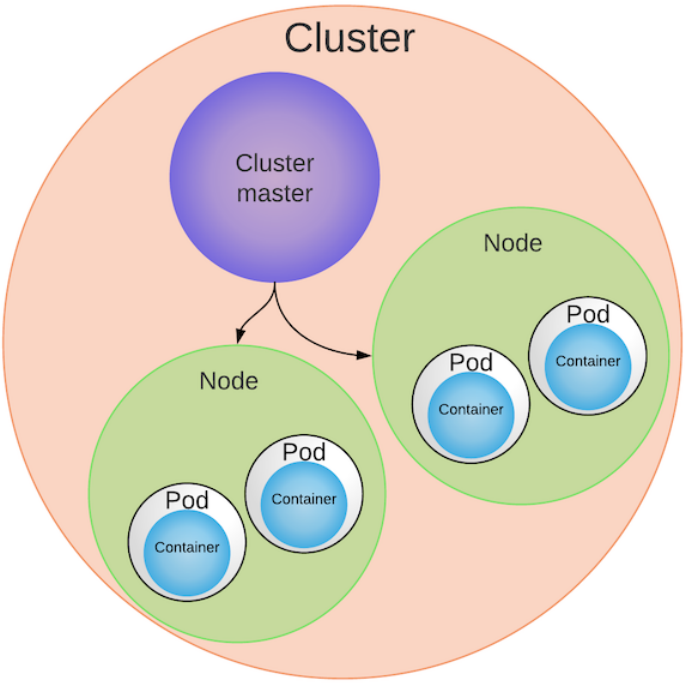
Cluster Level

Cluster

En un cluster distinguimos 2 roles principales: los nodos "master" y los nodos "worker".

En un mismo cluster podemos tener varios nodos trabajadores, que serán coordinados por un master node.

[23]



[23] © Jeff Hale en towardsdatascience.com. Consultado en noviembre de 2021

Worker Node

Realmente nadie lo llama “Worker Node”, sino simplemente “Node”. Es la abstracción a la máquina (virtual o física).

Uno o varios Pods funcionan en un Worker Node. El Master Node controla la ejecución de los workers.

Cluster Master

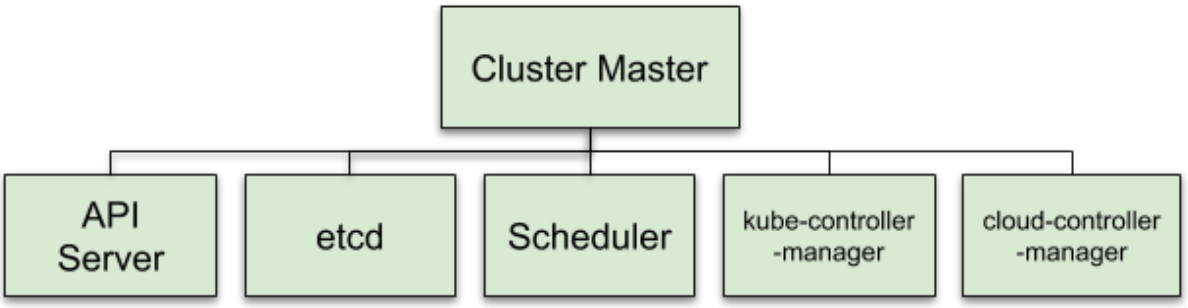
Controla y dirige el trabajo de los *workers*. Realiza decisiones sobre programación, responde a eventos, implementa cambios y monitoriza el Cluster.

También llamado Master Node, Kubernetes Master, Cluster Control Plane, Control Plane, o solamente Master.

Node Processes

Master Components

[24]



API Server — Exposes the K8s API. It’s the frontend for Kubernetes control. (aka. kube-apiserver) Think hub.

etcd — Distributed key-value store for Cluster state data. Think Cluster info.

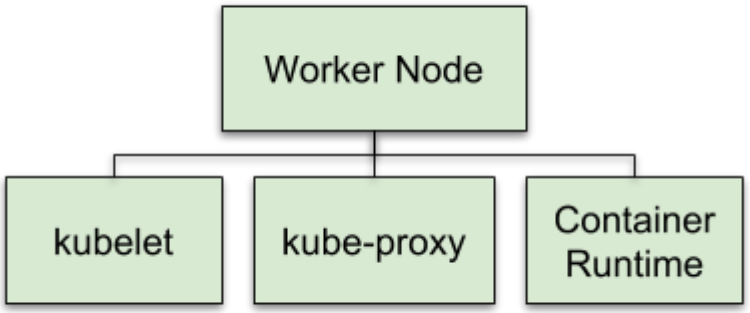
Scheduler — Selects the Nodes for new Pods. Good guide here. (aka kube-scheduler) Think matcher.

kube-controller-manager —Process that runs controllers to handle Cluster background tasks. Think Cluster controller.

cloud-controller-manager — Runs controllers that interact with cloud providers. Think cloud interface.

[24] © Jeff Hale en towardsdatascience.com. Consultado en noviembre de 2021

Worker Node Components



[25]

kubelet — Responsible for everything on the Worker Node. It communicates with the Master's API server. Think brain for Worker Node.

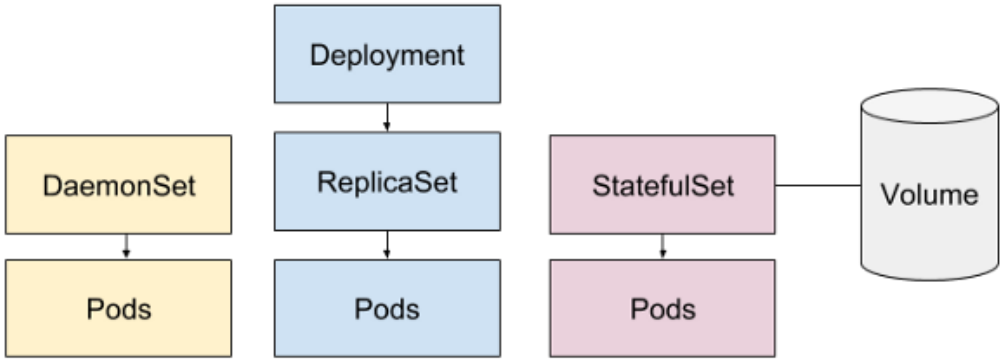
kube-proxy — Routes connections to the correct Pods. Also performs load balancing across Pods for a Service. Think traffic cop.

Container Runtime — Downloads images and runs containers. For example, Docker is a Container Runtime. Think Docker.

Docker Container Level

Cuando creamos el *Deployment,* indicamos a Kubernetes qué imagen de contenedor deben correr nuestros Pods.

Kubernetes NO crea contenedores. Crea Pods que tiene contenedores.



Como hemos visto antes, los/las *ReplicaSets* crea y gestiona los Pods.

[25] © Jeff Hale en towardsdatascience.com. Consultado en noviembre de 2021

Cuando un nodo se retira, el/la *ReplicaSet* crea un nodo nuevo.

¿Qué pasa con los datos? Si tengo que mantenerlos uso un *StateFullSet*

SatefulSet

Al igual que *ReplicaSet*, StateFullSet gestiona el despliegue y el escalado de un grupo de Pods basándose en las declaraciones.

En un despliegue de este tipo, los contenedores no son intercambiables[26]. Cada Pod tiene un identificador único y persistente que el controlador (StateFullSet) mantiene en cada reinicio. El estado de la información del pod es guardado en el volumen asociado.

DaemonSet

Están pensados para procesos contínuos. 1 pod = 1 nodo. Cada nodo nuevo es añadido a un Pod creado por el DaemonSet.

Están pensados para tareas en segundo plano

Servicios

Los servicios son utilizados para disponer de un único punto de acceso para un grupo de Pods. Provee una dirección IP estática (los Pods NO la tienen, *son creados para morir*).

Los servicios no solo sirven para los usuarios, sino que un Pod puede utilizar un servicio ofrecido por otros Pods.

Volum-cosas

Volumes

Un volumen es un directorio que mantiene datos. Es un componente del Pod, declarado durante su especificación.

Es accesible para todos los contenedores del Pod, pero se debe montar individualmente. “Sobrevive” a la *muerte* de los contenedores, pero NO del Pod.

Son más funcionales que los volúmenes de Docker. Un volumen puede proveer acceso a disco, memoria y almacenamiento remoto y un mismo pod puede los puede usar combinados.

Para los servicios administrados, existen posibilidades como awsElasticBlockStore y gcePersistentDisk

PersistentVolumes and PersistentVolumeClaims

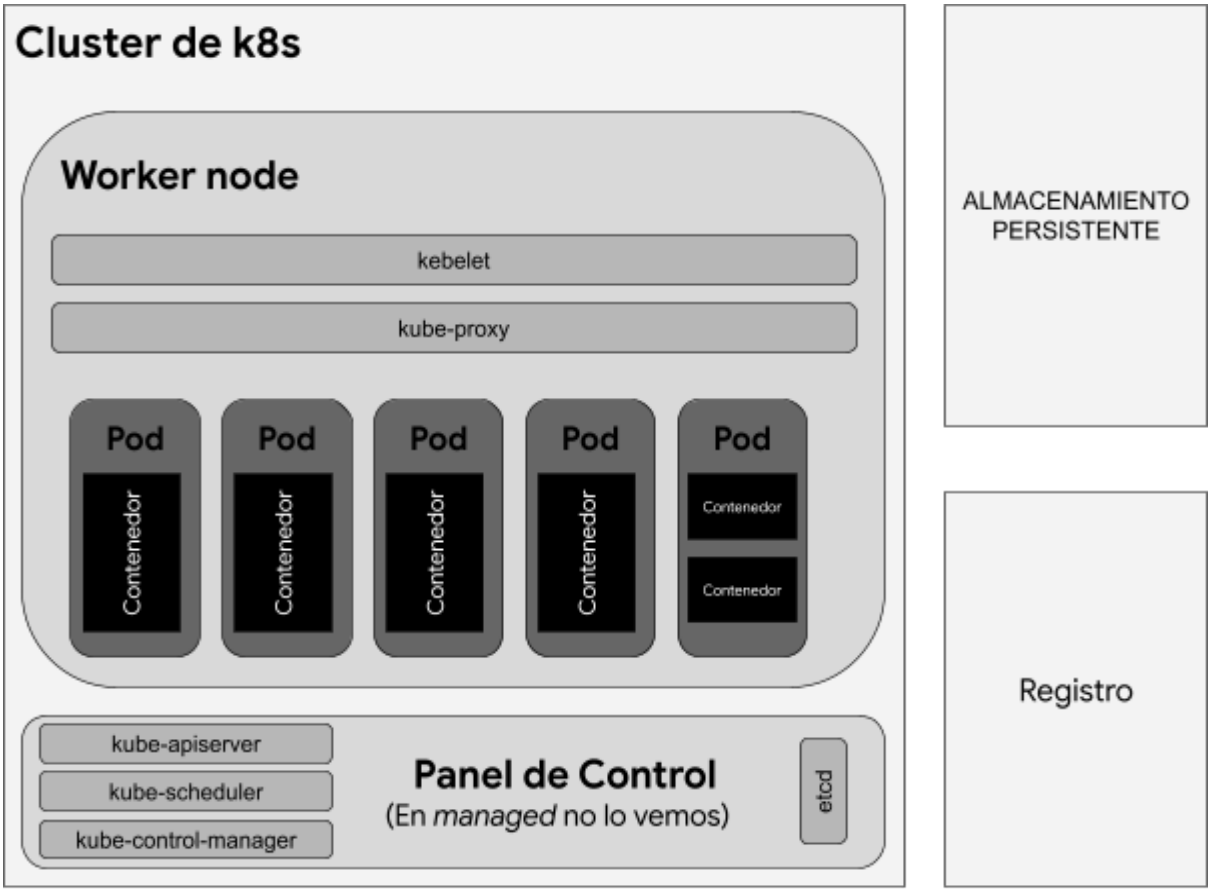
Cosas extrañas que solo la NASA utiliza.

[26] Sí lo son en, por ejemplo, un frontend estático.

Resumen de Kubernetes

[Buen resumen en esta entrada]

[En formato cómic]



Despliegues con Kubernetes

hello-app en GKE

1. Activamos Cloud Shell
    a. Entender qué es Cloud Shell
    b. Entender qué son los comandos gcloud, ver credenciales

```
gcloud auth list
```

    c. Ver proyectos:

```
gcloud projects list
```

    d. Seleccionar proyecto:

```
gcloud config set project gcp-test-pablo-glez-asir2
```

2. Seleccionamos zona por defecto para *compute*

```
gcloud compute zones list | grep NAME | grep eu
gcloud config set compute/zone europe-west1-b
```

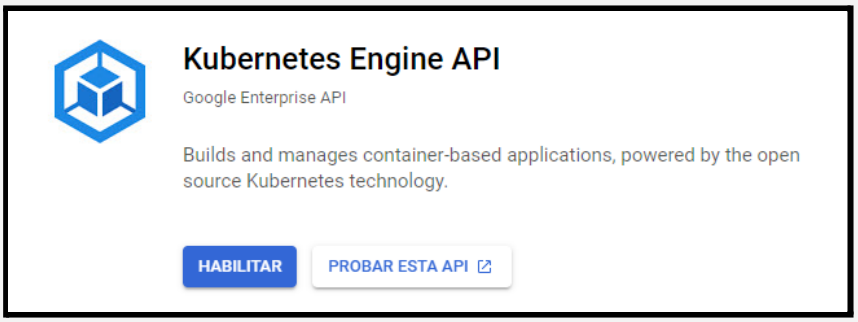
3. Desplegar el cluster

```
gcloud container clusters create asir2-test-cluster --region europe-west1-b
```

Si vemos un mensaje de error parecido al siguiente:

```
ResponseError: code=400, message=Failed precondition when calling the ServiceConsumerManager: tenantmanager::185014: Consumer 488992079897 should enable service:container.googleapis.com before generating a service account
```

Debemos navegar hasta https://console.cloud.google.com/kubernetes/list y activar las APIs de Google Kubernetes Engine

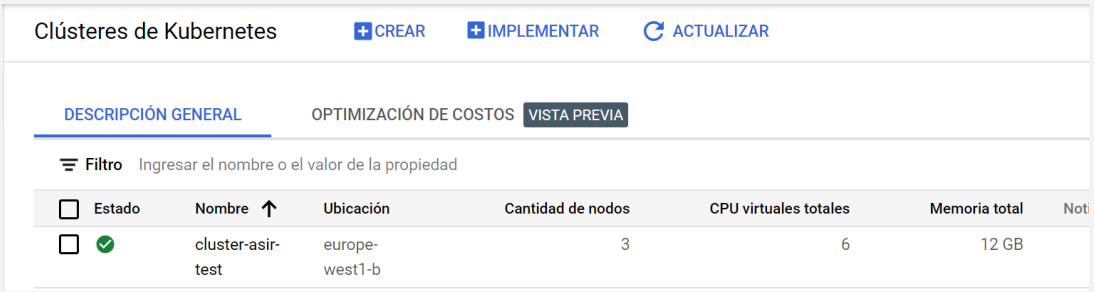


Una vez activadas, podemos volver a intentar crear el clúster. Tardará unos minutos.
Nos mostrará los detalles en la consola de CloudShell:

```
NAME: cluster-asir-test
LOCATION: europe-west1-b
MASTER_VERSION: 1.21.5-gke.1302
MASTER_IP: 35.187.97.204
MACHINE_TYPE: e2-medium
NODE_VERSION: 1.21.5-gke.1302
NUM_NODES: 3
STATUS: RUNNING
```

4. Ver detalles del cluster en GUI.

   Tenemos el cluster desplegado.



   También los *workers nodes* como instancias en Compute Engine:

5. Para "hacer cosas" con el cluster necesitamos credenciales:

```
gcloud container clusters get-credentials $NOMBRE
```

6. Implementar aplicación en cluster

```
kubectl create deployment hello-app
--image=gcr.io/google-samples/hello-app:1.0
```

   ¿Qué hace esto? Crear el *Deployment*.

7. Escalemos el deployment:

```
kubectl scale deployment hello-app --replicas=3
```

8. También podemos definir una política de auto escalada:

```
kubectl autoscale deployment hello-app --cpu-percent=80 --min=1 --max=5
```

9. Con `kubectl get pods` podemos listar los pods (""contenedores"") funcianando.

10. Exponemos el servicio:

```
kubectl expose deployment hello-app --name=hello-app-service
--type=LoadBalancer --port 80 --target-port 8080
```

¿Qué hace esto? Crear el *Service*.

NOTA: Usamos GUI para crear un deployment con X replicas y ab test. Comparamos medians. Observamos distribución en metric explorer.

11. Ver detalles del servicio:

```
kubectl get service
```

```
NAME                TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
hello-app-service   LoadBalancer   10.72.13.179   <pending>     80:31818/TCP   32s
kubernetes          ClusterIP      10.72.0.1      <none>        443/TCP        6m22s
```

12.
13. Eliminar los recursos (💸)

```
gcloud container clusters delete $NOMBRE
```

Comparación de rendimiento. 1 réplica vs 3 réplicas:

```
Server Hostname:        34.88.108.7
Server Port:            80

Document Path:          /
Document Length:        52 bytes

Concurrency Level:      1000
Time taken for tests:   2.531 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      1690000 bytes
HTML transferred:       520000 bytes
Requests per second:    3950.74 [#/sec] (mean)
Time per request:       253.117 [ms] (mean)
Time per request:       0.253 [ms] (mean, across all
concurrent requests)
Transfer rate:          652.03 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       37   37   0.4     37      43
Processing:    37  198 158.5    144    1311
Waiting:       37  195 159.8    136    1311
Total:         75  235 158.5    182    1349
```

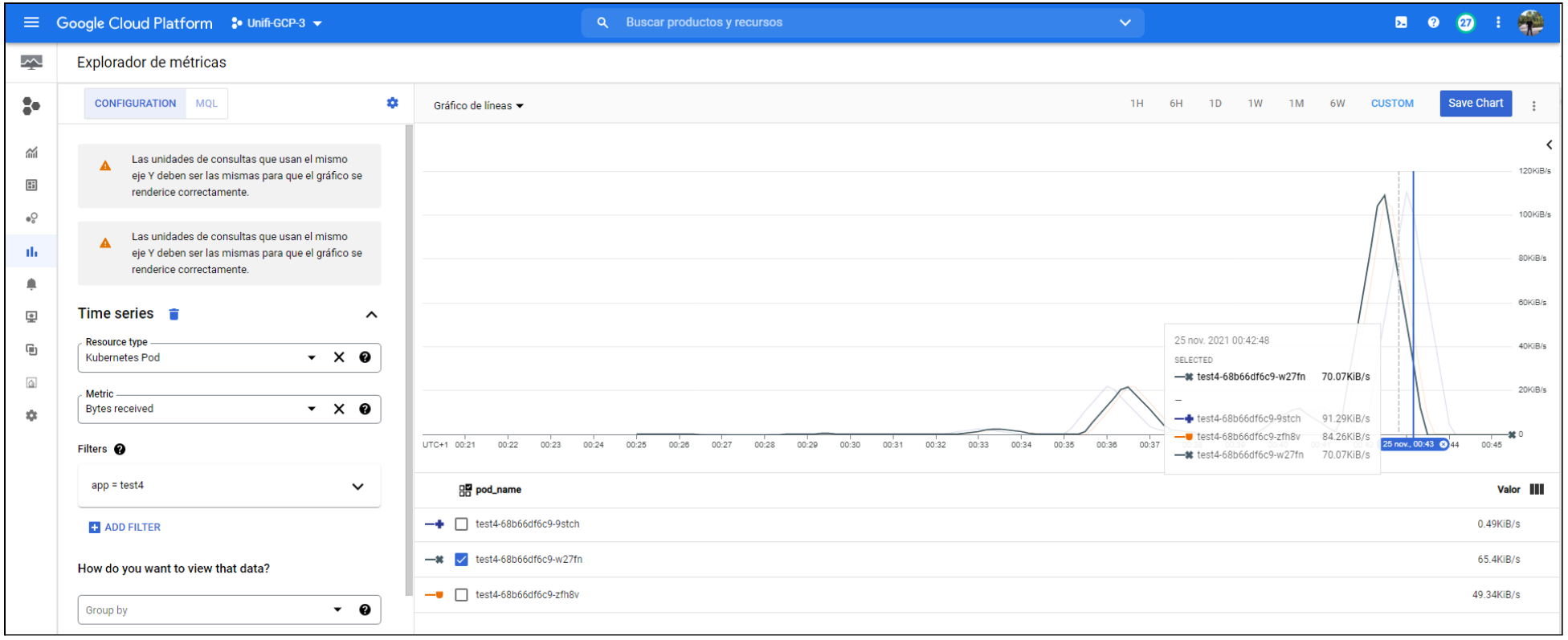
```
Server Hostname:        35.228.175.143
Server Port:            80

Document Path:          /
Document Length:        62 bytes

Concurrency Level:      1000
Time taken for tests:   1.226 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      1790000 bytes
HTML transferred:       620000 bytes
Requests per second:    8157.10 [#/sec] (mean)
Time per request:       122.593 [ms] (mean)
Time per request:       0.123 [ms] (mean, across all
concurrent requests)
Transfer rate:          1425.90 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       37   41   4.2     38      54
Processing:    37   67  35.1     56     410
Waiting:       37   65  34.5     54     405
Total:         75  108  34.7     98     448
```

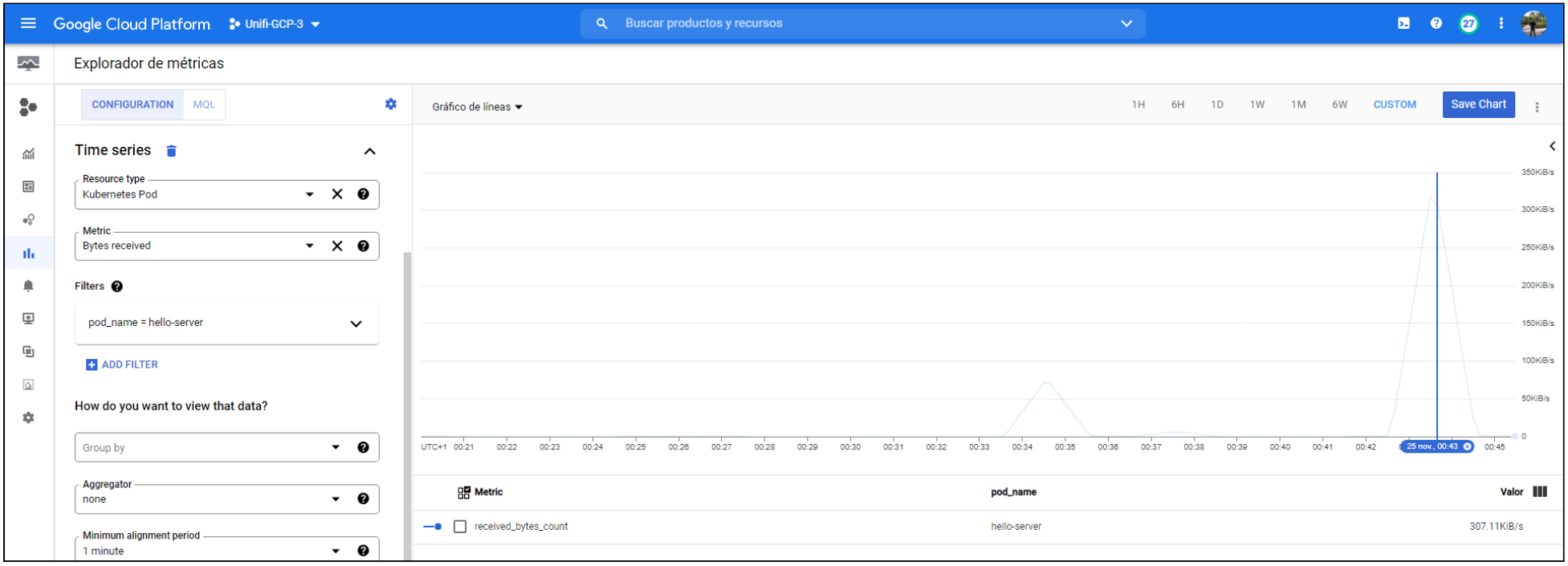
```
Server Hostname:        34.88.108.7
Server Port:            80

Document Path:          /
Document Length:        52 bytes

Concurrency Level:      1000
Time taken for tests:   13.865 seconds
Complete requests:      50000
Failed requests:        0
Total transferred:      8450000 bytes
HTML transferred:       2600000 bytes
Requests per second:    3606.24 [#/sec] (mean)
Time per request:       277.297 [ms] (mean)
Time per request:       0.277 [ms] (mean, across all
concurrent requests)
Transfer rate:          595.17 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       37   37   4.5     37    1042
Processing:    37  236 150.3    215    1388
Waiting:       37  233 151.4    212    1388
Total:         75  273 150.4    252    1425
```

```
Server Hostname:        35.228.175.143
Server Port:            80

Document Path:          /
Document Length:        62 bytes

Concurrency Level:      1000
Time taken for tests:   5.682 seconds
Complete requests:      50000
Failed requests:        0
Total transferred:      8950000 bytes
HTML transferred:       3100000 bytes
Requests per second:    8799.31 [#/sec] (mean)
Time per request:       113.645 [ms] (mean)
Time per request:       0.114 [ms] (mean, across all
concurrent requests)
Transfer rate:          1538.16 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       37   40   3.6     39      58
Processing:    37   70  37.8     56     412
Waiting:       37   67  37.3     54     412
Total:         74  110  37.4     98     450
```

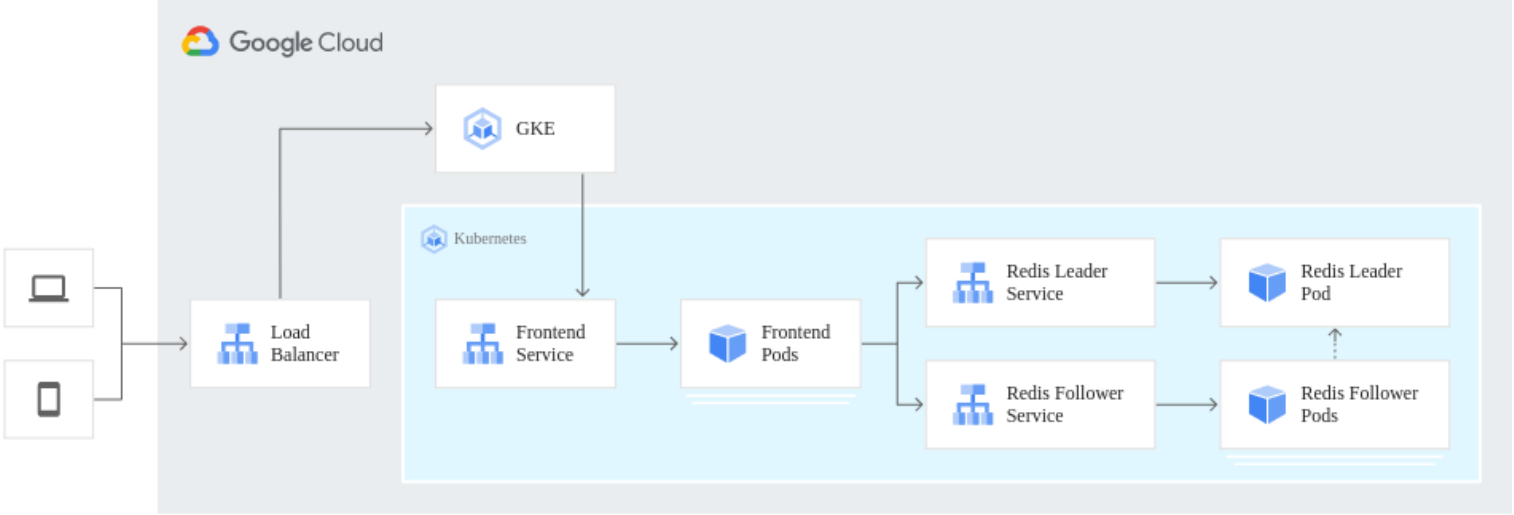
Chuleta:

`apt install apache2-utils`	`ab -n TOTAL -c CONCURRENT URL`


Google Cloud Platform
Unifi-GCP-3
Buscar productos y recursos
Explorador de métricas
CONFIGURATION
MQL
Gráfico de líneas
1H
6H
1D
1W
1M
6W
CUSTOM
Save Chart
Las unidades de consultas que usan el mismo eje Y deben ser las mismas para que el gráfico se renderice correctamente.
Las unidades de consultas que usan el mismo eje Y deben ser las mismas para que el gráfico se renderice correctamente.
Time series
Resource type
Kubernetes Pod
Metric
Bytes received
Filters
app = test4
ADD FILTER
How do you want to view that data?
Group by
25 nov. 2021 00:42:48
SELECTED
test4-68b66df6c9-w27fn 70.07KiB/s
test4-68b66df6c9-9stch 91.29KiB/s
test4-68b66df6c9-zfh8v 84.26KiB/s
test4-68b66df6c9-w27fn 70.07KiB/s
pod_name
Valor
test4-68b66df6c9-9stch 0.49KiB/s
test4-68b66df6c9-w27fn 65.4KiB/s
test4-68b66df6c9-zfh8v 49.34KiB/s


Google Cloud Platform
Unifi-GCP-3
Buscar productos y recursos
Explorador de métricas
CONFIGURATION
MQL
Gráfico de líneas
1H
6H
1D
1W
1M
6W
CUSTOM
Save Chart
Time series
Resource type
Kubernetes Pod
Metric
Bytes received
Filters
pod_name = hello-server
ADD FILTER
How do you want to view that data?
Group by
Aggregator
none
Minimum alignment period
1 minute
Metric
pod_name
Valor
received_bytes_count
hello-server
307.11KiB/s

Libro de visitas con Redis y PHP



1. Entender qué son los comandos gcloud, ver credenciales

```
gcloud auth list
```

   a. Ver proyectos:

```
gcloud projects list
```

   b. Seleccionar proyecto:

```
gcloud config set project fair-analogy-325309
```

2. Seleccionamos zona por defecto para *compute*

```
gcloud compute zones list
gcloud config set compute/zone
```

3. Desplegar el cluster

```
gcloud container clusters create  $NOMBRE-2 --num-nodes=4
```

4. Para ver los detalles del cluster

```
gcloud container clusters list
gcloud container clusters describe $NOMBRE_2
```

5. Creamos los secretos para poder gestionar el contenedor desde la línea de comandos de Cloud Shell:

```
gcloud container clusters get-credentials guestbook
```

6. El libro de visitas utiliza Redis (una base de datos alojada en memoria) para almacenar los datos. Los datos son escritos en un líder y leídos desde varios seguidores.

7. Debemos copiar los manifiestos:

```
git clone https://github.com/GoogleCloudPlatform/kubernetes-engine-samples
cd kubernetes-engine-samples/guestbook
git checkout abbb383
```

8. Implementamos el líder de redis utilizando el archivo YAML:

```
kubectl apply -f redis-leader-deployment.yaml
```

9. Comprobamos mediante la GUI y *gcloud* que el despliegue se ha producido de forma correcta:

```
kubectl get pods
```

10. Revisamos los logs que se haya iniciado de forma correcta y está listo para recibir conexiones del resto de pods.

```
kubectl logs deployment/redis-leader
```

11. Creamos el servicio (recordar la necesidad de exponer los servicios)

```
kubectl apply -f redis-leader-service.yaml
```

12. Comprobar con GUI y gcloud:

```
kubectl get service
```

13. Crear el despliegue para los seguidores de redis

```
kubectl apply -f redis-follower-deployment.yaml
```

14. Comprobamos mediante GUI y gcloud los Pods que están en funcionamiento

```
kubectl get pods
```

15. (Explicar leader/follower) Comprobamos que existe conexión entre los seguidores:

```
kubectl logs deployment/redis-follower
```

16. Crear el servicio para los seguidores de redis

```
kubectl apply -f redis-follower-service.yaml
```

17. Comprobar con GUI y gcloud:

```
kubectl get service
```

18. Creamos el despliegue para el *frontend* (ver código):

```
kubectl apply -f frontend-deployment.yaml
```

19. Comprobamos los pods:

```
kubectl get pods [-l tier=frontend]
```

20. Creamos un service para el frontend ("exponer") y comprobamos:

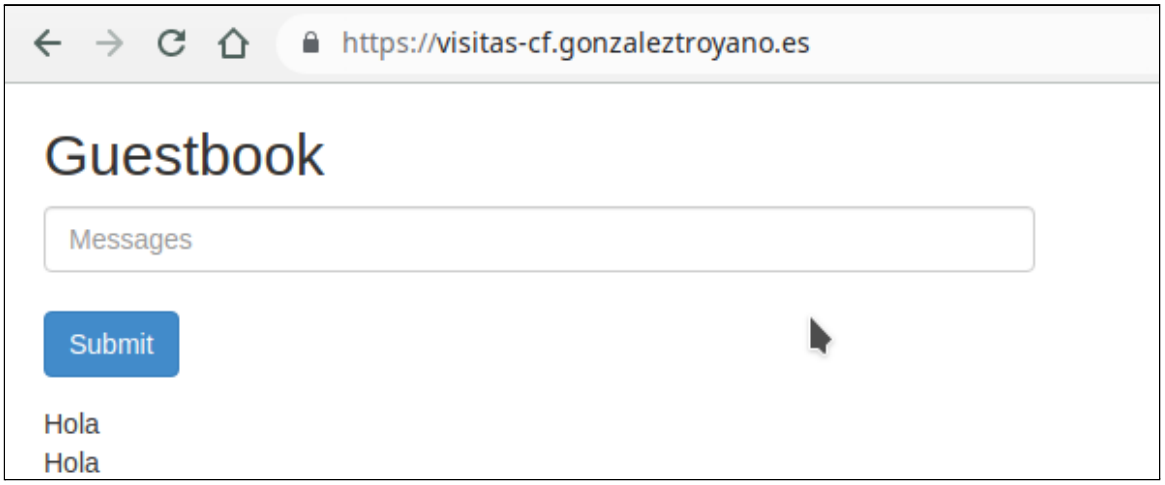
```
kubectl apply -f frontend-service.yaml
```

21. (Esperar a la IP externa). Añadir registro A con un dominio. Desde el instituto no se permite acceder. Probar desde el móvil.
22. Probar a acceder y añadir registros.
23. Para escalar el frontend:

```
kubectl scale deployment frontend --replicas=X
```

24. Eliminar los recursos (💸)

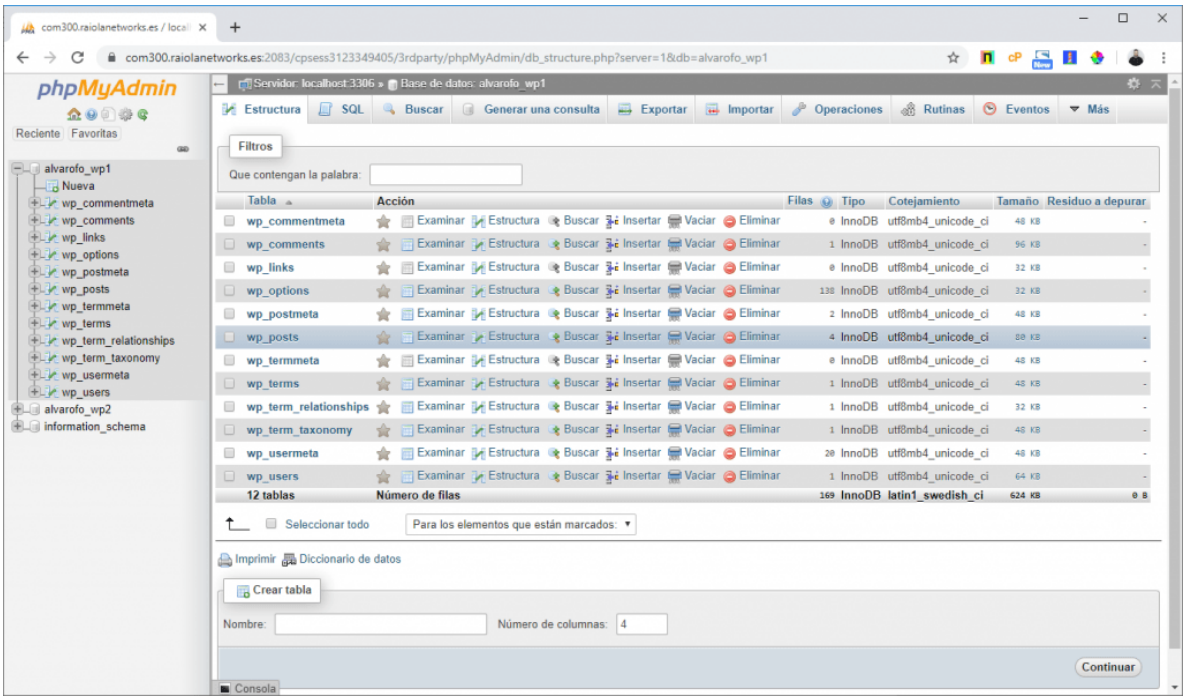
Si además utilizamos Cloudflare como CDN, DDoS mitigation service y LB[27]:



[27] También podríamos utilizar el servicio de LB de GCP.

Wordpress con PersistentDisk y CloudSQL

Como ya sabemos gracias a IAW, wordpress almacena TODO en una base de datos relacional.



28

Atención, pregunta: ¿Por qué no podemos/debemos almacenar datos en un pod?

Qué aprenderemos:

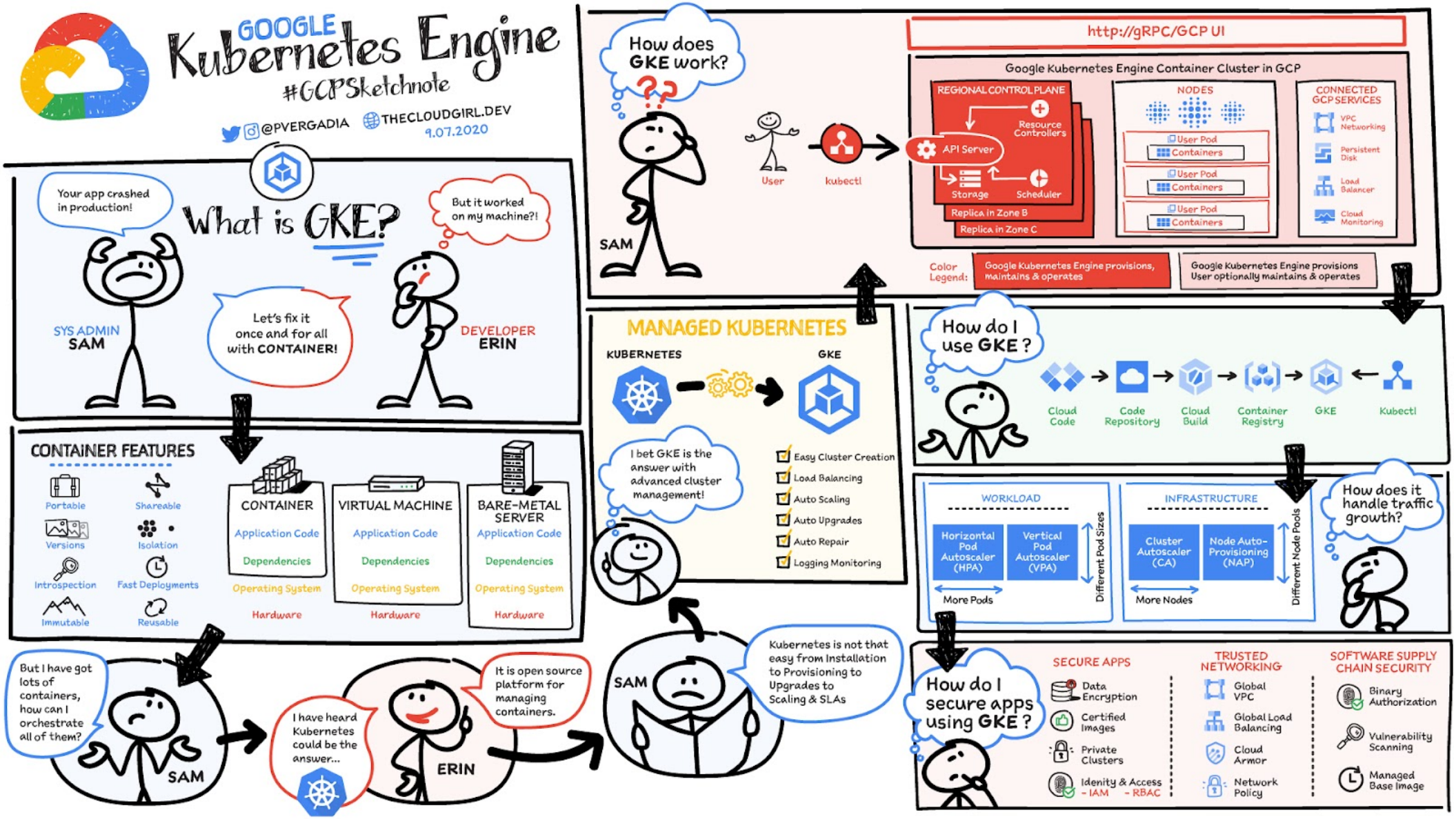
- Crear un clúster de GKE
- Crear un PV y un PVC con respaldo de Persistent Disk
- Crear una instancia de Cloud SQL para MySQL
- Implementar WordPress
- Configurar tu blog de WordPress

Ver instrucciones en [este enlace](#).

Otras opciones

- Deploy Kubernetes Load Balancer Service with Terraform
- NGINX Ingress Controller on Google Kubernetes Engine
- Setting up a Minecraft Server on Google Compute Engine

[28] imagen: alvarofontela.com, consultado el 25 nov 2021


GOOGLE Kubernetes Engine
#GCPSketchnote
@PVERGADIA
THECLOUDGIRL.DEV
9.07.2020
What is GKE?
Your app crashed in production!
But it worked on my machine?!
Let's fix it once and for all with CONTAINER!
SYS ADMIN SAM
DEVELOPER ERIN
CONTAINER FEATURES
Portable
Shareable
Versions
Isolation
Introspection
Fast Deployments
Immutable
Reusable
CONTAINER
Application Code
Dependencies
Operating System
Hardware
VIRTUAL MACHINE
Application Code
Dependencies
Operating System
Hardware
BARE-METAL SERVER
Application Code
Dependencies
Operating System
Hardware
But I have got lots of containers, how can I orchestrate all of them?
SAM
I have heard Kubernetes could be the answer...
ERIN
It is open source platform for managing containers.
SAM
Kubernetes is not that easy from Installation to Provisioning to Upgrades to Scaling & SLAs
How does GKE work?
SAM
User
kubectl
http://gRPC/GCP UI
Google Kubernetes Engine Container Cluster in GCP
REGIONAL CONTROL PLANE
Resource Controllers
API Server
Storage
Scheduler
Replica in Zone B
Replica in Zone C
NODES
User Pod
Containers
User Pod
Containers
User Pod
Containers
CONNECTED GCP SERVICES
VPC Networking
Persistent Disk
Load Balancer
Cloud Monitoring
Color Legend:
Google Kubernetes Engine provisions, maintains & operates
Google Kubernetes Engine provisions User optionally maintains & operates
MANAGED KUBERNETES
KUBERNETES
GKE
I bet GKE is the answer with advanced cluster management!
Easy Cluster Creation
Load Balancing
Auto Scaling
Auto Upgrades
Auto Repair
Logging Monitoring
How do I use GKE ?
Cloud Code
Code Repository
Cloud Build
Container Registry
GKE
Kubectl
WORKLOAD
Horizontal Pod Autoscaler (HPA)
Vertical Pod Autoscaler (VPA)
Different Pod Sizes
More Pods
INFRASTRUCTURE
Cluster Autoscaler (CA)
Node Auto-Provisioning (NAP)
Different Node Pools
More Nodes
How does it handle traffic growth?
How do I secure apps using GKE ?
SECURE APPS
Data Encryption
Certified Images
Private Clusters
Idenity & Access - IAM - RBAC
TRUSTED NETWORKING
Global VPC
Global Load Balancing
Cloud Armor
Network Policy
SOFTWARE SUPPLY CHAIN SECURITY
Binary Authorization
Vulnerability Scanning
Managed Base Image

Bibliografía

¿Cuánto cuesta comprar un avión?
Cloud Computing
AWS | Elastic compute cloud (EC2) de capacidad modificable en la nube
Almacenamiento de datos seguro en la nube (S3)
Cloud Storage
Professional Cloud Architect Certification
PayPal | Customers
https://services.google.com/fh/files/misc/next-19-google-cloud-customer-voices-digital-book.pdf
Best practices for operating containers | Cloud Architecture Center
5 principles for cloud-native architecture—what it is and how to master it
Open data lakehouse on Google Cloud
Want repeatable scale? Adopt infrastructure as code on GCP
Site Reliability Engineering (Libros y metodología)
Cloud Run: What no one tells you about Serverless (and how it's done)
Open data lakehouse on Google Cloud
How Lowe's improved incident response processes with SRE
Where should I run my stuff? Choosing a Google Cloud compute option
What is Compute Engine? Use cases, security, pricing and more
Google Cloud: Cloud Computing Services
Cloud Services - Amazon Web Services (AWS)
Microsoft Azure: Servicios de informática en la nube
Web hosting, cloud y servidores dedicados
Reliable Secure Cloud Solutions to Empower Your Global Business
¿Qué es IBM Cloud?
Set Up Network and HTTP Load Balancers
example-gke-k8s-service-lb
Los contenedores en comparación con las máquinas virtuales
https://trends.google.com/trends/explore?date=all&q=Docker,%2Fm%2F07yf2
Using StatefulSets - Kubernetes documentation
DaemonSet - Kubernetes documentation
DaemonSet - Kubernetes documentation
Volumes - Kubernetes documentation
Kubernetes 101: Pods, Nodes, Containers, and Clusters | Google Cloud - Community
Deploy code faster: with CI/CD and Kubernetes
Introducción a la arquitectura de Kubernetes
Creating Database for WordPress
Entendiendo la base de datos de WordPress [MySQL o MariaDB]
Instance groups | Compute Engine Documentation | Google Cloud
Sistemas con estado y sin estado - Documentación de Red Hat
Instance templates | Compute Engine Documentation | Google Cloud
Storage classes | Google Cloud
Backend service-based external TCP/UDP Network Load Balancing overview | Google Cloud
Cloud Load Balancing overview
Maglev: A Fast and Reliable Software Network Load Balancer – Google Research
Google_Cloud_networking_in-depth:_How_Andromeda_2.2_enables_high-throughput_VMs | Google Cloud Blog
Traffic management overview for global external HTTP(S) load balancers
En pablo-glez, bq-public-datasheets. Public1 Public2 Sh