

# Técnicas de Cloud Computing

Administración de Sistemas

Unai Lopez Novoa  
unai.lopez@ehu.eus

eman ta zabal zazu



Universidad  
del País Vasco

Euskal Herriko  
Unibertsitatea

# Introducción

- Definición de computación en la nube<sup>1</sup>:

*Cloud computing* es la disponibilidad bajo demanda de recursos de computación como servicios a través de Internet.

Esta tecnología evita que las empresas tengan que encargarse de aprovisionar, configurar o gestionar los recursos y permite que paguen únicamente por los que usen.

- Es el uso de servidores remotos para almacenar, administrar y procesar datos bajo demanda.



<sup>1</sup>De: <https://cloud.google.com/learn/what-is-cloud-computing?hl=es>

# Introducción

- Hay 3 tipos principales de servicios en la nube<sup>1</sup>:
  - Infrastructure as a Service (**IaaS**):  
P.e.: máquinas virtuales, almacenamiento, ...
  - Platform as a Service (**PaaS**):  
P.e.: entornos de desarrollo, bases de datos, ...
  - Software as a Service (**SaaS**)  
P.e.: software usable por usuarios finales (Dropbox, Office 365, ...)
- *Cada uno cubre cierto grado de gestión.*





<sup>1</sup>Fuente: <https://www.redhat.com/es/topics/cloud-computing/iaas-vs-paas-vs-saas>

# Introducción

- Hay 3 tipos principales de servicios en la nube<sup>1</sup>:

	On-site	IaaS	PaaS	SaaS
	Applications	Applications	Applications	Applications
	Data	Data	Data	Data
	Runtime	Runtime	Runtime	Runtime
	Middleware	Middleware	Middleware	Middleware
	O/S	O/S	O/S	O/S
	Virtualization	Virtualization	Virtualization	Virtualization
	Servers	Servers	Servers	Servers
	Storage	Storage	Storage	Storage
	Networking	Networking	Networking	Networking

 You manage

 Service provider manages



<sup>1</sup>Fuente: <https://www.redhat.com/es/topics/cloud-computing/iaas-vs-paas-vs-saas>

# Google Cloud Platform

- IaaS:
  - Compute Engine, Kubernetes Engine, ...
  - Cloud Storage, ...
- PaaS:
  - Cloud Run, App Engine, ...
- SaaS:
  - Google Workspace (Gmail, Docs, ...)



# Cloud Computing

- Hay servicios ofrecidos por proveedores Cloud que se pueden reemplazar localmente.
  - P.e. crear MVs manualmente, gestionar usuarios, ...
- *Pero ...*
- Hay servicios exclusivos de entornos Cloud.
  - En este tema veremos los más relevantes.



# Contenido

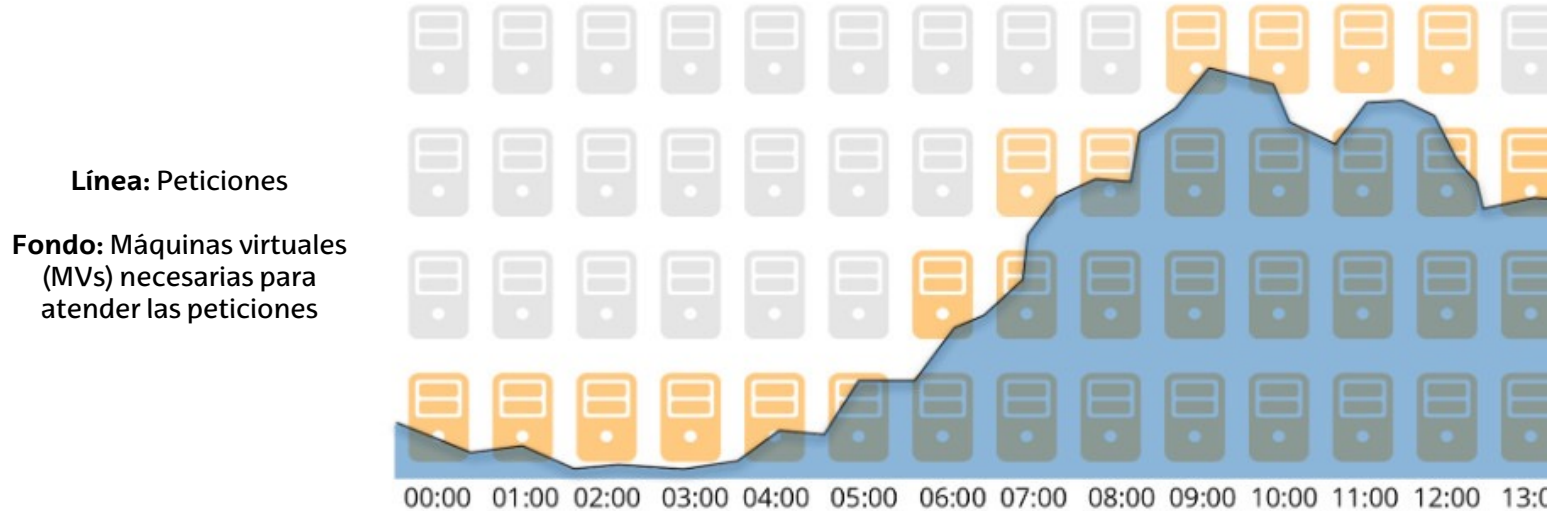
1. Introducción
2. Auto-escalado
3. Serverless
4. BBDD como servicio



# Auto-escalado

- Ejemplo:

- Nuestra aplicación recibe las siguientes peticiones un día normal:



- Para satisfacer la demanda sin exceso de costes:
  - Entre las 00:00 y 05:00, 1 MVs
  - Entre las 06:00 y 07:00, 2 MVs
  - ...





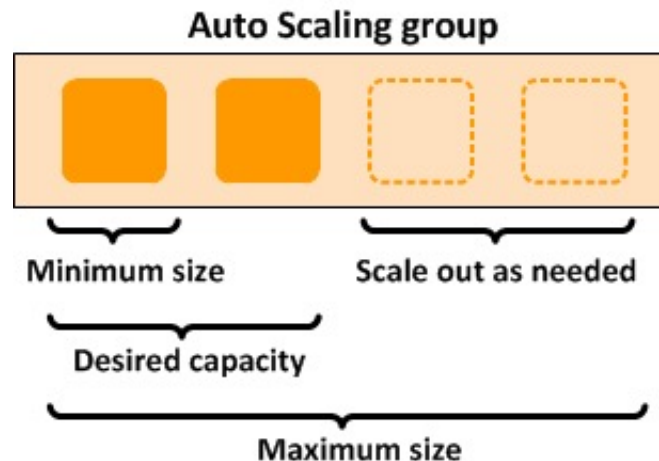
# Auto-escalado

- Las técnicas de auto-escalado permiten crear y eliminar recursos de forma automática en base a umbrales.
- Ventajas principales:
  - Ahorro de costes, manteniendo la calidad de servicio.
  - Mayor tolerancia a fallos.
- Se suelen usar junto con técnicas de balanceo de carga.
  - Para repartir peticiones/cómputo entre los recursos disponibles.



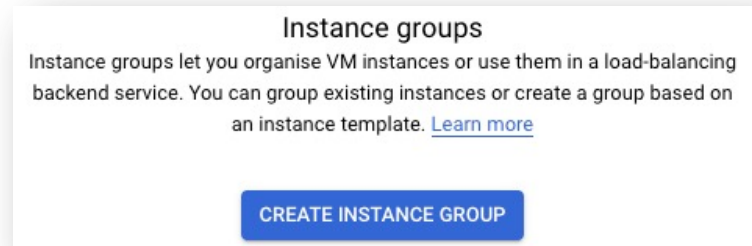
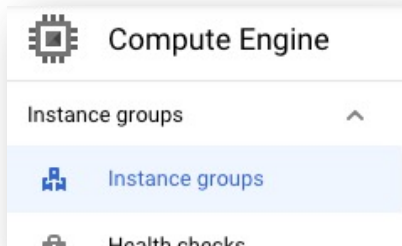
# Auto-escalado

- Un grupo de auto-escalado controla cuántas MVs tener activas en cada momento.
  - El nº de MVs se decide en base al uso de recursos.
    - P.e. uso de de CPU.
- Ejemplo:
  - Grupo con 2 MVs activas, tendrá mínimo 1 MV y máximo 4 MVs.



# Auto-escalado en GCP

- Compute Engine permite crear MVs con auto-escalado usando la función Grupos de Instancias.



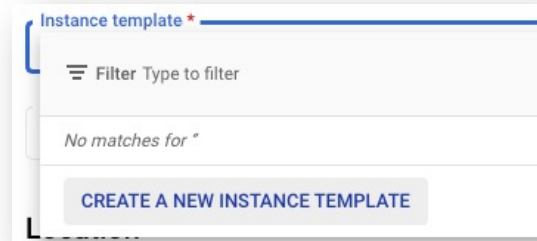
- Proceso de 2 pasos:
  - 1) Definir una plantilla de máquina virtual.
  - 2) Definir las características del grupo de instancias.
    - Incluyendo los umbrales que controlan el auto-escalado.



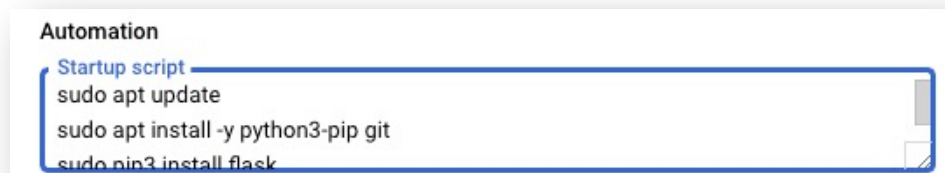
# Auto-escalado en GCP

## 1) Definir una plantilla de máquina virtual.

- Configuración equivalente a la creación de una nueva MV.
  - Todos las MVs del grupo se crearán en base a la plantilla.



- En la plantilla, se puede definir un script que se ejecute automáticamente al inicio de la MV.
  - Permite la instalación y arranque automático del software en cada MV.
  - Sección "Gestión" de la plantilla.



# Auto-escalado en GCP

## 2) Definir las características del grupo de instancias.

- Definir el nº mínimo y máximo de MVs que habrá en el grupo.
- Configurar el auto-escalado:
  - Por defecto, Compute Engine monitoriza el uso de CPU en cada MV.
  - Cuando el % de uso de CPU supera el umbral definido, crea nuevas MVs.
  - Cuando el % de uso de CPU es reducido, elimina MVs.

### Auto-scaling

Use auto-scaling to automatically add and remove instances to the group for periods of high and low load. [Learn more](#)

**Auto-scaling mode**  
On: add and remove instances to and from the group

Minimum number of instances \*

1

?

Maximum number of instances \*

5

?

**Autoscaling metrics**  
Use metrics to help determine when to scale the group. [Learn more](#)

CPU utilisation: 60% (default)

Predictive auto-scaling is off

▼



# Auto-escalado en GCP

## 2) Definir las características del grupo de instancias.

- Es conveniente definir correctamente el periodo de inicialización
- Es el periodo que tarda nuestra aplicación en iniciarse.
  - Compute Engine no monitoriza el % de uso de CPU durante el periodo de inicialización.
  - Se considera como un tiempo en el que la MV no responde a peticiones.
  - Por defecto, 60 segundos.

**Initialisation period** ^

Specify how long it takes for your app to initialise from boot time until it is ready to serve. [Learn more](#)

Initialisation period \*

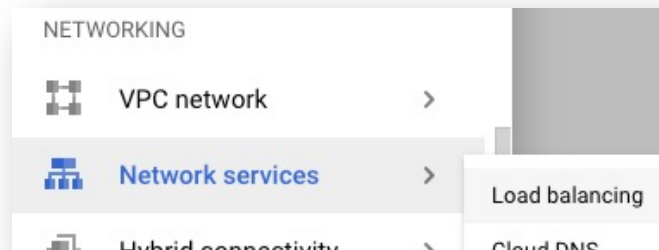
60

seconds ?



# Auto-escalado en GCP

- Se puede añadir un balanceador de carga que reparta peticiones externas entre las diferentes MVs.
  - Provee una única IP pública para el grupo de instancias.



- Se configura en 2 partes:
  - Front-end: IP pública y puertos a exponer.
  - Back-end: vínculo con el grupo de instancias.



# Auto-escalado en GCP

- Configuración de Back-end en el Load Balancer:
  - Definir el grupo de instancias y los puertos que utiliza la aplicación:



New backend

Instance group \*  
mi-grupo-instancias

Port numbers \*  
80

- Definir un "Health check"
  - Forma de comprobar que la MV funciona y está sirviendo una aplicación.
  - Se hace una petición a un puerto en un intervalo definido.



Name \*  
mi-health-check-web

Lowercase, no spaces.

Description  
Sondeo del puerto 80

Protocol  
HTTP

Port \*  
80





# Ejercicio 1

- Definir un grupo de instancias con las siguientes características:
  - Plantilla de MVs:
    - MVs tipo n1-standard-1, SSOO Ubuntu 22.04 LTS.
    - Se debe instalar el servidor web Apache automáticamente al inicio.
  - Cantidad de MVs: Mínimo 1, Máximo 4.
  - Umbral de auto-escalado: 80%
- Acceder por ssh a la 1ª MV del grupo y utilizar stress-ng para provocar un 90% de uso de CPU durante 2 minutos.
  - Verificar que se crean nuevas MVs.
  - Verificar que cada MV tiene un servidor Web en marcha.



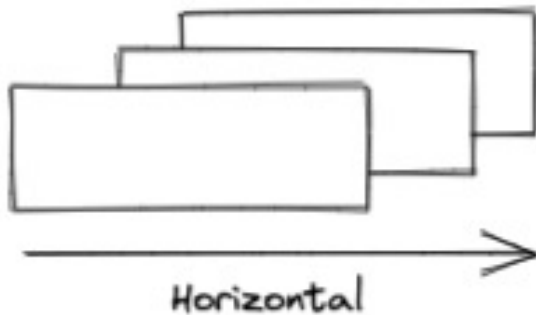
# Auto-escalado

- Las técnicas de auto-escalado se pueden aplicar a múltiples tipos de recursos.
  - No sólo máquinas virtuales.
- Es posible aplicar auto-escalado en Kubernetes.
  - Generalmente, aplicado a los Pods de los Deployment.



# Auto-escalado

- Hay 2 tipos principales de auto-escalado:



Añadir más recursos  
del mismo tipo



Aumentar la capacidad  
del recurso



# Auto-escalado

- Se consideran 2 tipos de auto-escalado:
  - ¿Cuándo usar cada uno?
- *Horizontal*: Servicios sin estado.
  - P.e. servidores Web.
- *Vertical*: Servicios con estado.
  - P.e. servidores de BBDD tradicionales, servicios heredados.



# Auto-escalado en Kubernetes

- Auto-escalado horizontal
  - Objeto HorizontalPodAutoscaler
    - Controla automáticamente el número de Pods réplica que contiene un Deployment.
    - Ejemplo:
      - Aplicado sobre un Deployment con nombre "nginx"

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: mi-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

Nombre del Deployment  
sobre el que va a operar

Promedio deseado de uso de  
CPU de todos los Pods del  
Deployment. Se calcula con los  
datos del último minuto



# Auto-escalado en Kubernetes

- Auto-escalado horizontal
  - Objeto HorizontalPodAutoscaler

- Aplicar el objeto:

```
kubectl apply -f <objeto>.yaml
```

- Mostrar información sobre el escalado:
    - El parámetro opcional --watch mantiene la información visible.

```
kubectl get hpa <--watch>
```

- Mostrar información sobre las decisiones de escalado:

```
kubectl describe hpa
```



# Auto-escalado en Kubernetes

- Auto-escalado vertical
  - Objeto VerticalPodAutoscaler
    - Ejemplo:

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: mi-vpa
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  resourcePolicy:
    containerPolicies:
      - containerName: "*"
        minAllowed:
          cpu: 100m
          memory: 50Mi
        maxAllowed:
          cpu: 1
          memory: 500Mi
        controlledResources: ["cpu", "memory"]
```

Políticas de  
escalado vertical



# Auto-escalado en Kubernetes

- Auto-escalado vertical

- Objeto VerticalPodAutoscaler

- Es posible utilizarlo sin fijar recursos. Calculará valores mínimos, medios y máximos para el objeto a escalar.
    - Ejemplo:

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: mi-vpa
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  updatePolicy:
    updateMode: Off
```

Indicar "Off" para que las recomendaciones no se apliquen o "Auto" para que se apliquen





# Auto-escalado en Kubernetes

- Auto-escalado vertical
  - Objeto VerticalPodAutoscaler
    - Aplicar el objeto:

```
kubectl apply -f <objeto>.yaml
```

- Mostrar información sobre el escalado:
  - El parámetro opcional --watch mantiene la información visible.

```
kubectl get vpa <--watch>
```

- Mostrar información sobre las decisiones de escalado:

```
kubectl describe vpa
```



# Auto-escalado en Kubernetes

- Auto-escalado vertical
  - Objeto VerticalPodAutoscaler
    - Para mostrar: las recomendaciones de valores mínimos, medios y máximos para el objeto:

```
kubectl describe vpa
```

- Ejemplo:

```
recommendation:
  containerRecommendations:
    - containerName: nginx
      lowerBound:
        cpu: 40m
        memory: 3100k
      target:
        cpu: 60m
        memory: 3500k
      upperBound:
        cpu: 831m
        memory: 8000k
```



# Auto-escalado en Kubernetes

- Obtener métricas de los recursos:

- De un Pod:

```
kubectl top pod
```

- De un nodo:

```
kubectl top node
```

- En las asignaciones de tiempo de CPU, se utiliza la unidad de "miliunidades".

- Denotado como "m".
  - Equivale a la milésima parte de una unidad.
  - Ejemplo: "55m" equivale a un uso de CPU del 0.55%.



# Ejercicio 2

- *Realizar este ejercicio en parejas, seréis A y B.*
- A crea una aplicación Kubernetes con:
  - Un Deployment que gestione Pods con la imagen "nginx"
  - Un LoadBalancer que expone el puerto 80 del Pod al exterior
  - Un VerticalPodAutoscaler aplicado al Deployment en modo Auto
- B crea un script que realiza peticiones a la web de Nginx cada 0.05 segundos.
- B ejecuta el script y A monitoriza las recomendaciones del VPA.



# Google Cloud

- El auto-escalado en Compute Engine y Kubernetes Engine automatiza el control de la infraestructura.
- *Pero ...*
- En ocasiones, tanto control es prescindible.
  - Si nuestro código no es muy complejo, podemos usar otras plataformas con mayor abstracción.



# Serverless

- Ejecución de código sin definición infraestructura.
  - O con una definición mínima.
  - La gestión de recursos y escalabilidad es transparente.
- Se denomina Computación *Serverless* ("Sin servidores").
  - Aunque realmente los hay, pero no se muestran.
- Generalmente, se paga por cada ejecución del código.
  - Si el código no se usa, no se generan gastos.



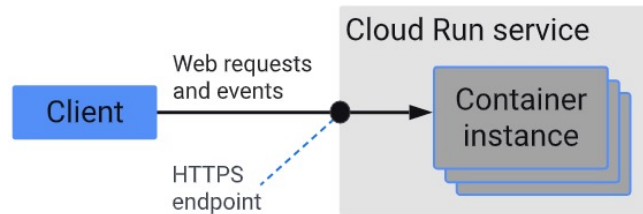
# Serverless en Google Cloud

- Cloud Run
  - Gestiona la ejecución Serverless de contenedores.
  - Por defecto, se factura en base a las solicitudes HTTP que recibe el contenedor.
- Cloud Functions
  - Gestiona la ejecución Serverless de funciones de código.
    - Acepta código en Python, Java, PHP, Ruby, NodeJs, .NET y Go.
  - Se denomina FaaS (Function as a Service).
  - Se factura por el tiempo de ejecución de cada función.
    - Redondeado a 100 milisegundos.



# Cloud Run

- Permite ejecución de contenedores en modo Serverless<sup>1</sup>.
- Pensado para contenedores que responden a peticiones HTTP.
  - Proporciona un endpoint HTTPS para cada contenedor.



- Recomendado para aplicaciones web o APIs REST.



<sup>1</sup>Cloud Run, Google Cloud: <https://cloud.google.com/run/docs/overview/what-is-cloud-run?hl=es-419>



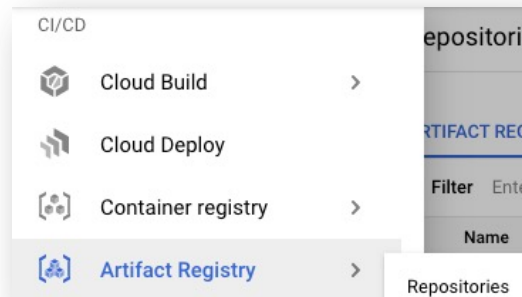
# Cloud Run

- Limitaciones:
  - No hay gestión del estado.
    - Cada ejecución de un contenedor debe ser atómica.
  - No proporciona persistencia.
    - Cloud Run no guarda datos de ningún tipo.
    - Si nuestro contenedor necesita persistencia, es necesario gestionarla aparte.
  - No se provee un DNS interno.
    - El endpoint asignado a cada contenedor siempre es público.
  - Requiere utilizar Artifact Registry, el registro Docker de Google.
    - No es posible cargar imágenes directamente (p.e.) desde DockerHub.



# Cloud Run + Artifact Registry

- Antes de crear un contenedor con Cloud Run, la imagen debe estar en Artifact Registry.
  - Se encuentra en la sección CI/CD de Google Cloud.



- Para subir una imagen, es necesario crear un repositorio:



# Cloud Run + Artifact Registry

- Para subir una imagen existente en DockerHub a Artifact Registry, se recomienda utilizar Cloud Shell:

- Descargar la imagen desde Docker Hub.

```
docker pull <imagen>
```

- Renombrar imagen para que su registro sea Artifact Registry.

```
docker tag <imagen> <repositorio-artifact-registry>/<imagen>
```

- Activar permiso *push* para repositorio en Artifact Registry.

- Donde <región-repositorio> es p.e. europe-west3-docker.pkg.dev

```
gcloud auth configure-docker <región-repositorio>
```

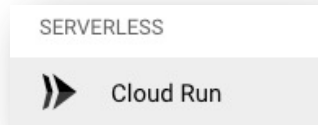
- Subir a Artifact Registry

```
docker push <repositorio-artifact-registry>/<imagen>
```



# Cloud Run

- Ejecutar un contenedor con Cloud Run.
  - Abrir Cloud Run y crear un nuevo servicio.



- En la creación del servicio, indicar, al menos:
  - Imagen de contenedor en Artifact Registry.
  - Tipo de tráfico permitido (externo/interno de Google Cloud).
  - Autenticación permitida.
  - Puerto en el que recibir peticiones.
- Cloud Run devuelve una URL con el endpoint en el que se está ejecutando el contenedor.
  - Formato: *https://<nombre-imagen>-...-run.app*



# Cloud Run

- Internamente, los contenedores de Cloud Run se ejecutan sobre Knative.
  - Una versión de Kubernetes orientada a Serverless.
- Se puede ver el objeto Kubernetes creado para ejecutar el cotenedor
  - Detalles del servicio, sección YAML

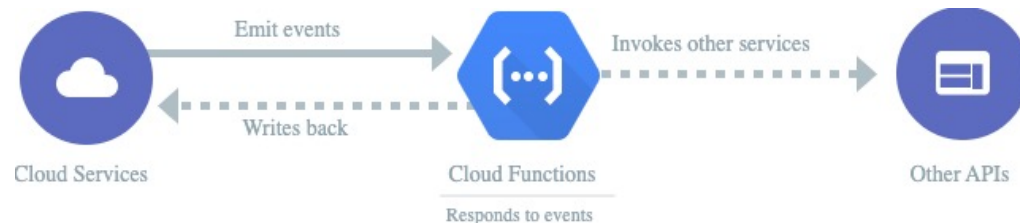


```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: <nombre>
  namespace: '<código>'
...
```



# Cloud Functions

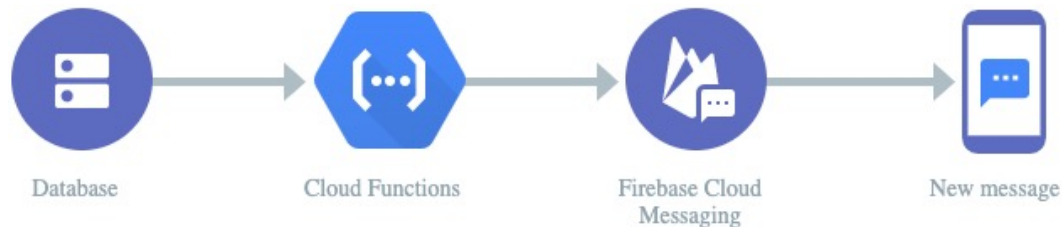
- Ejecución de código sin necesidad de definir instancias virtuales ni contenedores.
- Definido como Function as a Service (FaaS).
- Adecuado para código sencillo que se debe ejecutar como reacción a un evento.



# Cloud Functions

- Ejemplos de uso:

- 1) Generar un mensaje de texto tras una inserción en una tabla de una BBDD.

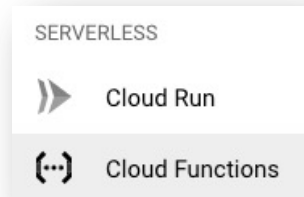


- 2) Limpieza/pre-procesado de datos.



# Cloud Functions

- Ejecutar código con Cloud Functions
  - Abrir Cloud Functions y crear una nueva función.



- La creación de la función se hace en 2 pasos:
  - Configuración de la función (Autenticación y forma de acceso)
  - Definición del código (en Python, Java, ...)
- Cloud Functions devuelve una URL con un endpoint para utilizar la función
  - Formato: *https://<nombre-función>- ... - run.app*





# Cloud Functions

- Internamente, una función en Cloud Functions se ejecuta en un contenedor de Cloud Run.
  - Se puede verificar desde la consola de Cloud Run.
- La autenticación se gestiona con usuarios y tokens de Google Cloud.
  - Ejemplo: llamar a una función usando un token de autenticación y un JSON como dato.

```
curl
-H "Authorization: bearer $(gcloud auth print-identity-token)"
-H "Content-Type: application/json"
https://mi-funcion- ... .run.app
-d '{"name": "Unai"}'
```

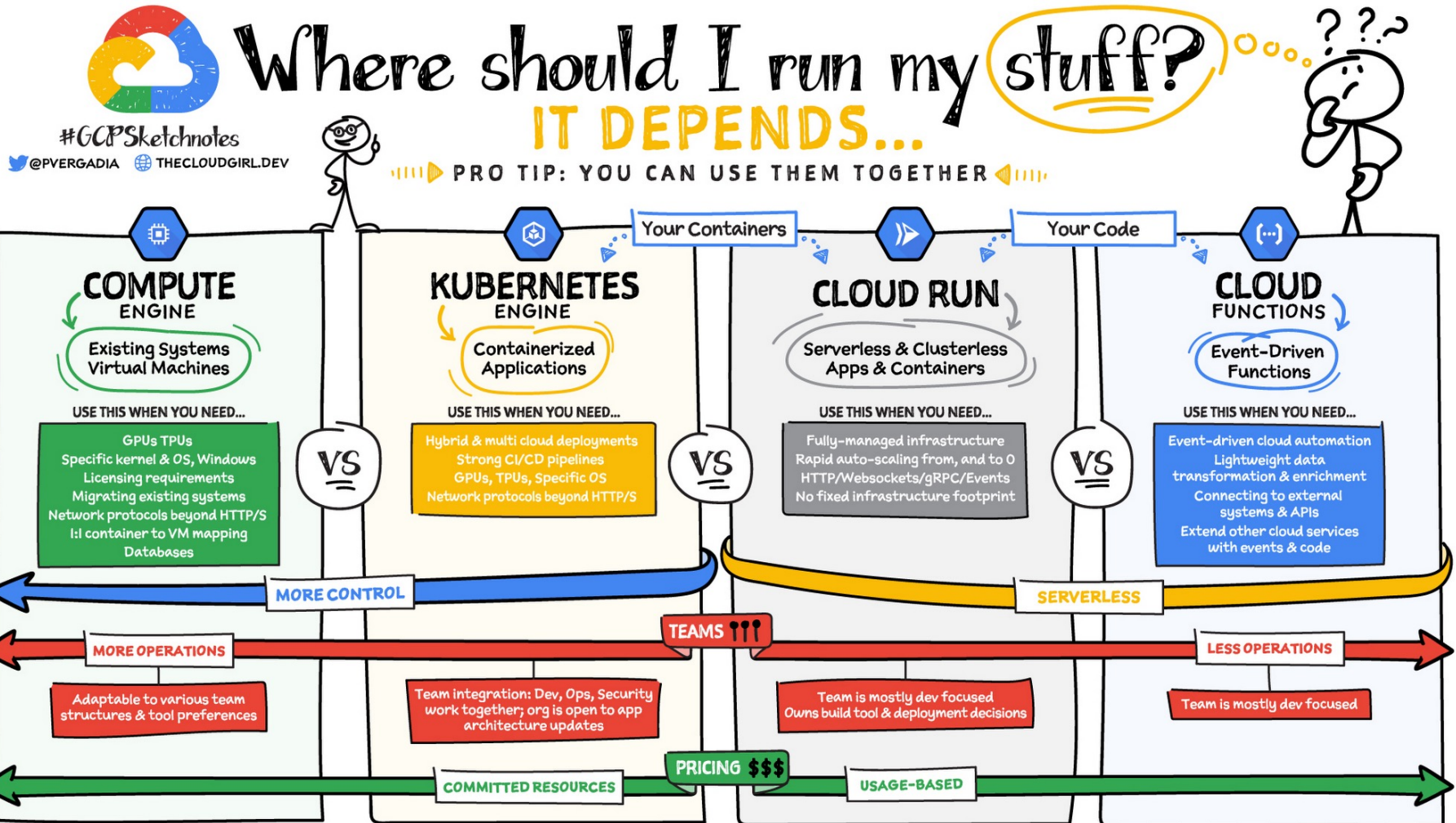


# Ejercicio 3

- El endpoint <https://jsonplaceholder.typicode.com/users/> contiene los datos de 10 personas aleatorias.
  - Se puede acceder a los datos de una persona concreta añadiendo /X al final de la URL, donde X es un número de 1 a 10.
- Crear una función con Cloud Function que, en cada ejecución, recupere los datos de una de las personas de forma aleatoria y muestre su nombre.



# Cómputo en GCP



# Cloud Computing

- Comparativa de servicios en diferentes proveedores:

Proveedor	Máquinas virtuales	Motor Kubernetes	Function as a Service (FaaS)
Amazon Web Services (AWS)	Elastic Compute Cloud	Elastic Kubernetes Service	Lambda
Microsoft Azure	Azure Virtual Machines	Azure Kubernetes Service	Azure Functions
Google Cloud Platform	Compute Engine	Google Kubernetes Engine	Cloud Functions
IBM Cloud	IBM Virtual Servers	IBM Kubernetes Service	IBM Cloud Functions
Oracle Cloud	Oracle Compute	Oracle Container Engine	Oracle Functions



# Cloud Computing

- Además de cómputo, los proveedores Cloud ofrecen SGBDs como servicio.
- Abstraen la complejidad de mantener un SGBD.
  - Escalado, actualizaciones, ... se gestionan transparentemente.
- Permiten diseñar aplicaciones 100% basadas en servicios en la nube.
- Se factura por minutos de uso y almacenamiento.



# BBDD como servicio en Google Cloud

## BBDD Relacionales

---



Cloud  
SQL



Cloud  
Spanner

## BBDD No-SQL

---



Cloud  
Firestore



Cloud  
Bigtable

## En memoria

---



Memorystore  
(Redis/Memcached)

# BBDD relacionales en Google Cloud

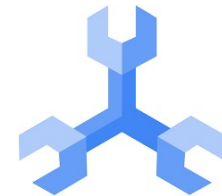
- Cloud SQL

- Instancias de MySQL, PostgreSQL o SQL Server.
- Escalabilidad vertical.
- Apropiado para aplicaciones tradicionales.



- Cloud Spanner.

- Tecnología propia de Google.
- Escalabilidad vertical y horizontal.
- Apropiado para aplicaciones de gran escala.



# BBDD No-SQL en Google Cloud

- Cloud Firestore

- BD orientada a documentos.
- Escalabilidad horizontal.
- Apropriada como reemplazo de MongoDB (o similar).



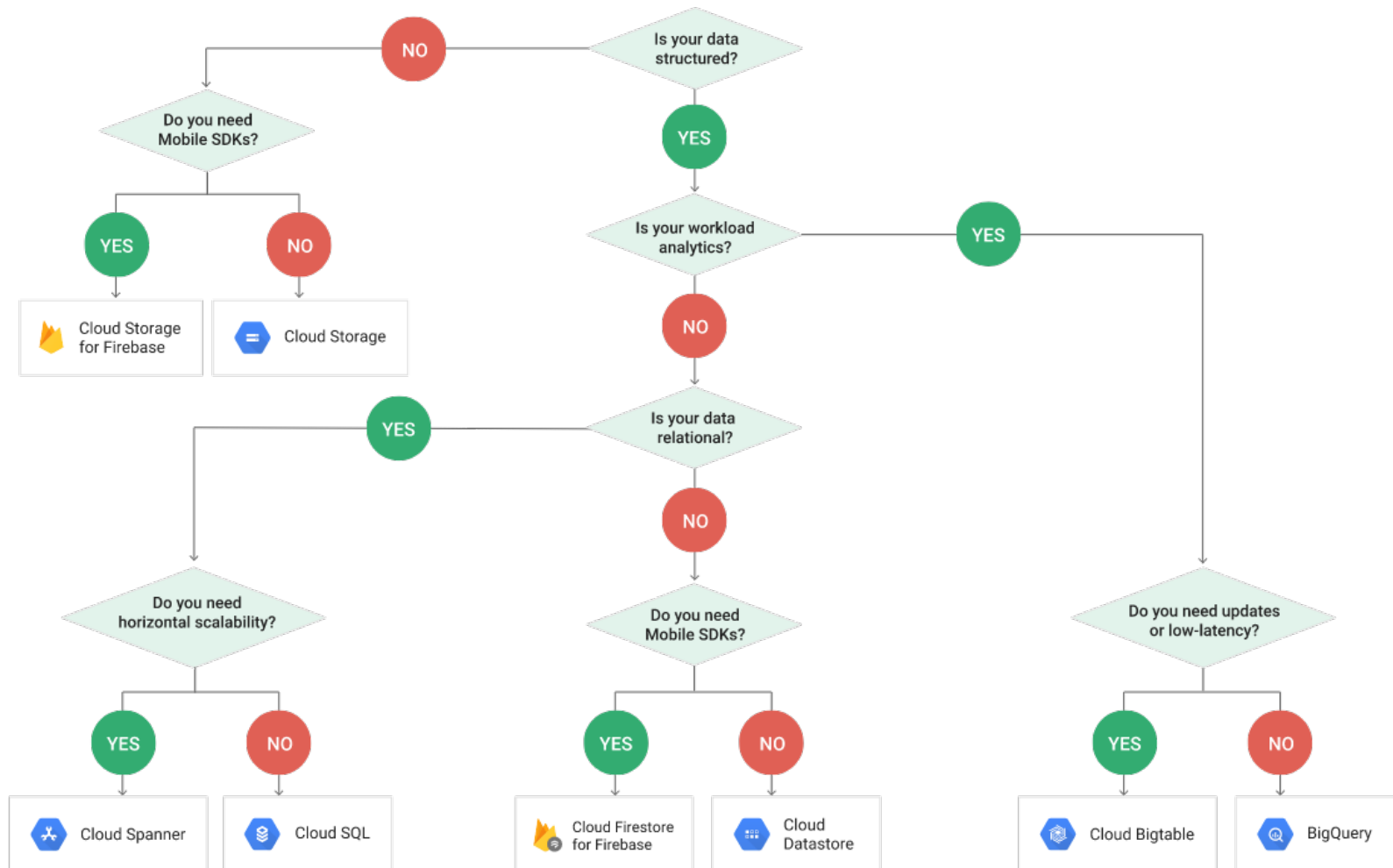
- Cloud Bigtable

- BD en formato columnar.
- Escalabilidad horizontal.
- Apropriada para entornos de gran escala / Big Data.





# Almacenamiento en Google Cloud



# Ejercicio 4

- Crear una instancia de Cloud SQL con una BD "BDEj4" y una tabla "Ciudades":

Ciudad	Habitantes
Madrid	3.280.782
Barcelona	1.636.193

- Crear una app con Cloud Functions que devuelva la información de la tabla "Ciudades"
  - Referencia: <https://cloud.google.com/sql/docs/mysql/connect-functions?hl=es-419>



# Bibliografía

- Dan Sullivan. “Google Cloud Associate Cloud Engineer: Get Certified”, Udemy, 2022.
  - <https://www.udemy.com/course/google-certified-associate-cloud-engineer-2019-prep-course/>
- Google Cloud Docs, Serverless Computing, 2022.
  - <https://cloud.google.com/serverless?hl=es-419>
- Consultados entre julio y diciembre 2022.

