Grado de Ingeniería Informática

Trabajo Final de Grado

Orquestación de Contenedores en un entorno Linux con GKE, Anthos y Google Cloud

Alumno: Víctor Buján Fernández [1]

Consultor: Joaquín López Sánchez-Montañés

Palabras Clave

Contenedores, Linux, Docker, Google Cloud, Kubernetes, Pods, Malla de Servicios, Anthos, GKE, Nube, Seguridad, Escalabilidad.

Key Words

Containers, Linux, Docker, Google Cloud, Kubernetes, Pods, Service Mesh, Anthos, GKE, Cloud, Security, Scalability.

---

1. Ingeniero en Telecomunicaciones, especialidad en telemática, Máster en Software Libre, Máster en Seguridad de las TIC y Postgrado de Cloud Computing, todos en la UOC. Actualmente cursando y finalizando, en la misma universidad, el grado de Ingeniería Informática.

Contexto y justificación del proyecto.

La tecnología de la información ha experimentado una notable evolución en la implementación y gestión de aplicaciones y servicios. En sus primeras etapas, el alojamiento de servicios o la gestión de aplicaciones se basaba en la utilización de servidores físicos, donde cada aplicación era ejecutada en una máquina dedicada resultando costoso y complicado el mantenimiento de la infraestructura y la administración de las aplicaciones.

Sin embargo, surgió la virtualización de los servidores permitiendo la creación de múltiples máquinas virtuales en un solo servidor físico. Permitía trabajar de forma aislada corriendo servicios distintos en un mismo *hardware*. La virtualización redujo los costes de infraestructura, simplificó la gestión de las aplicaciones y permitió una asignación eficiente de recursos.

En los últimos años se ha padecido un cambio aún más disruptivo: la adopción de los contenedores. Los contenedores encapsulan aplicaciones y sus dependencias en entornos aislados y portátiles. Además, comparten y usan el sistema operativo subyacente del host sin necesidad de virtualizarlo y así ser más eficientes en términos de recursos y despliegue.

Esta evolución ha permitido que el despliegue y gestión de las aplicaciones se haga de una forma más eficiente en cuanto a tiempo y recursos. Este proyecto tiene como objetivo explicar cómo funciona esta tecnología centrándonos en la implementación de clústeres de contenedores Kubernetes en un entorno basado en sistemas operativos Linux, utilizando Google Cloud y Anthos para simplificar el despliegue y gestión de aplicaciones.

Context and justification of the project.

Information technology has undergone a remarkable evolution in the implementation and management of applications and services. In its early stages, service hosting or application management was based on the use of physical servers, where each application was executed on a dedicated machine, resulting in costly and complicated infrastructure maintenance and application management.

However, server virtualization emerged, allowing the creation of multiple virtual machines on a single physical server. It made it possible to work in isolation, running different services on the same hardware. Virtualization reduced infrastructure costs, simplified application management and enabled efficient resource allocation.

The last few years have seen an even more disruptive change: the adoption of containers. Containers encapsulate applications and their dependencies in isolated, portable environments. In addition, they share and use the underlying host operating system without the need to virtualize it, thus being more efficient in terms of resources and deployment.

This evolution has allowed the deployment and management of applications to be done in a more efficient way in terms of time and resources. This project aims to explain how this technology works by focusing on the implementation of Kubernetes container clusters in a Linux OS-based environment, using Google Cloud and Anthos to simplify the deployment and management of applications.

Índex del Trabajo Final de Grado

1. Introducción

1.1 Planificación del proyecto

Semana 1 (09 -15 de octubre)

1. **Introducción:**
    - **09 – 11 de octubre:** Contexto y justificación del proyecto.
    - **12 – 13 de octubre:** Planificación del proyecto.
    - **14 – 15 de octubre:** Estructura del trabajo de final de grado.

Semana 2 y 3 (16 -29 de octubre)

2. **Fundamentos teóricos:**
    - **16 – 18 de octubre:** Sistema operativo Linux.
    - **19 – 20 de octubre:** Kubernetes y su papel en la orquestación.
    - **21 – 22 de octubre:** Google Cloud y la infraestructura en la nube.
    - **23 – 24 de octubre:** Google Kubernetes Engine (GKE).
    - **25 – 27 de octubre:** Anthos y su función en la administración.
    - **28 – 29 de octubre:** Revisión y retoque final del aparado.

Semana 4 (30 de octubre – 5 de noviembre)

3. **Configuración inicial:**
    - **30 octubre – 1 de noviembre:** Creación cuenta en Google Cloud Platform.
    - **02 – 04 de noviembre:** Configuración del proyecto.
    - **05 de noviembre:** Revisión y retoque final del aparado.

Semana 5 y 6 (06 -19 de noviembre)

4. **Implantación del clúster de Kubernetes en GKE:**
    - **06 – 09 de noviembre:** Creación de la VPC en la región de Madrid.
    - **10 – 19 de noviembre:** Creación y configuración de un clúster de GKE.

Semana 7, 8 y 9 (20 de noviembre – 10 de diciembre)

5. **Activación y despliegue de aplicaciones con Anthos:**

- **20 – 22 de noviembre:** Activación de Anthos para el despliegue.
- **23 – 25 de noviembre:** Seguridad del clúster antes del despliegue.
- **26 – 05 de diciembre:** Despliegue de WordPress mediante GKE y Anthos.
- **06 – 09 de diciembre:** Plan de backup (copias de seguridad).
- **10 de diciembre:** Revisión retoque final del apartado.

Semana 10 (11 – 17 de diciembre)

6. **Pruebas y validación:**
    - **11 – 13 de diciembre:** Borrado "accidental" del clúster.
    - **14– 16 de diciembre:** Resultados y análisis de las pruebas.
    - **17 de diciembre:** Revisión retoque final del apartado.

Semana 11 (18 – 24 de diciembre)

7. **Conclusión y futuras direcciones:**
    - **8 – 19 de diciembre:** Resumen de los resultados y logros.
    - **20– 21 de diciembre:** Lecciones aprendidas.
    - **22 – 23 de diciembre:** Posibles direcciones futuras para la investigación.
    - **24 de diciembre:** Revisión retoque final del apartado.

Semana 12 (25 – 31 de diciembre)

8. **Bibliografía.**

Semana 13 (01 – 05 de enero)

9. **Agradecimientos:**
    - **01 – 02 de enero:** Agradecimientos.
    - **03 – 05 de enero:** Revisión retoque final del apartado y del trabajo.

A continuación, se plasman estas semanas en un diagrama de Gantt donde se puede observar en forma de cascada la planificación del proyecto:

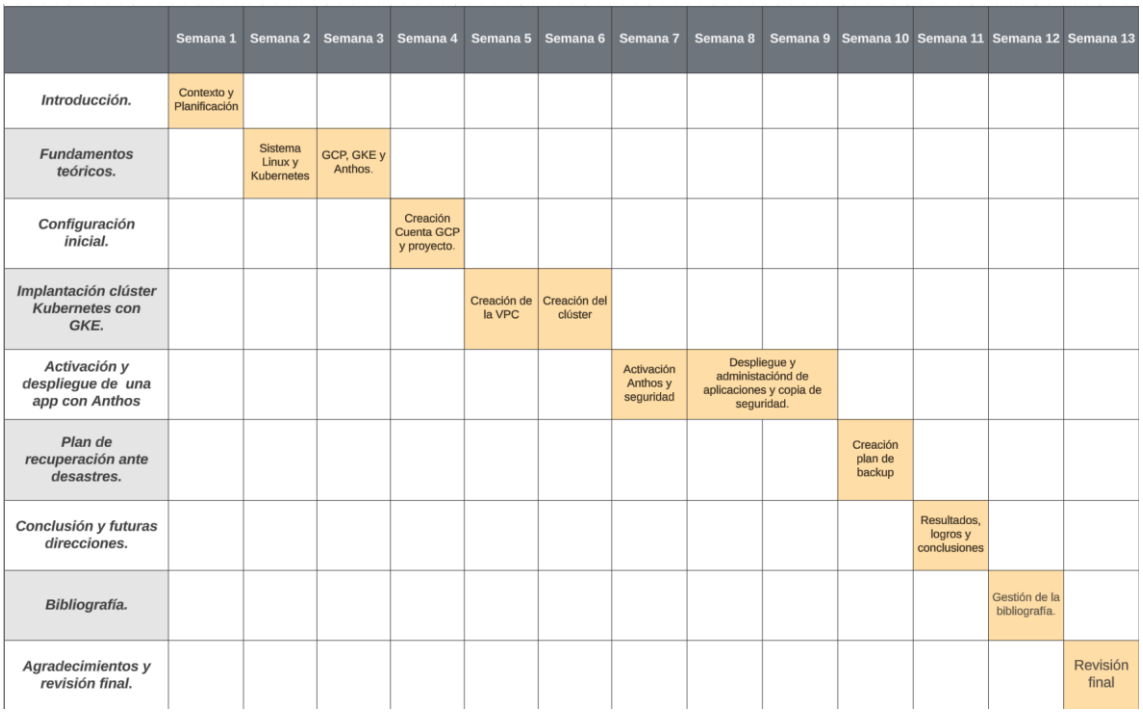

Figura 1. Cronograma del proyecto.

1.2 Estructura del trabajo final de grado.

El presente trabajo de final de grado de Ingeniería Informática sigue una estructura planificada para abordar de manera integral la orquestación de contenedores, la gestión de clústeres de Kubernetes y el despliegue de aplicaciones utilizando Google Kubernetes Engine (GKE) y Anthos en un entorno Linux. La estructura del proyecto está dividida en nueve secciones claves que se desarrollarán a lo largo del documento, permitiendo al lector una comprensión gradual y detallada del proyecto que se aborda.

2. Fundamentos teóricos

2.1 Linux como base para la gestión de contenedores.

2.1.1 Historia de Linux.

Linux comenzó su andadura el 1 de agosto 1991 como un proyecto de un estudiante de informática de la universidad de Helsinki llamado Linus Torvalds. Su objetivo era desarrollar un sistema operativo que tuviera similitudes con Unix y que pudiera ejecutarse en su ordenador. Linux decidió desarrollar el núcleo (*kernel*) y almacenarlo en un servidor FTP donde compartió el código con la comunidad universitaria. Este *kernel* de Linux fue desarrollado como software libre y de código abierto (*open source*) para que cualquier persona con conocimientos en desarrollo lo estudiará, lo modificará e incluso, lo distribuyera.

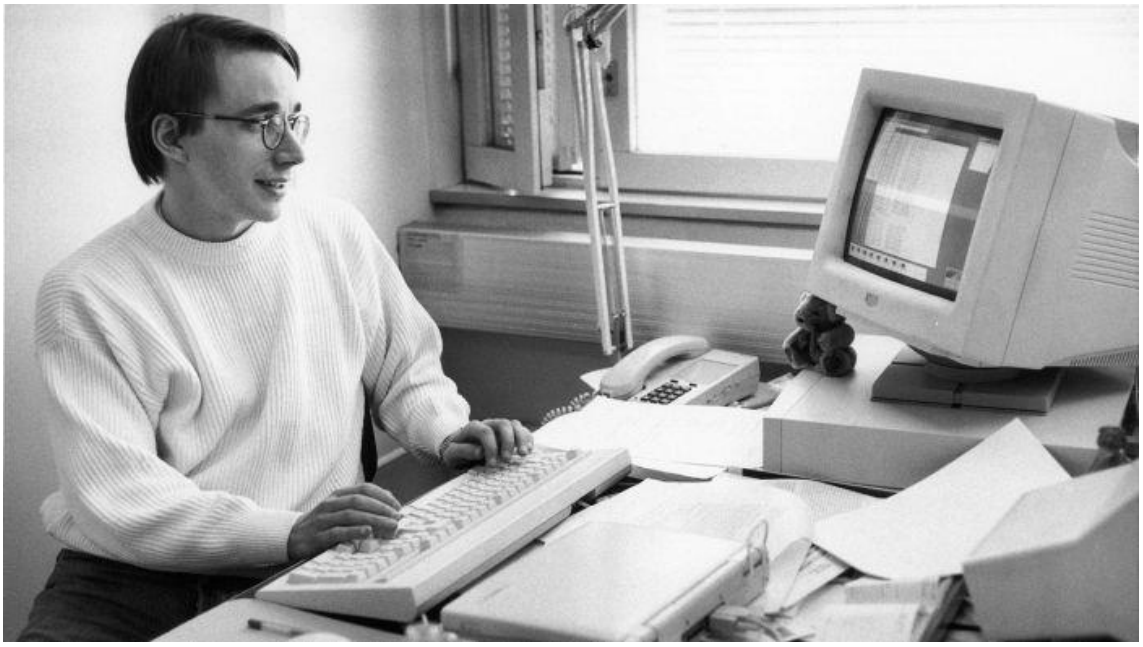

Figura 2. Linus Torvalds en 1993 cuando desarrollo Linux 1.0.

La primera versión de Linux fue la 0.01 y no era ni ejecutable ya que solamente se incluía los principios del *kernel* del sistema que estaba escrito en ensamblador. La segunda versión, la primera oficial y la que sirvió como base, fue la 0.02 lanzada el 5 de octubre de 1991 en la que Linus Torvalds pudo ejecutar Bash y el compilador gcc.

Es cierto que Linus se inspiró en Unix para desarrollar Linux, pero podemos decir que no es ningún clon de ningún sistema Unix específico. En lugar de eso, implementa estándares y convenciones de Unix para garantizar la compatibilidad utilizando un modelo de permisos basado en usuarios y grupos, proporciona una interfaz de línea de comandos (CLI) y emplea una estructura de archivos jerarquía similar a Unix. Con el paso del tiempo Linux ha desarrollado

características y prestaciones únicas con lo que se ha ido diferenciando del sistema operativo Unix tradicional.

Según avanzaba el tiempo, más desarrolladores se unieron al proyecto Linux. Poco a poco se convirtió en un sistema operativo maduro y totalmente funcional. Se fueron creando distribuciones que proporcionaban diferentes opciones en cuanto a programas y utilidades, pero siempre manteniendo el núcleo de Linux, facilitando su adopción en diferentes ámbitos como el residencial o el empresarial. Distribuciones como Debian, Slackware o Red Hat Linux fueron apareciendo y ganando fama en servidores y estaciones de trabajo ya que aportaban estabilidad, seguridad y capacidad de personalización respecto a otras opciones que había en ese momento.

Actualmente Linux es totalmente ubicuo. Podemos tener Linux en servidores web, sistemas embebidos, superordenadores, teléfonos móviles y tables mediante Android y estaciones de trabajo. También está presente y es la base de muchas plataformas de nube como Amazon Web Services (AWS), Microsoft Azure o Google Cloud Platform (GCP), plataforma que utilizaremos en este proyecto de final de grado. La comunidad de desarrollo sigue existiendo y ha crecido de forma exponencial des del 1991, con miles de desarrolladores alrededor del mundo dando soporte, conocimiento y contribuyendo al núcleo y a las aplicaciones.

En el tema que nos compete en este proyecto, Linux está en el corazón de la virtualización y la contenerizaciíon. Tecnologías como Docker o Kubernetes aprovechan su flexibilidad y su seguridad para gestionar aplicaciones en contenedores, facilitando de forma rápida la implementación, el despliegue y la escalabilidad en entornos de nube y en centros de datos.

Figura 3. Linus Torvalds en la actualidad.

2.1.2 Docker y el papel de Linux en la gestión de contenedores.

La relación de Linux con la gestión de contenedores es fundamental para comprender cómo este sistema operativo ha impulsado la revolución de utilizar los contenedores como una tecnología eficiente en el despliegue y la administración de infraestructuras y aplicaciones.

El *kernel* de Linux es la base sobre la cual se ejecutan los contenedores. Antes de seguir, debemos definir que los contenedores son entornos ligeros y aislados que comparten el sistema operativo subyacente a diferencia de las máquinas virtuales, que deben tener cada una de ellas, su sistema operativo. Esta diferencia hace que Linux sea una elección prioritaria y casi natural a la hora de utilizar contenedores por su capacidad de proporcionar aislamiento de recursos y procesos. El núcleo de Linux ofrece características y prestaciones como, por citar algunas, el control de recursos (cgroups) y el aislamiento de espacios de nombres (namesspaces) que son vitales para una gestión eficiente de contenedores.

Docker, una de las tecnologías de contenedores más conocidas, se basa en Linux utilizando el *kernel* para crear, ejecutar y administrar contenedores. Docker simplificó el proceso de empaquetar aplicaciones y sus dependencias/librerías en contenedores, facilitando su portabilidad y despliegue en múltiples entornos.

Figura 4. Logo de Docker

Docker ha utilizado diferentes tecnologías a lo largo de su historia. Inicialmente se utilizó LXC (*Linux Containers*) como tecnología para la creación y gestión de los contenedores. LXC se basa en las características del *kernel* de Linux para proporcionar aislamiento de recursos. Sin embargo, Docker según fue madurando creó una tecnología propia llamada *libcontainer* que está escrita en Go y que le permitía un control más directo sobre los contenedores y así eliminar la dependencia que le infligía LXC

Desde la versión de Docker 0.9 en adelante, ya podemos encontrar libcontainer como el motor de contenedor predeterminado. Libcontainer proporciona una interfaz de bajo nivel para la creación, manejo y control de contenedores en sistemas Linux. Esta biblioteca permite interactuar directamente con las características de aislamiento de núcleo de Linux.

Como se ha comentado antes, libcontainer está desarrollado en Go que es un lenguaje de programación desarrollado por Google y que se ha vuelto muy popular en el mundo del desarrollo de software ya que proporciona eficiencia, simplicidad y facilidad de uso.

2.2 Kubernetes y su papel en la orquestación de contenedores.

Kubernetes es una plataforma de código abierto que tiene como objetivo la orquestación de contenedores. Definimos orquestar contenedores a la gestión y coordinación de múltiples contenedores de software en un entorno para garantizar que las aplicaciones se ejecutan de manera eficiente y confiable. Esto incluye la programación de contenedores en nodos, la gestión de recursos del sistema, la detección y recuperación de fallos, el balanceo de carga y otras tareas que implica el despliegue de aplicaciones.

Kubernetes se ha convertido en una de las herramientas más populares en el mundo de la contenerización debido a su capacidad de simplificar y optimizar la gestión de aplicaciones distribuidas mediante contenedores en entornos de producción.

Figura 5. Logo de Kubernetes

2.2.1 Historia de Kubernetes.

Kubernetes nació en el año 2014 de la mano de Google para abordar el desafío de gestionar de forma más eficiente, escalable, confiable y a gran escala contenedores en entornos productivos.

Esta tecnología se basa en código abierto por lo que es accesible a cualquier persona y/o empresa ganando popularidad en poco tiempo y atrayendo a la Cloud Native Foundation (CNCF), organización sin ánimo de lucro que promueve tecnologías nativas en la nube.

Figura 6. Logo de Cloud Native Cpomputing Foundation

El hito más importante fue en julio del 2015, Kubernetes estaba listo para su uso en entornos de producción con la versión 1.0. Se formó, con el tiempo, una comunidad activa de desarrolladores, empresas y usuarios que contribuían al proyecto ayudándolo a madurar como fue el caso de Linux comentado anteriormente. Se introdujeron nuevas características y mejoras en cada versión hasta que llegó a pasar a ser la plataforma líder en orquestación de contenedores de la industria informática.

2.2.2 Prestaciones y características de Kubernetes.

Kubernetes es una plataforma en constante evolución y que ofrece una serie de prestaciones y características que hacen que sea una herramienta a tener en cuenta en el despliegue y gestión de aplicaciones en entornos contenedor.

Las prestaciones principales son las siguientes:

1. *Escalabilidad automática*.
   Proporciona la capacidad de escalar automáticamente la cantidad de unidades/réplicas de un contenedor en función de la demanda del tráfico o de los recursos disponibles.

2. *Servicio de descubrimiento y balanceo de carga*.
   Proporciona un sistema que permite a las aplicaciones de un mismo entorno descubrirse y comunicarse entre ella de forma automática. También ofrece balanceo de carga para distribuir el tráfico entre contenedores de una misma aplicación.

3. *Alta disponibilidad*.
   Ofrece herramientas para replicar una misma aplicación en diferentes o múltiples nodos y así poder dotar de alta disponibilidad y redundancia evitando así un único punto de fallo.

4. *Actualizaciones automáticas*.
   Facilita un servicio de actualización automática que actualiza las aplicaciones de forma controlada permitiendo volver a una versión anterior en caso de problemas.

5. *Despliegue de aplicaciones en etapas*.
   Permite realizar despliegue de aplicaciones en etapas facilitando la implementación gradual de nuevas versiones.

6. *Seguridad*.
   Incluye características de seguridad como las políticas de acceso basadas en roles (RBAC) o autenticación y autorización.

7. *Monitorización y registro*.
   Permite la integración de herramientas de monitorización y registro para obtener métricas de rendimiento y salud de las aplicaciones en general y contenedores en particular.

8. *Utilización de plugins*.
   Kubernetes es altamente extensible y permite la utilización de plugins que extienden las funcionalidades cubriendo las necesidades de las aplicaciones.

9. *Comunidad activa*.
   Cuenta con una gran comunidad de usuarios y desarrolladores por lo que hay disponible para todos los miembros documentación, tutoriales y herramientas de terceros.

2.3 Google Cloud como proveedor de infraestructura en la nube.

Google Cloud Platform es uno de los principales proveedores de infraestructura en la nube (IaaS) del mundo. Rivaliza con Microsoft Azure, Amazon Web Services y Oracle Cloud a nivel mundial y con los proveedores locales en cada país. Ofrece una amplia gama de servicios de computación.

Figura 7. Logo de Google Cloud Platform

2.3.1 Historia de Google Cloud Platform.

Las andanzas de Google Cloud Platform (GCP) empiezan en el año 2008 con el lanzamiento de Google App Engine, un servicio en la nube que daba la posibilidad a los desarrolladores de alojar aplicaciones web en una plataforma gestionada. Seguidamente, en el 2010, Google consolidó los servicios en la nube bajo la marca de Google Cloud Platform (GCP). Este movimiento no solo incluyó a App Engine sino que también a servicios de infraestructura en la nube como Google Compute Engine o Google Cloud Storage. En el 2013, se incluyó en Google compute Engine un servicio de infraestructura de máquinas virtuales o añadió BigQuery, un servicio de análisis de grandes almacenes de datos.

Fue en el 2015, cuando Google hizo un movimiento importante al lanzar Kubernetes. Este sistema de orquestación de contenedores de código abierto se convirtió en parte esencial del ecosistema GCP para ser donado más adelante a la Cloud Native Computing Foundation (CNCF).

A lo largo de los años, Google Cloud expandió su presencia global con la apertura de nuevos centros de datos y regiones, permitiendo a los usuarios y empresas ejecutar aplicaciones alrededor del mundo.

En junio del 2020, en plena pandemia del COVID-19, Telefónica de España y Google Cloud firman un acuerdo estratégico donde el primero, mediante su data center de Madrid y catalogado como TIER IV alojará la región española de Google Cloud además de desarrollar soluciones basadas en cobertura 5G y utilizando la plataforma Mobile Edge Computing.

Actualmente Google Cloud Platform sigue evolucionando y creciendo, con un claro enfoque a la sostenibilidad, la seguridad y la analítica y explotación avanzada de datos. Se ha convertido en un proveedor de nube esencial a nivel mundial y continúa siendo un competidor importante a Amazon Web Services (AWS) y Microsoft Azure.

2.3.2 Servicios más destacados de Google Cloud

Google Cloud dispone de infinidad de servicios para el usuario final, pero en este apartado se quiere remarcar los más importantes:

- *Google Compute Engine* (GCE)
  Es el servicio de cómputo en la nube que permite crear y administrar máquinas virtuales en modalidad infraestructura como servicio (IaaS). Se ofrece un gran abanico de máquinas virtuales, desde instancias para propósitos generales a instancias optimizadas para cómputo, gráficos o almacenamiento. Además, GCE ofrece opciones de escalabilidad automática para gestionar picos de carga.

- *Google Cloud Storage*
  Es el servicio de almacenamiento de objetos escalable que permite almacenar y recuperar datos en la nube. Ofrece tres clases de almacenamiento, lo que permite que el usuario elegir la opción que mejor se adapta a sus necesidades en cuanto a rendimiento y coste. Además, ofrece capacidades avanzadas de administración de datos como control de versiones, cifrado y acceso controlado por políticas o roles (RBAC).

- *Google Kubernetes Engine (GKE)*
  Es un servicio de orquestación de contenedores basado en Kubernetes y que se detallará más adelante en un capítulo específico. Permite implementar, administrar y escalar aplicaciones en contenedores de forma eficiente y segura. También ofrece prestaciones como el autoescalado, la administración automatizada de clústeres y la integración con herramientas de desarrollo y DevOps.

- *Google App Engine*
  Es una plataforma totalmente gestionada que permite desarrollar y desplegar aplicaciones sin preocuparse por la infraestructura subyacente. Es recomendable para aplicaciones web y móviles ya que se encarga de la administración de recursos, el escalado automático y las actualizaciones de la plataforma, por lo que el desarrollador solo debe centrarse en escribir código.

- *Google Cloud SQL*
  Es un servicio de base de datos relacionales gestionada que es compatible con MySQL y PostgreSQL. Ofrece alta disponibilidad, copias de seguridad automáticas y escalabilidad vertical para que el usuario pueda gestionar sus bases de datos de manera eficiente y confiable.

- *Google Cloud CDN*
  Es un servicio que proporciona una red de entrega de contenidos global para acelerar la entrega del contenido web y aplicaciones a los usuarios finales de todo el mundo. Utiliza infraestructura global de Google para llevar el contenido de forma más próxima a los usuarios y así reducir la latencia al acceso de la información.

2.4 Google Kubernetes Engine como plataforma de Kubernetes.

Google Kubernetes Engine, abreviado como GKE, es un servicio de gestión de contenedores en la nube que ofrece Google Cloud. Se basa en Kubernetes, una plataforma de código abierto para orquestación de contenedores ampliamente adoptada. GKE se ha convertido en una de las soluciones más importantes para desplegar y administrar aplicaciones basadas en contenedores debido a su integración nativa a Googlee Cloud y sus prestaciones avanzadas respecto a otras soluciones.

Analizaremos sus beneficios y arquitectura para comprender como GKE puede potenciar la eficiencia y la agilidad de levantar servicios mediante contenedores.

2.4.1 Prestaciones y beneficios de utilizar GKE.

A continuación, se enumeran algunas de las prestaciones y beneficios clave de utilizar Google Kubernetes Engine:

- *Escalabilidad automática.*
  GKE ofrece una escalabilidad automática tanto en horizontal (más contenedores) como en vertical (más recursos) para las aplicaciones en contenedores. Se pueden configurar reglas de escalado automático basadas en la carga de cómputo y el tráfico garantizando que las aplicaciones se mantengan disponibles en todo momento.

- *Gestión de contenedores simplificada.*
  GKE se encarga de la gestión y el despliegue de los contenedores de forma automática, reduciendo la carga operativa de las tareas derivadas.

- *Alta disponibilidad.*
  GKE está diseñado para ofrecer alta disponibilidad y resiliencia. Las aplicaciones se ejecutan en clústeres altamente disponibles distribuidos en múltiples zonas de disponibilidad lo que garantiza la redundancia y la continuidad del servicio en caso de fallas.

- *Actualizaciones y parches automáticos.*
  GKE realiza actualizaciones y parches de seguridad de forma automática, lo que permite mantener las aplicaciones y clústeres protegidos.

- *Integración con servicios de Google Cloud.*
  GKE se integra con otros servicios de GCP como Cloud Monitoring, Cloud Logging o Cloud IAM lo que facilita la administración y el monitoreo de los contenedores y las aplicaciones.

- *Seguridad Avanzada*
  Ofrece características como aislamiento de redes, control de acceso basado en roles (RBAC), imágenes de contenedores firmadas o escaneo de vulnerabilidades.

- *Soporte y servicio de calidad.*
  Google ofrece un alto de soporte técnico y SLA (acuerdo de nivel de servicio) para GKE, lo que garantiza la disponibilidad y la resolución rápida de problemas críticos.

- *Costes flexibles.*
  GKE y Google Cloud ofrecen precios flexibles, incluyendo precios por uso y descuentos por compromiso a largo plazo, normalmente de 1 a 3 años, lo que permite ahorrar respecto a la contratación PAYG (*pay as you go*, pago por uso).

2.4.2 Arquitectura de un clúster de contenedores en GKE.

La arquitectura de GKE se compone de varios componentes clave que trabajan de forma conjunta para proporcionar un entorno de Kubernetes de alta disponibilidad y confiable. A continuación, se puede observar un diagrama de la arquitectura de un clúster de GKE:

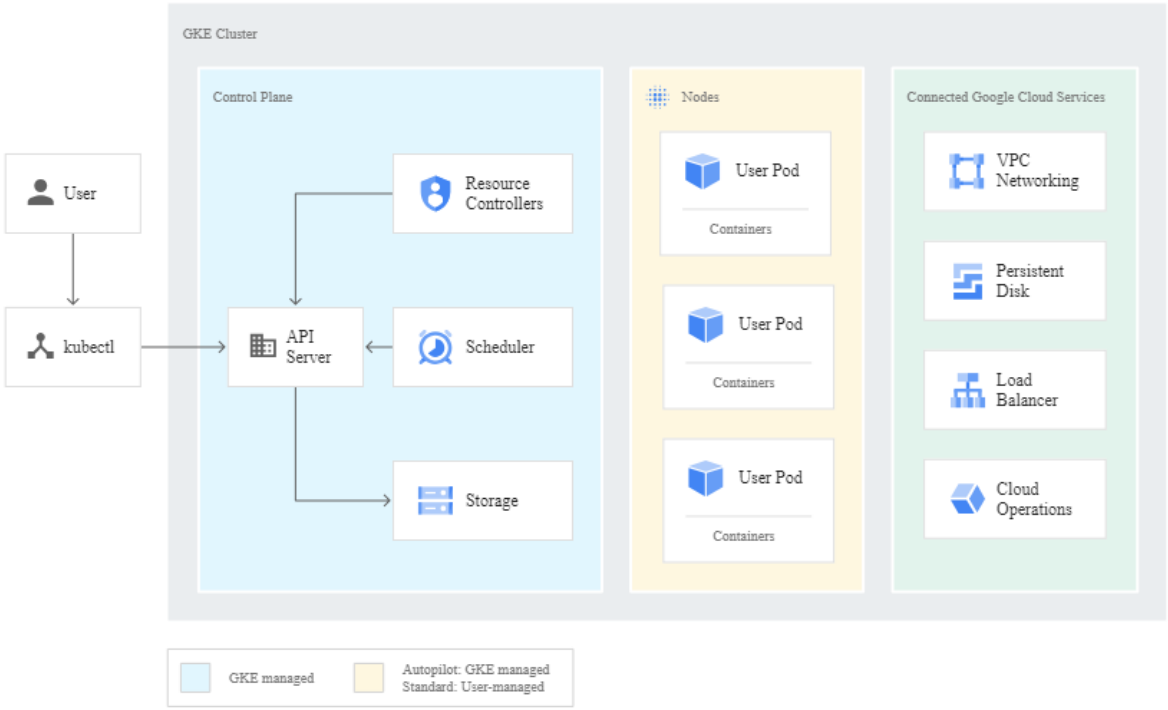

Figura 8. Diagrama de la arquitectura de GKE.

En el diagrama se puede observar los diferentes componentes los cuales se va a proceder a la realización de una pequeña explicación para que se entienda su funcionalidad:

- *Máster Clúster*
  Es la parte central del sistema y contiene componentes maestros de Kubernetes. Estos componentes son:

  - *API Server*
    Es el punto de entrada para las operaciones en el clúster. Los usuarios y aplicaciones interactúan con el clister a través de la API Server, que expone la API de Kubernetes.

  - *Control Plane*
    Es el responsable de tomar decisiones sobre el estado del clúster y garantizar que se cumpla. Esto incluye la gestión de los nodos, las aplicaciones y su escalabilidad.

  - *Etcd*
    Es una base de datos distribuida que almacena la configuración y el estado del clúster. Etcd garantiza que la configuración y el estado del clúster se mantengan consistentes.

- *Node Pools*
  Son grupos de nodos de cómputo configurados para satisfacer las necesidades específicas de las aplicaciones en el clúster.

- *Nodos*
  Son instancias de máquinas virtuales de Google Cloud que forman la base del clúster. Cada nodo ejecuta un sistema operativo Linux y tiene un software necesario para ejecutar los contenedores. Estos se ejecutan en estos nodos y comparten recurso como CPU, memoria y almacenamiento.

- *Pods*
  Son la unidad más pequeña de implementación en Kubernetes y representan un entorno en el que se ejecutan uno o varios contenedores. Los pods son escalables y pueden moverse entre nodos, lo que facilita la administración de aplicaciones.

- *Kubelet*
  Es un agente que se ejecuta en cada nodo del clúster con el objetivo de comunicarse con el maestro del Máster Clúster y garantizar que los pods se ejecutan correctamente en el nodo. El Kubelet supervisa la salud de los pods y reporta su estado al Máster Clúster.

- *Servicios de Google Cloud*
  GKE se integra con otros servicios de Google Cloud como Google Cloud Storage para almacenamiento, Google Cloud Load Balancing para equilibrio de carga y Google Cloud Identity and Acess Management (IAM) para la gestión de identidades. Esto proporciona a los clústeres de GKE acceso a los servicios adicionales de Google Cloud que pueden ser utilizados por las aplicaciones en contendores.

- *Red de Google Cloud*
  La red de Google Cloud interconecta los nodos del clúster y permite la comunicación entre los pods. Google Cloud ofrece una red avanzada que garantiza el aislamiento y la seguridad del tráfico entre pods y la conectividad con servicios externos.

- *Herramientas de Monitoreo y Diagnóstico.*
  GKE ofrece integración con herramientas de monitoreo y diagnóstico de Google Cloud como Google Cloud Monitoring y Google Cloud Logging. Esto permite a los equipos de operaciones supervisar y depurar las aplicaciones en contenedores y el clúster de Kubernetes para garantizar la disponibilidad y el rendimiento.

- *Respaldo y recuperación:*
  La arquitectura de Google Kubernetes Engine incluye capacidades de respaldo y recuperación para garantizar la continuidad operativa de las aplicaciones en contenedores. GKE permite la creación de instantáneas de clústeres y nodos facilitando la recuperación en caso de fallos o pérdida de datos.

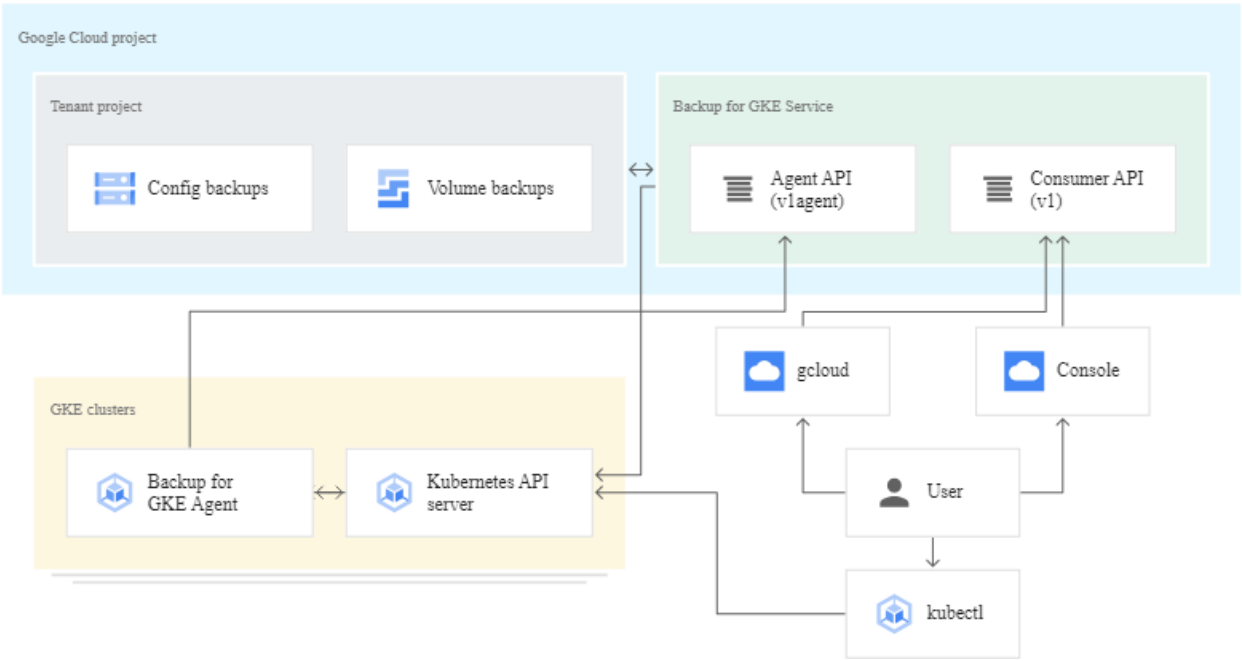

Figura 9. Diagrama del servicio de backup para GKE.

2.5 Anthos y la administración de aplicaciones multinube y locales.

En este apartado, se explorará la función crucial de Anthos, una solución de Google Cloud, en la administración de aplicaciones que deben operar en entornos multinube y locales. Anthos ofrece la capacidad de implementar y gestionar aplicaciones de manera coherente entre diversas plataformas, incluyendo Google Cloud, otras nubles públicas y entornos locales. Esta flexibilidad brinda la agilidad necesaria para adaptarse a diferentes infraestructuras según las necesidades.

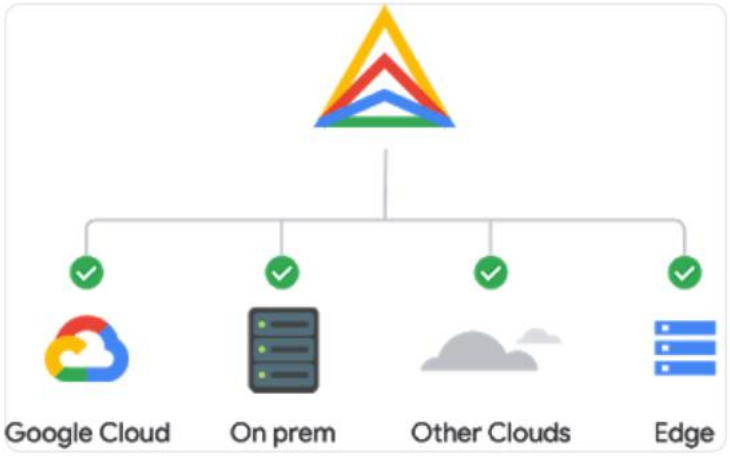

Figura 10. Logo de Anthos y sus entornos.

2.5.1 Historia de Anthos

La historia de Anthos se encuentra ligada a la evolución de la tecnología de contenedores y la visión de Google Cloud de proporcionar una solución integral para la gestión de aplicaciones en entornos multinube y locales.

Anthos nace en 2014 junto con el lanzamiento de Kubernetes por parte de Google. A medida que Kubernetes ganaba popularidad, Google Cloud continuaba innovando en torno a esta tecnología. En 2017, lanzaron proyectos como Istio, una plataforma de malla y Knative, una solución para la construcción y el despliegue de aplicaciones *serverless* en entornos Kubernetes. Estos proyectos fortalecieron la posición de Google Cloud en el mundo de las aplicaciones basadas en contendores y reflejaron su compromiso con la adopción de tecnologías de vanguardia.

En abril del 2019, en la conferencia Google Cloud Next, se anunció oficialmente Anthos (conocido anteriormente como *Cloud Services Plattform*). Anthos se presentó al mundo como una solución revolucionaria que permitía gestionar aplicaciones en contenedores de manera coherente en cualquier entorno.

A medida que pasaba el tiempo, Google expandía la disponibilidad de Anthos a nivel mundial y colaboraba con una variedad de *partners* para ofrecer soluciones integrales de administración de aplicaciones multinube. Anthos se integró con servicios y herramientas de Google Cloud para mejorar su capacidad de administración, seguridad y monitoreo.

2.5.2 Arquitectura de Anthos y sus beneficios.

La arquitectura de Anthos es fundamental para comprender cómo esta plataforma permite la administración de aplicaciones multinube y locales de manera coherente, integrada y eficiente. A continuación, se proporciona una descripción detallada de sus componentes principales:

- *GKE (Google Kubernetes Engine)*
  Anthos aprovecha GKE como base para la administración de clústeres de Kubernetes.

- *Anthos GKE On-Prem*
  Este componente permite la implementación de clústeres de Anthos en centros de datos locales, lo que extiende la capacidad de administración de Anthos a entornos *on premise*. Esto es esencial para aquellas corporaciones que desean mantener cargas de trabajo en su infraestructura local y trabajar en un entorno híbrido

- *Anthos Config Management*
  Es una herramienta que permite definir y aplicar políticas de configuración en todos los clústeres de Anthos, ya sea en Google Cloud o en entornos locales. Esto asegura la consistencia en la configuración de las aplicaciones y la seguridad en todos los clústeres.

- *Anthos Service Mesh (Istio)*
  Es una plataforma que ofrece una capa de servicios de malla para conectar, supervisar y asegurar servicios dentro del clúster. Utiliza Istio como tecnología subyacente para administrar el tráfico de red y proporcionar políticas de seguridad avanzadas.

- *Anthos Service Mesh (Istio)*
  Este componente ofrece funcionalidades de gestión de identidades y autenticación en entornos multinube y locales garantizando un acceso seguro a aplicaciones y servicios.

La arquitectura de red en Anthos es esencial para garantizar la conectividad entre los clústeres de Anthos en diferentes ubicaciones geográficas y entornos de nube. Anthos aprovecha la red de Google Cloud y las redes definidas por software (SDN) para proporcionar conectividad segura entere los clústeres.

Además, Anthos se basa en medidas de seguridad y cumplimiento rigurosas. Ofrece características de seguridad avanzadas, como la autenticación basada en identidades y políticas de acceso. Además, está diseñado para cumplir las regulaciones de seguridad y privacidad, lo que es fundamental para las empresas.

Y, por último, Anthos proporciona herramientas avanzadas de gestión y monitoreo que permiten a los equipos de operaciones supervisar y mantener las aplicaciones en entornos multinube y locales. Esto incluye la capacidad de realizar un seguimiento del rendimiento de las aplicaciones, generar registros y recibir alertas en tiempo real.

3. Configuración inicial.

En este apartado da comienzo la parte práctica del proyecto que consistirá en la implantación de los conceptos explicados anteriormente en un entorno real. A modo resumen, esto incluirá la configuración de un clúster Kubernetes con Google Kubernetes Engine (GKE) que actuará como base para administrar la aplicación que queremos implantar mediante la utilización de contenedores en Google Cloud. De forma paralela, veremos cómo se integra el sistema operativo Linux en este entorno. Por último, Anthos nos dará visibilidad de como administra la aplicación.

3.1 Creación de una cuenta en Google Cloud Platform.

En esta primera fase del proyecto, el primer paso será la creación de una cuenta de Google Cloud Platform (GCP). Esta cuenta proporcionará acceso a las herramientas y servicios de Google Cloud que se precisará para configurar y gestionar el clúster de Kubernetes en Google Kubernetes Engine (GKE) y realizar otras tareas relacionadas con el proyecto.

Vamos a dar de alta una cuenta en Google Cloud Platform, para ello debemos de acceder al sitio web oficial:

https://cloud.google.com/

En la página de inicio de GCP, debemos de dar clic en el botón “Comenzar Gratis”. Esto nos llevará al proceso de registro para crear una cuenta de prueba gratuita con un saldo de 300$.

El alta en Google Cloud Platform se va a proceder mediante la utilización de mi cuenta personal basada en Gmail. Google nos traslada que nos proporcionará 300$ (284€) de saldo iniciales de prueba. Seguimos avanzando con el registro el cual nos pedirá nombre completo, documento nacional de identidad (DNI), dirección, número de teléfono móvil y, por último, el número de la tarjeta de crédito.

Una vez introducido todos los datos y completado el proceso de registro, ya tenemos acceso a Google Cloud Console, que es la interfaz de administración. Ahora ya se puede crear proyectos, configurar recursos y comenzar a utilizar servicios de GCP.

3.2 Configuración del proyecto en Google Cloud Platform.

En este punto se va a configurar un proyecto en GCP. Para ello, creamos uno haciendo clic en el botón “crear o selecciona proyecto”:

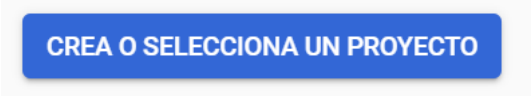

Figura 11. Botón para crear un proyecto en GCP.

A continuación, y al estar creando un nuevo proyecto, GCP nos pedirá que le asignemos un nombre:

Figura 12. Nombre del proyecto en GCP

Una vez creado el proyecto, verificamos que nos aparece en nuestro panel de gestión:

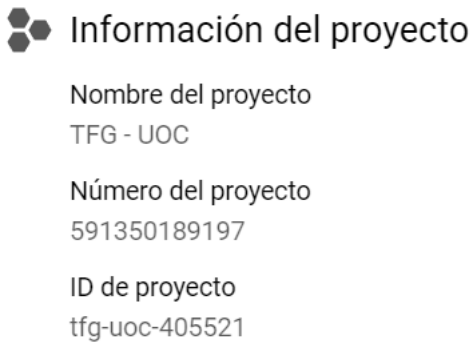

Figura 13. Nombre del proyecto en GCP.

Una vez creado, debemos explorar Google Cloud para familiarizarnos con las herramientas y los recursos disponibles. Ahora ya podemos crear y gestionar instancias de máquinas virtuales, bases de datos y más recursos como un clúster de GKE.

4. Implantación del clúster de Kubernetes en GKE.

Una vez creada la cuenta que nos permitirá operar en Google Cloud, exploraremos en detalle los pasos clave para configurar y desplegar un clúster de Kubernetes en GKE. Veremos como definir la topología de clúster, configurar los aspectos de red, gestionar los nodos de trabajo y como aprovechar las características avanzadas que GKE ofrece para mejorar la escalabilidad y la disponibilidad de las aplicaciones. Además, se discutirá que consideraciones de seguridad se deben de tener en cuenta al llevar a cabo este proceso de despliegue.

4.1 Creación de la VPC en la región de Madrid.

Antes de sumergirnos en la creación de un clúster de Kubernetes, es esencial establecer una base sólida con la creación de una Red de Nube Privada Virtual (VPC) en Google Cloud. En este apartado, nos centraremos en la configuración de una VPC en la región de Madrid, una decisión estratégica que influirá en el rendimiento del clúster al estar yo ubicado en Zaragoza / Lleida y disponer de una baja latencia.

Primero de todo, accedemos a la consola de Google Cloud y vamos a la sección de Red de VPC. Una vez dentro de la sección, se va a proceder a la creación de la red proporcionando un nombre descriptivo:

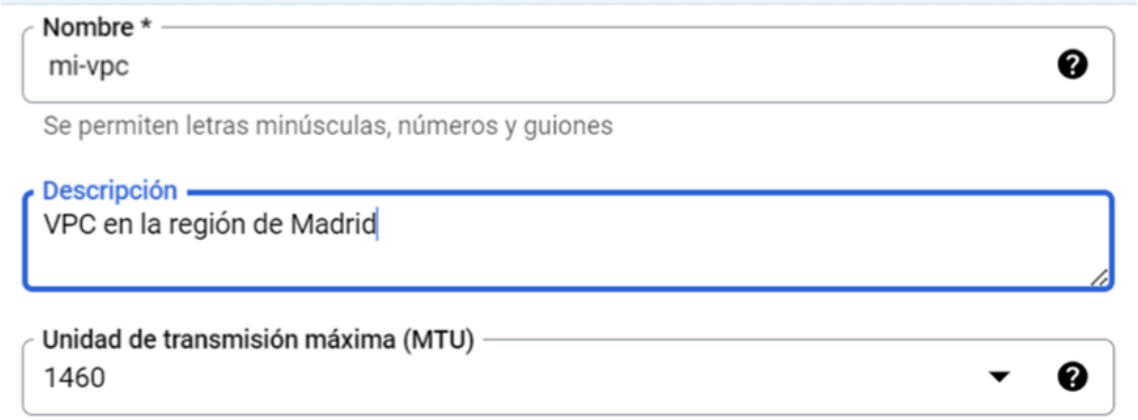

Figura 14. Introducción nombre de la VPC.

En la imagen anterior se muestra el nombramiento de la VPC como "*mi-vpc*", y la descripción de esta la cual definimos que es una VPC en la región de Madrid. En la sección de "*Rango IPv6 interno de ULA de la red de VPC*" está deshabilitada significando que no se utilizará un rango de direcciones IPv6 de la ULA (*Unique Local Adress*) para dispositivos internos dentro de la VPC. Las ULAs son rango de direcciones IP's que se utilizan dentro de una red privada y no son enrutables en Internet.

Por último, el "*Modo de creación de subred*" está configurado como "*Personalizado*", indicando que las subredes no se crearán automáticamente, sino que serán definidas manualmente por el administrador de red.

Por ello, se va a crear una subred con un rango de IP's predeterminado para el uso de esta subred con el clúster Kubernetes que crearemos más adelante:

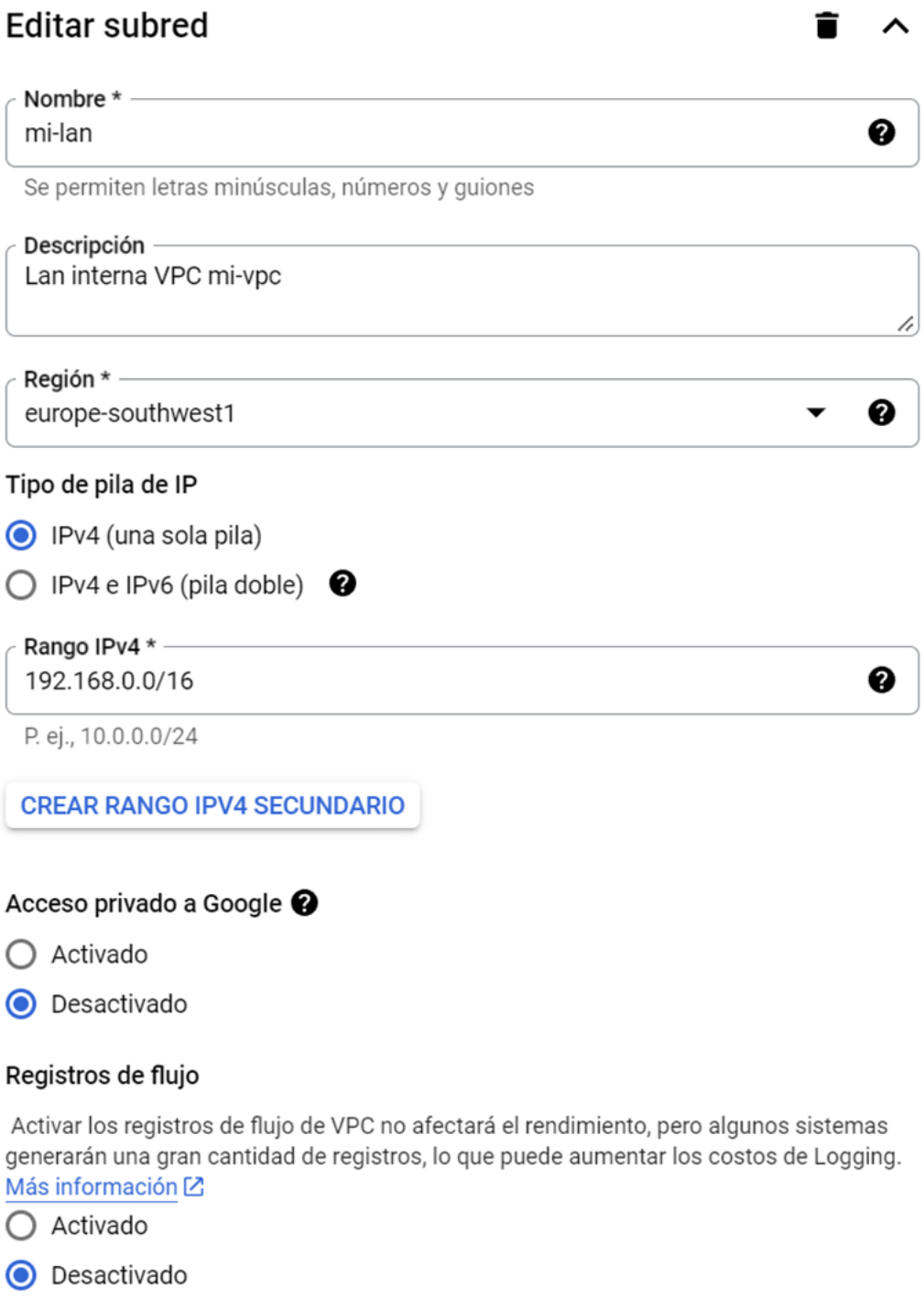

Figura 15. Creación de la subred a utilizar en GKE.

La imagen anterior muestra la configuración de una subred en una VPC en la región "*europe-southwest1*" que es la región de Madrid.

A continuación, un pequeño desglose de los campos mostrados:

- ***Tipo de pila de IP***
  Está seleccionado "*IPv4 (una sola pila)*", ya que la subred solamente usará direcciones IPv4. "*IPv4 e IPv6 (pila doble)*" permitiría el uso de ambos protocolos.

- ***Rango IPv4***
  "*192.168.0.0/16*" define el bloque de direcciones IP que se asignarán a los recursos dentro de esta subred. Este rango es parte del espacio de direcciones IP privadas que no son enrutables en Internet y se utilizan de forma común dentro de las redes privadas.

- ***Acceso privado a Google***
  Si estuviera activado, permitiría a las instancias de la subred acceder a los servicios de Google a través de una dirección IP interna en lugar de a través de Internet.

- ***Registros de flujo***
  Si estuviera activado, se capturaría registros detallados de todo el tráfico de la red, lo cual es útil para el análisis y soluciones de problemas y por temas de seguridad, pero aumenta los costos de logging.

Se continua la configuración con las reglas de firewall como se observa en la imagen siguiente:

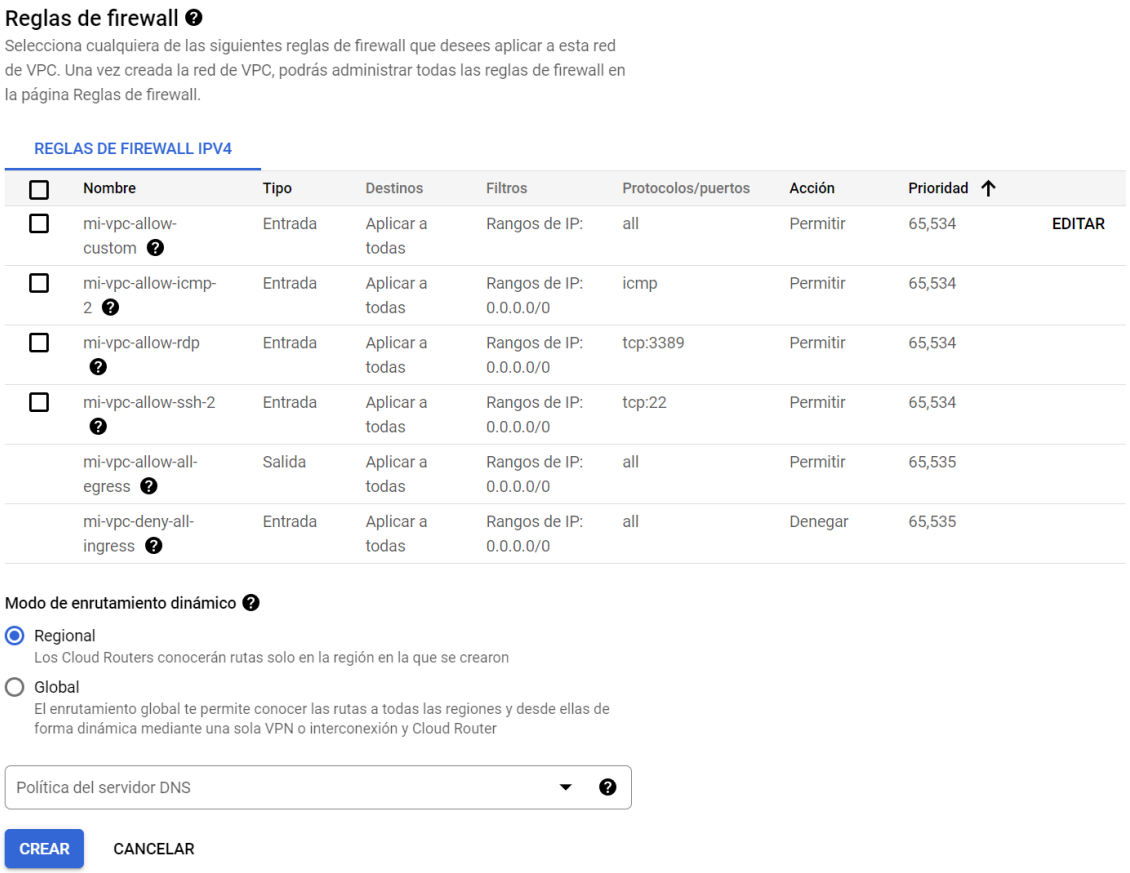

Figura 16. Reglas de Firewall de la VPC

Las reglas determinan el tráfico saliente y/o entrante permitido en la VPC y las que se describen a continuación se crean de forma predeterminada al crear la VPC:

- ***mi-vpc-allow-custom***
  Esta regla está configurada para permitir todo el tráfico entrante a la VPC. No especifica filtros de IP, por lo que significa que acepta conexiones entrantes de cualquier origen (*0.0.0.0*).

- ***mi-vpc-allow-icmp-2***
  Esta regla permite el tráfico entrante del protocolo ICMP que se utiliza para enviar mensajes de control y errores como "*ping*"

- ***mi-vpc-allow-rdp***
  Esta regla permite el tráfico entrante del protocolo RDP (*Remote Desktop Protocol*) en el puerto 3389, utilizado comúnmente en sistemas Windows para la conexión a escritorios remotos. Está regla la eliminaremos ya que no necesitamos el uso del protocolo RDP en la gestión de contenedores GKE por lo que podemos considerar deshabilitar la regla para reducir la superficie de ataque y seguir las practicas recomendadas de seguridad.

- ***mi-vpc-allow-ssh-2***
  Permite el tráfico entrante para SSH (*Secure Shell*) en el puerto 22 para la administración segura de sistemas de Kubernetes o máquinas Linux a través de una red.

En el apartado de seguridad veremos cuales de estas reglas debemos eliminar y que otras debemos de ajustar para pasar el clúster a "*producción*" en un entorno seguro.

Las reglas tienen una columna de "*Prioridad*" que indica la prioridad de procesamiento del tráfico, con números más bajos teniendo mayor prioridad. En este caso, todas las reglas tienen la misma prioridad para permitir que para denegar por lo que se aplicaran en el orden que se listan si las condiciones de múltiples reglas son satisfechas por un paquete de red.

Por último, el "*Modo de enrutamiento dinámico*" está configurado como "*Regional*", lo que significa que los "*Cloud Routers*" sólo conocerán rutas dentro de la región que se crea la VPC. La opción "*Global*" permitirá el enrutamiento a través de otras regiones de Google Cloud.

Una vez configurada la subred, se ha procedido a asignar dos IP's públicas que nos permitirán acceder a los servicios desplegados por el clúster de Kubernetes a través de Internet. Las IP's públicas serán estáticas por lo que siempre serán iguales, aunque apaguemos el servicio:

☐	Nombre	Dirección IP	Tipo de acceso	Región	Tipo ↓	Versión
☐	ipestatica	34.175.154.67	Externo	europe-southwest1	Estática	IPv4
☐	ipestatica1	34.175.143.171	Externo	europe-southwest1	Estática	IPv4

Figura 17. Las IP's estáticas que se utilizarán para acceder desde Internet.

4.2 Creación del clúster en GKE.

Avanzamos hacia la creación de un clúster de máquinas virtuales que permitirá una eficiente gestión de contenedores en Google Cloud con GKE. Este proceso marca el comienzo de la configuración de un entorno robusto y escalable para desplegar aplicaciones sin Autopilot. Abordaremos desde la elección inicial del tipo de clúster hasta las configuraciones específicas de hardware, software y red, asegurando que el clúster esté optimizado para las necesidades particulares del entorno de trabajo el cual queremos utilizar. Este último paso es muy importante para garantizar el correcto funcionamiento de las aplicaciones sino también para sentar las bases de una integración efectiva con otros servicios avanzados como Anthos que explicaremos más adelante.

Por otro lado, se define Autopilot como una opción de configuración de GKE que automatiza la administración de la infraestructura del clúster. Proporciona un entorno de Kubernetes gestionado y optimizado donde Google administra y escala automáticamente los nodos del clúster según las necesidades de carga de trabajo del momento. Autopilot, simplifica la operación y el mantenimiento del clúster, puede limitar las opciones de personalización en comparación con un clúster estándar. Por ello y para la elaboración del trabajo, se decide crear el clúster sin la ayuda de Autopilot para adaptarlo a los requerimientos específicos, como pueden ser configuraciones de hardware específicas y opciones de red avanzadas.

4.2.1 Aspectos básicos de la configuración del clúster.

Para empezar a crear el clúster vamos a su sección en Google Cloud:

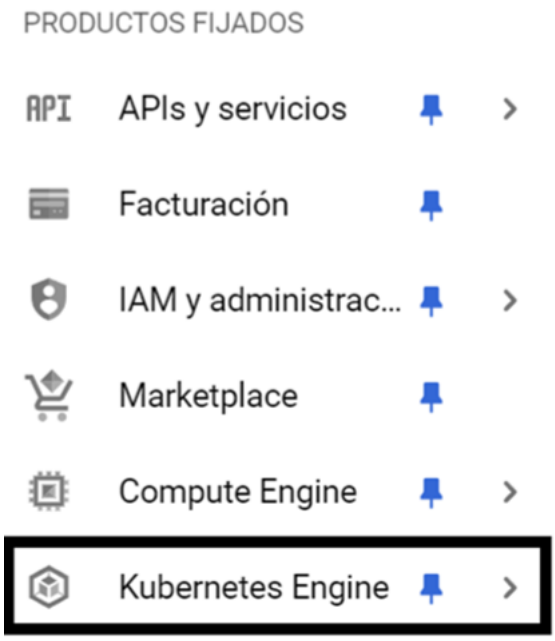

Figura 18. Enlace a la sección de Kubernetes.

Una vez accedido a la sección debemos de empezar el asistente de configuración de un clúster de Kubernetes:

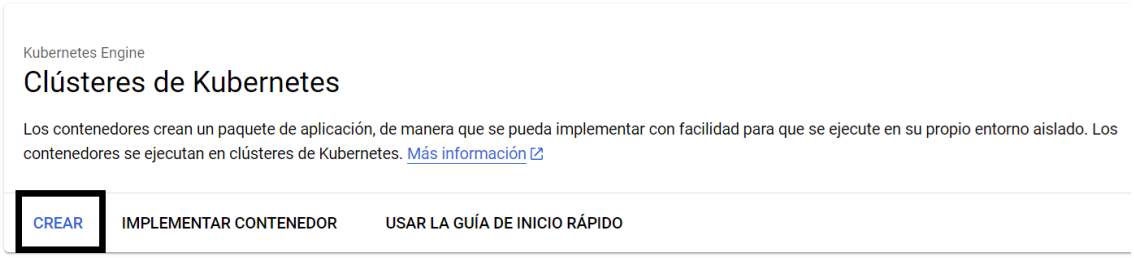

Figura 19. Inicio del asistente de creación de un clúster.

Elegimos la opción de crear un clúster sin la ayuda de Autopilot. Los aspectos básicos que configurar son:

- ***Nombre del clúster.***
  Aquí se especifica el nombre para identificar el clúster y una ubicación que el usuario debe definir. Una vez creado el clúster, el nombre y la ubicación donde está alojado no puede ser cambiado de ninguna forma. Los nombres para identificar los clústeres deben de comenzar en letra minúscula y pueden incluir hasta 39 caracteres.

- ***Tipo de ubicación.***
  Los precios de los recursos utilizados en el clúster pueden variar dependiendo de la región geográfica seleccionada. Hay dos tipos:

  - ***Zonal***: Se refiere que se levanta ese clúster dentro de zonas de disponibilidad diferentes, pero dentro de la misma región. Por ejemplo, *europe-southwest1-a* es una zona dentro de la región *europe-southwest1*.

  - ***Regional***: Abarca varias zonas dentro de una región geográfica más grande, lo que puede aumentar la disponibilidad y resistencia del clúster frente a fallos de una sola zona.

- ***Especificar las ubicaciones predeterminadas en nodos.***
  Esta opción permite seleccionar múltiples zonas dentro de una región para mejorar la disponibilidad del clúster. Si se seleccionan varias zonas, se distribuirá la misma cantidad de nodos en cada zona.

A continuación, podemos observar estas opciones en la siguiente figura:

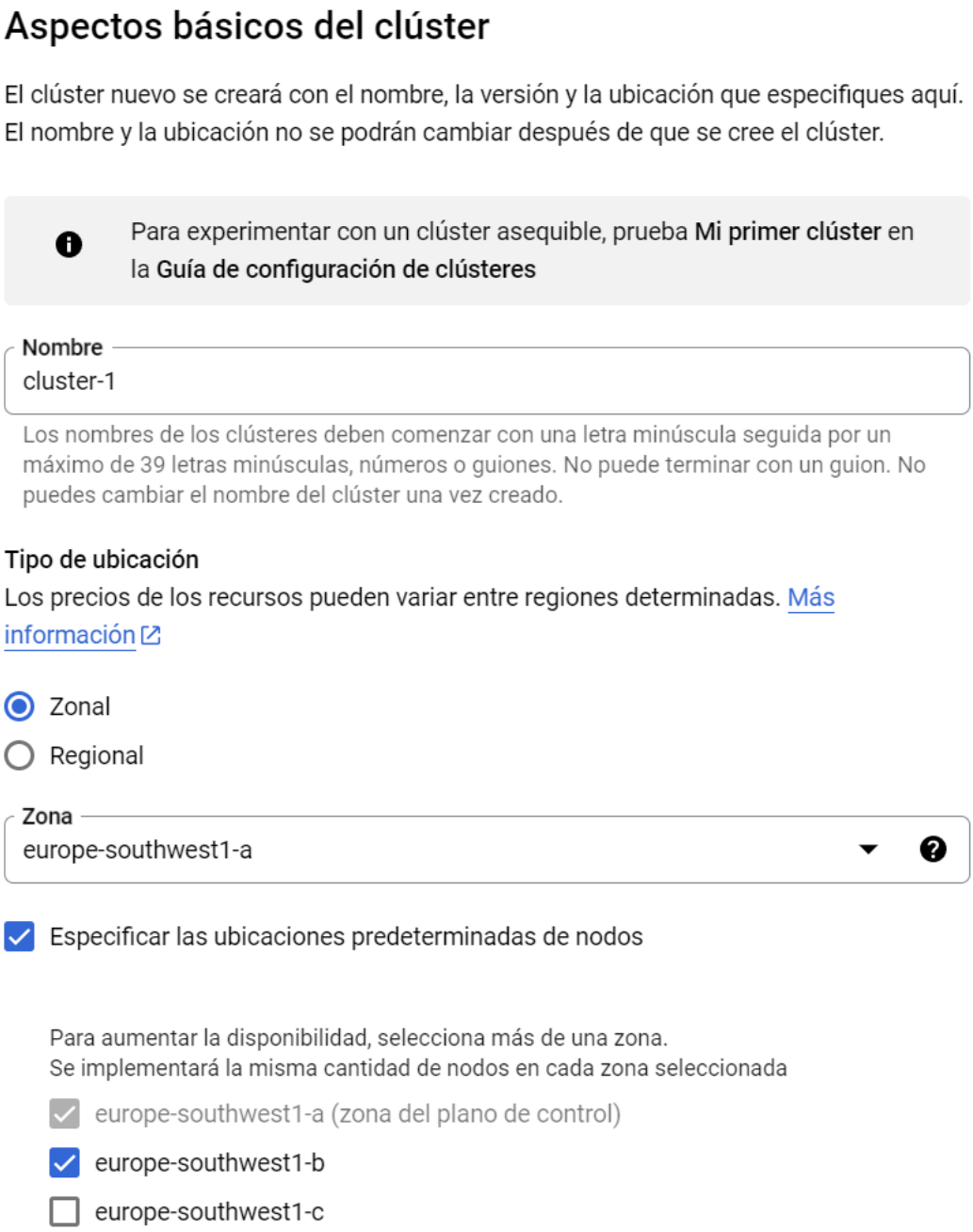

Figura 20. Aspectos básicos de creación de un clúster, parte 1.

Debemos de tener en cuenta que replicar los nodos en dos o tres zonas, es multiplicar el coste del clúster al mes. Actualmente, el clúster formando por tres máquinas virtuales con una serie de recursos por defecto y funcionando 730 horas (que es todo el mes), nos cuesta 195,59€/mes. Si añadimos una segunda zona (*europe-southwest1-b*) el precio asciende a 318.18€/mes, que es un precio inferior al doble. Si sumamos la tercera zona (*europe-southwest1-c*) el coste mensual asciende a 440,76€. Para ofrecer más redundancia y tolerancia a fallos, pero sin excedernos en el coste mensual, el clúster que crearemos lo replicaremos en las zonas a y b de Madrid.

Figura 21. Coste mensual del clúster a crear.

Seguimos con la configuración y ahora toca con la "*Versión del plano de control*" que es crítico para la gestión del estado y la operación del clúster. Esta parte permite al usuario decidir cómo gestionar las actualizaciones del plano de control del clúster:

- ***Versión estática***
  Con esta opción, el usuario administra manualmente las actualizaciones de la versión del plano de control y los nodos. GKE solo actualizará el plano de control y los nodos si es necesario para mantener la seguridad y compatibilidad.

- ***Canal de versiones.***
  Se permite que GKE administre automáticamente la versión del plano del clúster. Esto implica que GKE decidirá cuándo y cómo actualizarlo con el fin de mantener el clúster actualizada con las versiones seguras y compatibles.

Además, dentro del "*Canal de versiones*", se pueden elegir entre diferentes canales:

- ***Canal regular (predeterminado).***
  Es el canal seleccionado por defecto y que probablemente ofrece un mejor equilibrio entre la estabilidad y acceso a las nuevas funcionalidades.

- ***Versión.***
  La versión recomendada para el plano de control del clúster es la "*1.27.3-gke-100 (predeterminada)*". Esta versión ha pasado por la validación interna y se considera que es la mejor para ser utilizada.

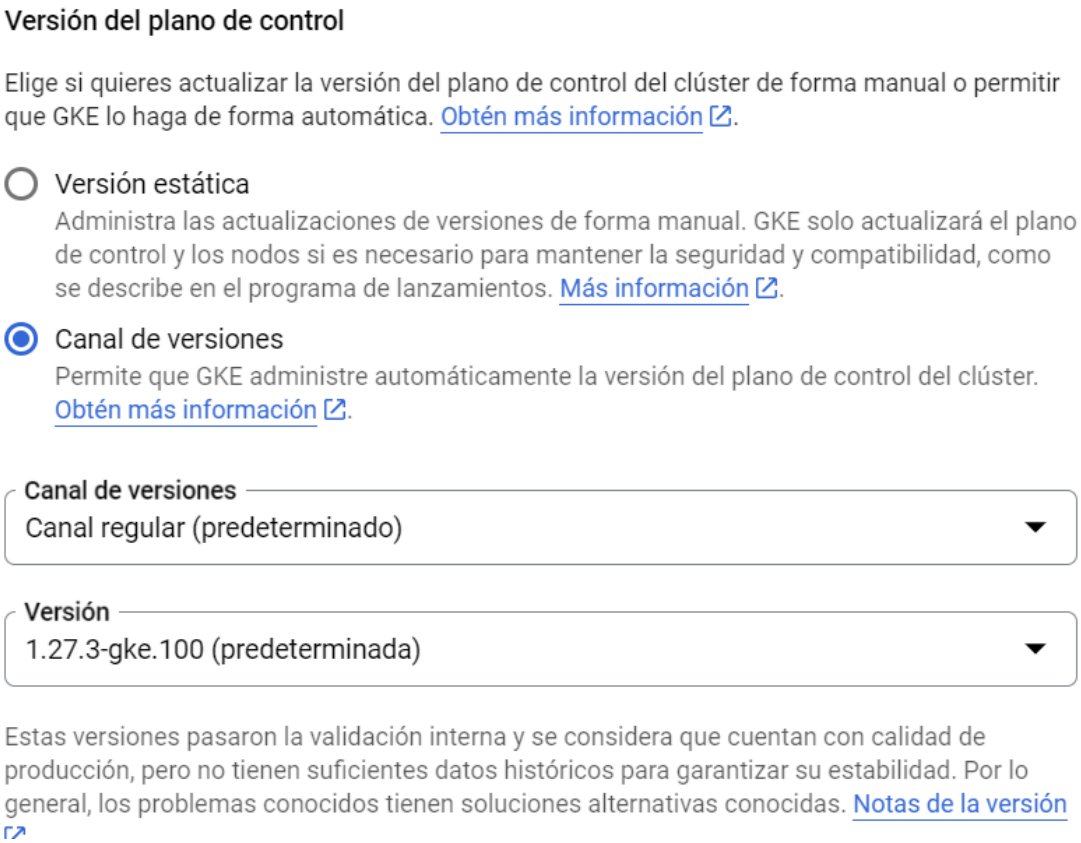

Figura 22. Aspectos básicos de creación de un clúster, parte 2.

Seguimos con la configuración de los nodos que formarán el clúster.

4.2.2 Prestaciones del grupo de nodos.

En este apartado se describe los nodos que forman parte del clúster que estamos creando. En un clúster se crea con al menos un grupo de nodos y se pueden agregar o quitar más grupos, incluso, cuando el clúster esté en funcionamiento.

A continuación, podemos observar una figura con el detalle del grupo de nodos:

Figura 23. Detalle del grupo de nodos, parte 1.

Debemos de elegir un nombre para el grupo de nodos. En nuestro caso es "*nodes-pool*".

Tal como se puede ver, se ha seleccionado que habrá 3 nodos por zona. Como se va a emplear la "*europe-southwest1-a*" y la "*europe-southwest1-b*" la consola nos indica que habrá un máximo de 6 nodos en el clúster.

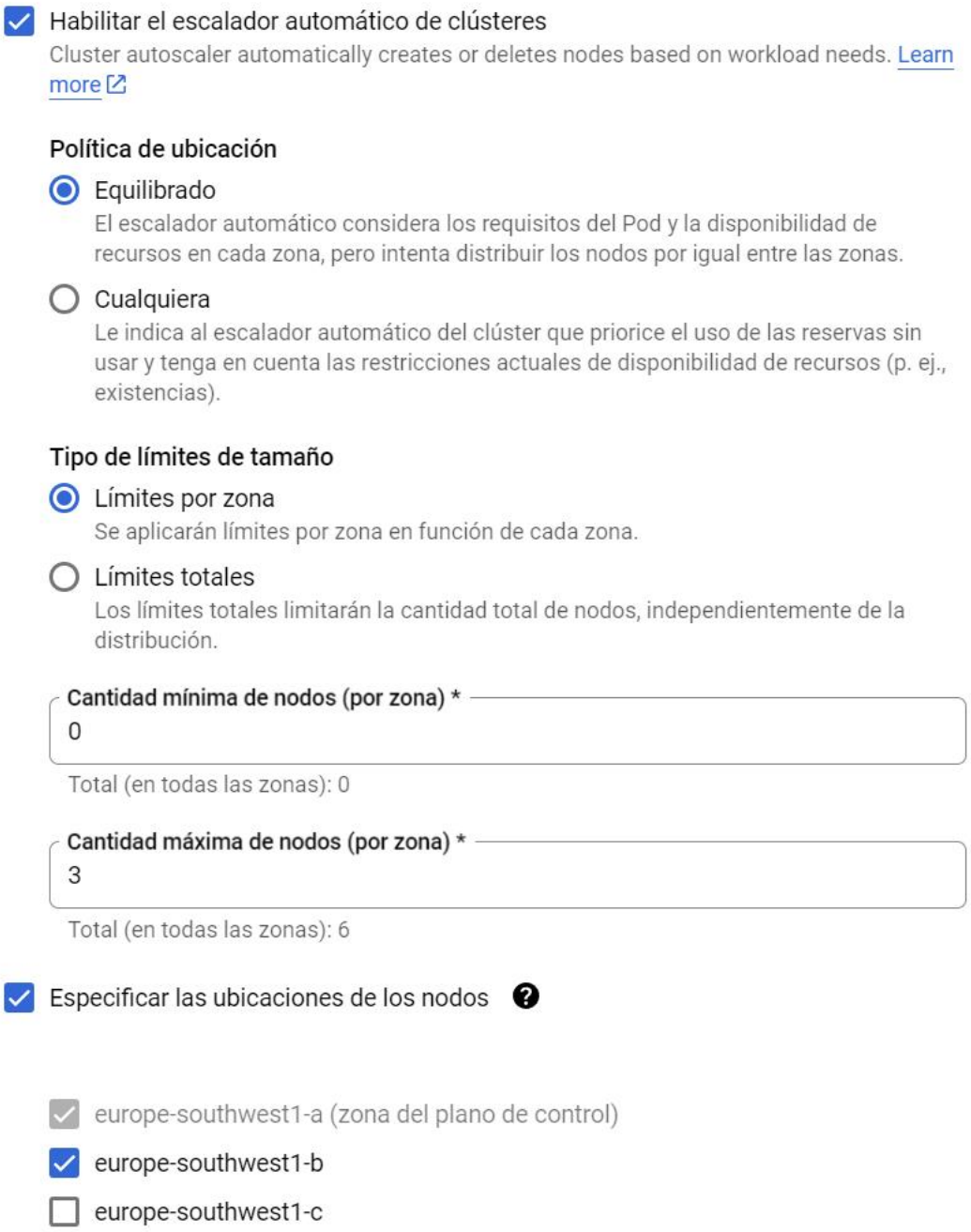

Figura 24. Detalle del grupo de nodos, parte 2.

Marcar la casilla "*Habilitar el escalador automático de clústeres*" establece que el clúster ajuste automáticamente el número de nodos basándose en las necesidades de carga de trabajo.

Respecto a la política de ubicación, tenemos dos opciones:

- ***Equilibrado.***
  Esta política indica que el escalador automático considerará tanto los requisitos del pod (*la unidad más pequeña de despliegue que se puede gestionar en Kubernetes*) como la

disponibilidad de recursos en cada zona, intentando distribuir los nodos de manera equitativa entre las zonas.

- ***Cualquiera.***

  Con esta opción, el escalador automático priorizará el uso de reservas sin usar y tomará en cuenta las restricciones actuales de disponibilidad de recursos como, por ejemplo, las existencias.

Podemos habilitar limites en el tamaño del clúster de la siguiente forma:

- ***Limites por zona.***

  Los límites aplicados serán específicos para cada zona. Esto significa que habrá un número máximo y mínimo de nodos configurados por zona.

- ***Limites totales.***

  Esta opción limitará la cantidad total de nodos en todo el clúster, independientemente de su distribución por zona.

Por último, se establece que el número mínimo de nodos por zona sea 0 y que el máximo sea 3, dando un total de 6 sumando las dos zonas con las que se quiere trabajar ("*europe-southwest1-a*" y "*europe-southwest1-b*"). Se procede la configuración con la automatización:

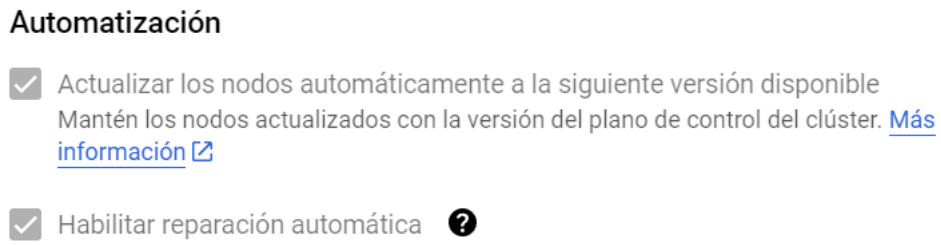

Figura 25. Automatización de los nodos.

Respecto a la primera opción marcada, el sistema asegura que los nodos del clúster se mantengan actualizados con la versión más reciente del plano de control de forma automática. Esto es útil para mantener la seguridad y las funcionalidades del clúster sin necesidad de intervención manual.

La segunda opción habilitada permite que, si un nodo falla o se detecta un problema en él, el sistema intentará repararlo automáticamente o remplazándolo por completo por otro nuevo.

4.2.3 Container-Optimized OS

Las máquinas virtuales llevan un sistema operativo llamado "*Container-Optimized OS con containerd*"de forma predeterminada:

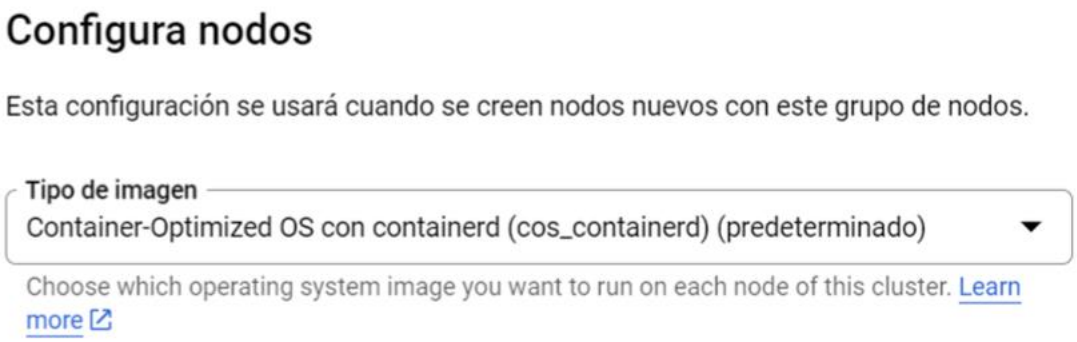

Figura 26. Sistema operativo de las VM's de los nodos.

Container-Optimized OS es un sistema operativo ligero diseñado por Google específicamente para la ejecución de contenedores en Google Cloud incluyendo GKE. Está basado en Chromium OS, por lo que se basa en Linux.

A continuación, a modo resumen, algunas características clave de este sistema operativo:

- ***Seguridad.***
  COS está diseñado con la seguridad como premisa. Al minimizar el sistema operativo base y utilizar características de seguridad como el bloqueo de acceso directo a la raíz y las actualizaciones automáticas, COS se protege de vulnerabilidades.

- ***Inmutabilidad.***
  Gran parte del sistema de archivos es de sólo lectura o montado temporalmente. Esto significa que los cambios no persisten después de un reinicio por lo que ayuda a mantener la integridad y la consistencia de los datos.

- ***Optimizado para contenedores.***
  Al ser un sistema operativo ligero, COS está optimizado para ejecutar contenedores con Docker o Kubernetes, lo que significa que tiene todas las dependencias y herramientas necesarias de forma nativa.

- ***Integración con Google Cloud.***
  COS se integra estrechamente con Google Cloud, lo que significa que se beneficia de la autenticación y autorización de la infraestructura de Google gestionándose a través de la misma consola.

4.2.4 Prestaciones y recursos de las máquinas virtuales.

En este apartado se va a configurar las máquinas virtuales que forman los nodos. Estas VM's tienen unos recursos en CPU, RAM y almacenamiento SSD además de un sistema operativo.

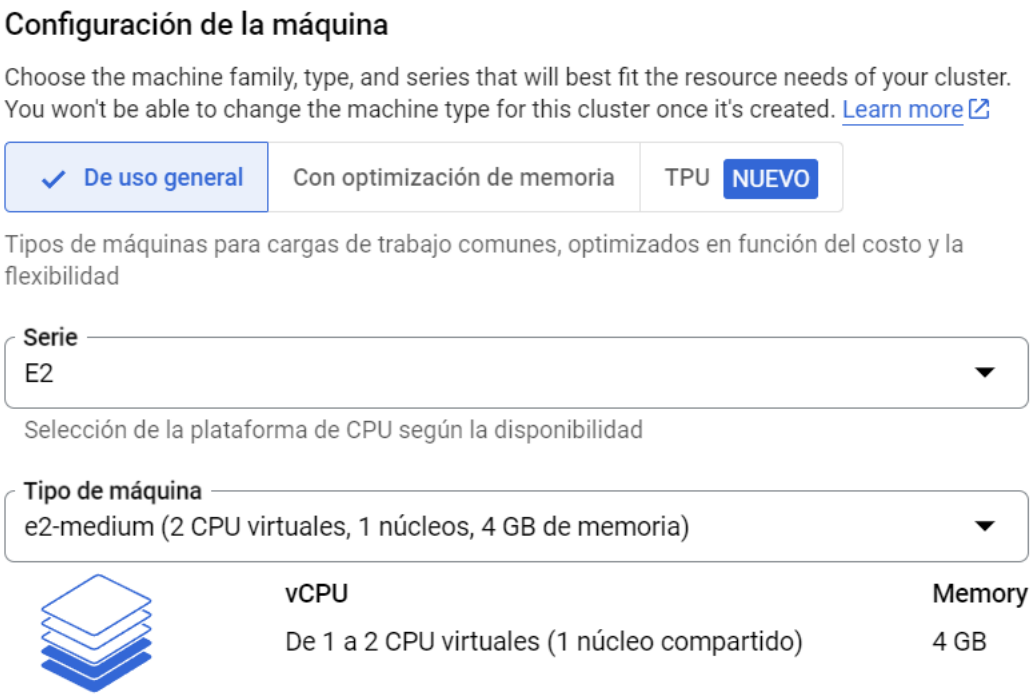

Figura 27. Recursos de las VM's.

Se ha selecciona una máquina de "*Uso General*" que es óptima para cargas de trabajo comunes y proporciona un buen balance entre coste y flexibilidad. La serie seleccionada es la "*E2*" y el tipo de máquina es la "*e2-medium*" que tiene 2 vCPU's y 4G de RAM.

Figura 28. Almacenamiento de las VM's.

He reducido el almacenamiento por defecto de 100GB a 40GB por cada uno de los nodos.

4.2.5 Seguridad en los nodos.

En esta sección se va a configurar la seguridad que se aplica a los nodos que forman el clúster:

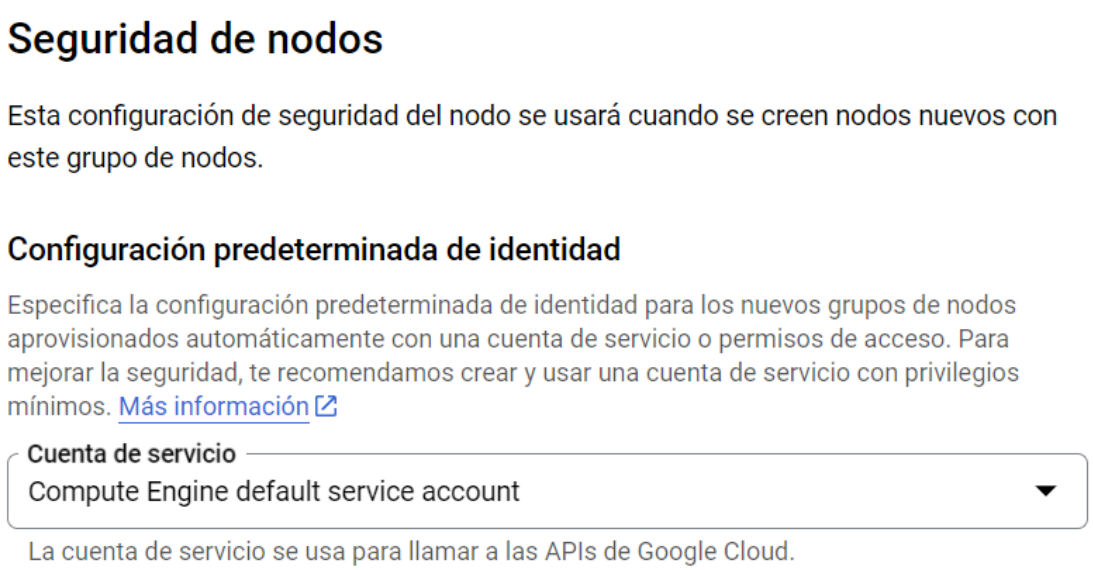

Figura 29. Configuración predeterminada de identidad.

La cuenta de servicio es un tipo de cuenta de usuario que está destinada a ser utilizada por aplicaciones o servicios para interactuar con los recursos de Google Cloud Platform. En este caso, se está utilizando la cuenta de servicio predeterminada de Compute Engine para realizar llamadas a las API's de Google Cloud.

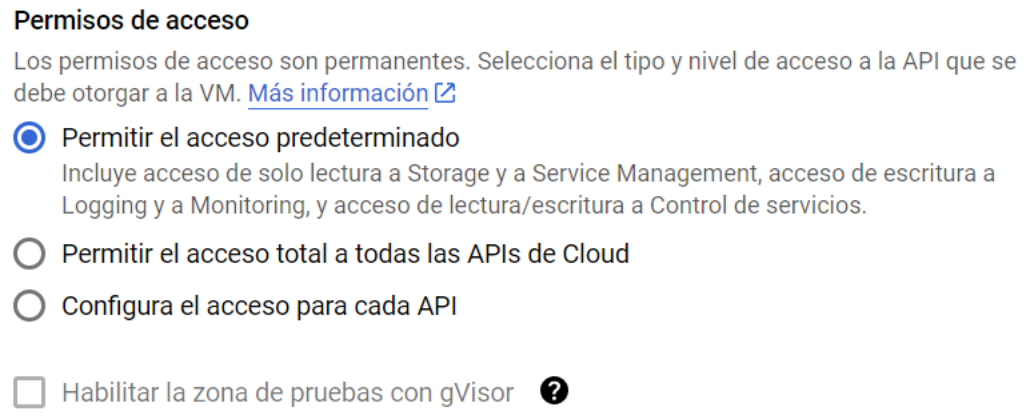

Figura 30. Permisos de acceso.

En la imagen anterior, se visualiza los permisos de acceso para las máquinas virtuales que forman los nodos. Hay tres opciones de configuración que se explican a continuación:

- ***Permitir el acceso predeterminado.***
  Esta opción incluye el acceso de solo lectura a servicios como *Storage* y *Service Management*. También acceso de escritura a *Logging* y *Monitoring*, además de acceso de lectura/escritura a Control de Servicios.

- ***Permitir el acceso total a todas las API's de Cloud.***
  Se habilita el acceso a todas las API's de Google Cloud.

- ***Configurar el acceso para cada API.***
  Esta opción permite especificar permisos de acceso individuales para cada API.

En la imagen anterior se parecía la posibilidad de habilitar la zona de pruebas con gVisor, que es una tecnología de aislamiento de contenedores para proporcionar seguridad adicional.

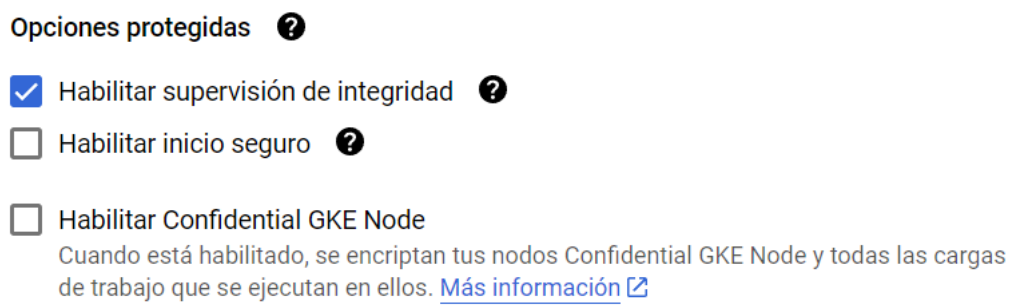

Figura 31. Opciones protegidas.

En la imagen anterior se visualiza las opciones de seguridad para GKE:

- ***Habilitar supervisión de integridad.***
  Esta opción está marcada y permite la supervisión de la integridad del nodo para detectar y reportar posibles compromisos de seguridad.

- ***Habilitar inicio seguro.***
  No está marcada, pero si se habilitara, proporciona una capa adicional de verificación durante el proceso de arranque del nodo para asegurarse de que no está comprometido.

- ***Habilitar Confidential GKE Node.***
  No está marcada, pero si se habilitara, proporciona la encriptación de los nodos de GKE y de las cargas de trabajo que se ejecutan en ellos, proporcionando una seguridad de datos mejorada en entornos nube.

4.2.6 Herramientas de redes.

Ahora toca la configuración de red en el clúster de Kubernetes, que define como se comunicarán las aplicaciones entre sí y con el plano de control de Kubernetes. También se configura como los clientes pueden acceder a ellas:

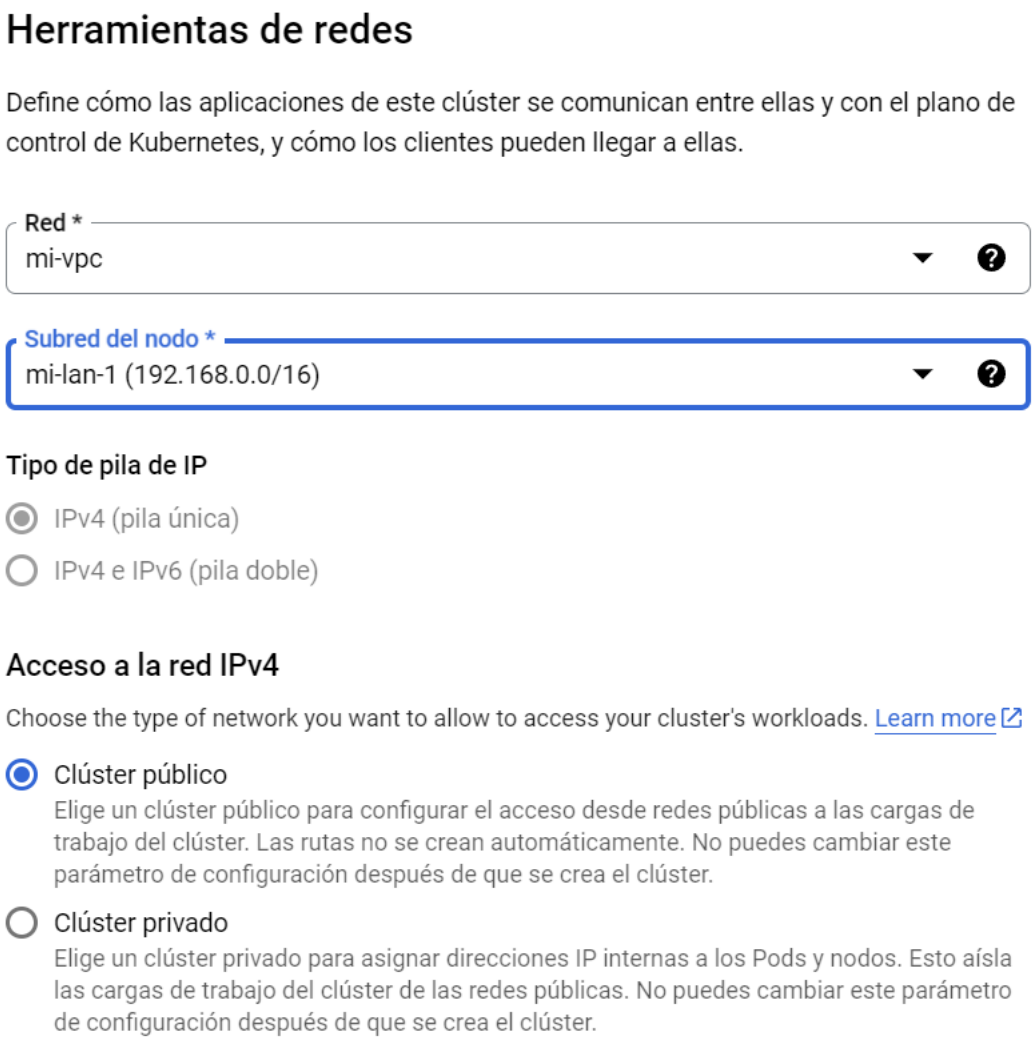

Figura 32. Configuración de red.

- ***Red***
  Se establece la red privada virtual configurada en el apartado 4.1.

- ***Subred de nodo***
  Se ha seleccionado la subred 192.168.0.0/16 como la subred donde residirán los nodos del clúster.

- ***Acceso a la red IPv4***
  Se ofrece la posibilidad de que el clúster sea accesible mediante redes públicas o de forma privada, aislando los nodos y los pods de las redes públicas.

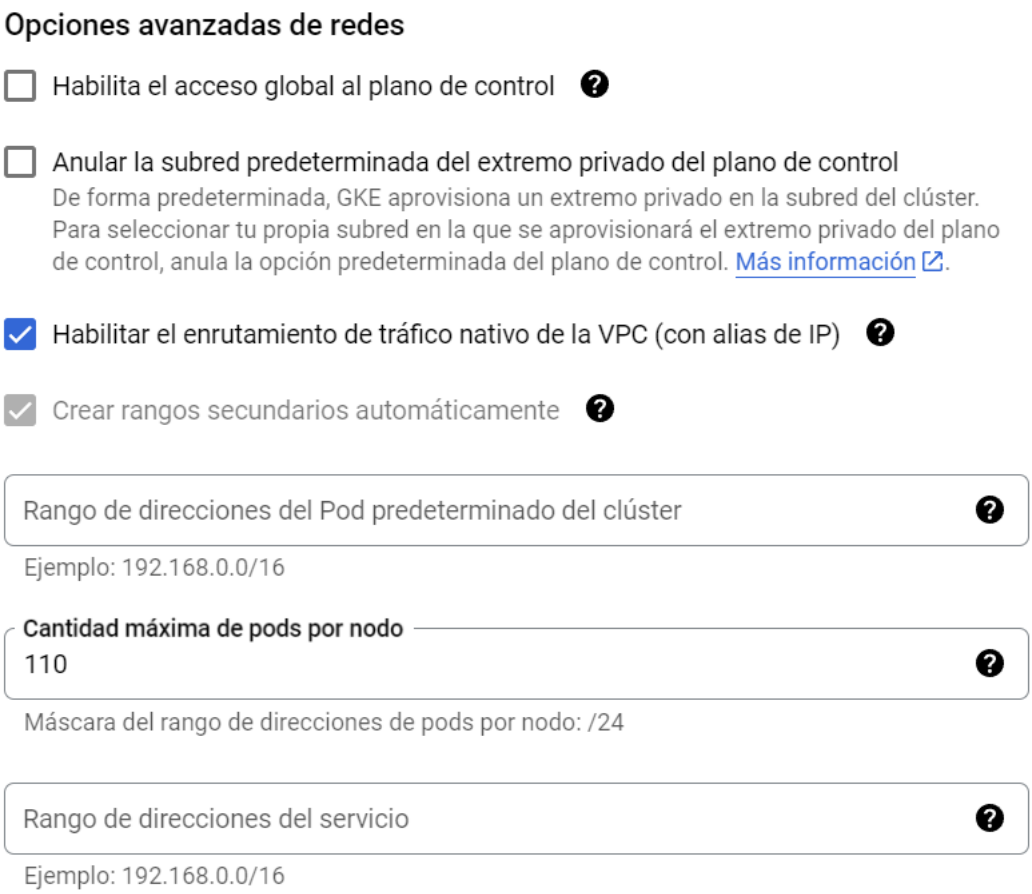

Figura 33. Opciones avanzadas de redes.

En la imagen anterior podemos visualizar algunas de las opciones avanzadas de redes, como la opción de habilitar el enrutamiento de tráfico nativo de la VPC, permitiendo el uso de alias de IP para los recursos del clúster.

La opción activada por defecto "*Habilitar el enrutamiento de tráfico nativo de la VPC con alias de IP*" es esencial para asignar direcciones IP a los pods y servicios dentro del clúster.

Otra opción activa por defecto es el balanceo de cargas HTTP. Es una técnica para distribuir el tráfico de red o las solicitudes de conexión entre varios servidores basado en el protocolo HTTP. Se utiliza para optimizar recursos, maximizar el rendimiento, minimizar los tiempos de respuesta y evitar la sobrecarga que padecería el utilizar solamente un servidor.

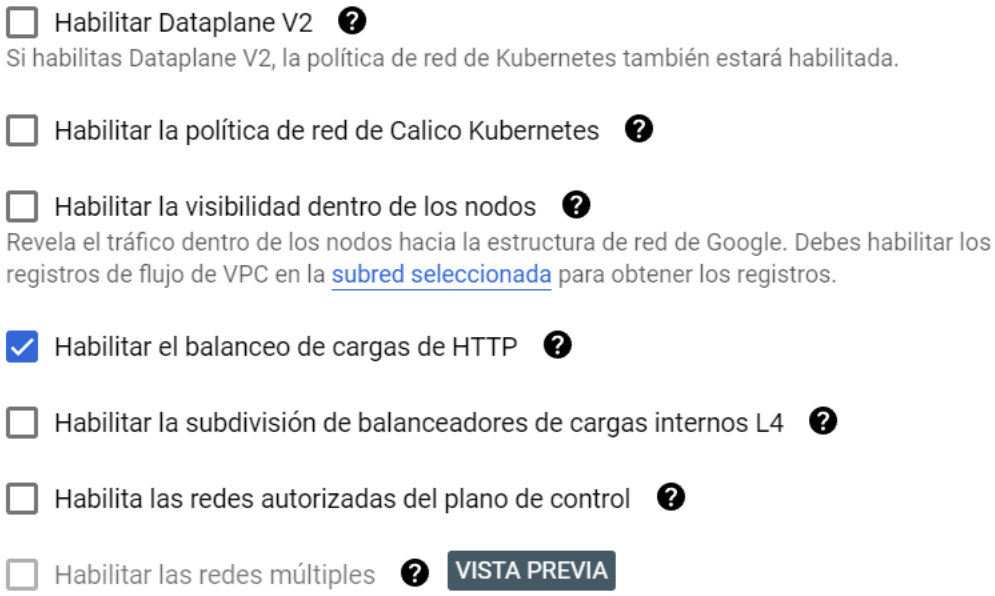

Figura 34. Opciones avanzadas de redes, parte 2.

Para finalizar este apartado, se muestra la configuración por defecto para un proveedor de DNS dentro de la interfaz de Kubernetes:

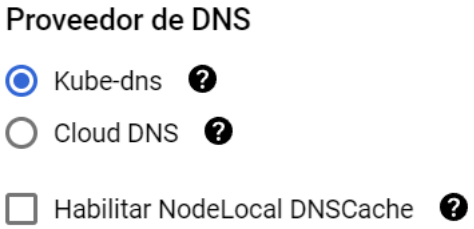

Figura 35. Proveedor de DNS.

- ***Kube-dns***
  Kube-dns es un servicio de DNS interno de Kubermetes que permite a las aplicaciones y servicios desplegados dentro de un clúster la posibilidad de conectarse mediante nombres de dominio DNS.

- ***Cloud DNS***
  Otra opción de proveedor DNS altamente disponible y escalable proporcionado por Google como puede ser Google Cloud DNS.

4.2.7 Seguridad en el clúster.

El punto 7 corresponde a la seguridad del clúster de Kubernetes. Nos sumergiremos explorando las herramientas y prácticas que Google proporciona para mantener los clústeres seguros y robustos. Abordaremos como GKE fortalece la seguridad en varios niveles, desde la gestión de identidad y el acceso hasta la protección en tiempo de ejecución de las cargas de trabajo.

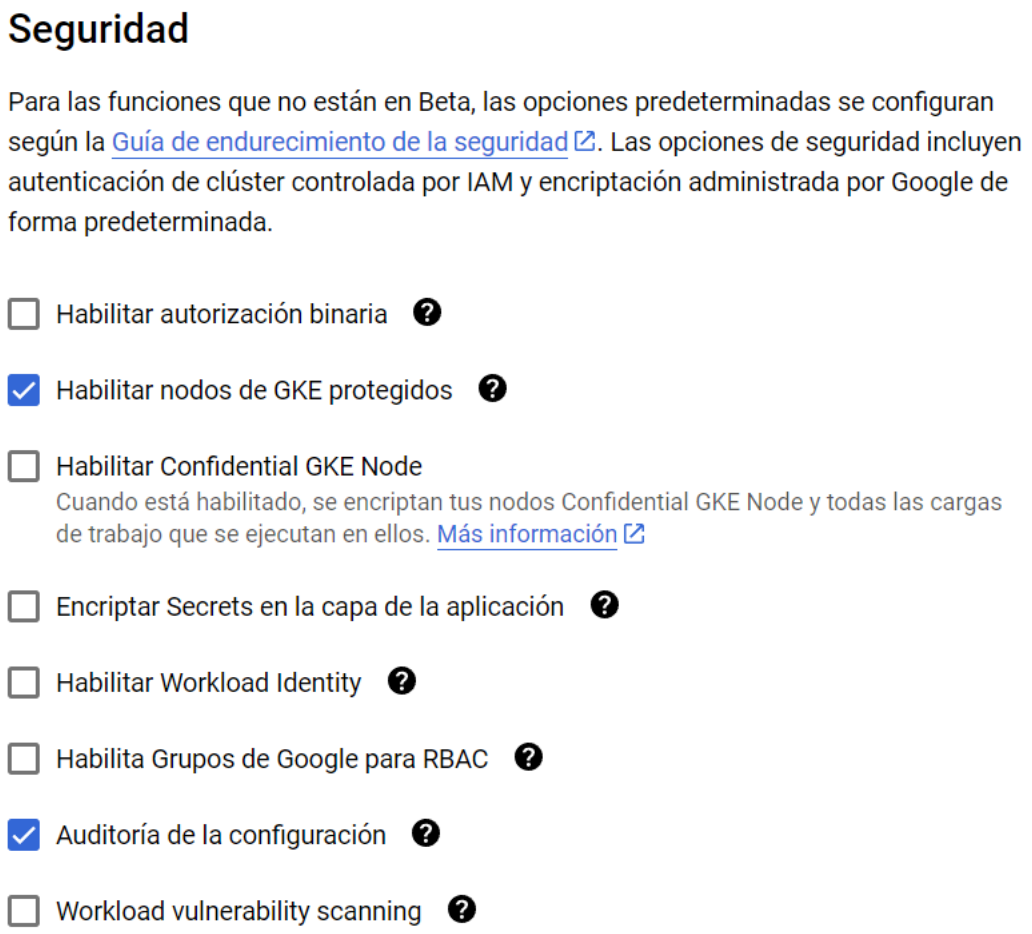

Figura 36. Seguridad básica en GKE.

- ***Habilitar autorización binaria.***
  La autorización binaria es un proceso de seguridad en GKE que permite a los administradores requerir que las imágenes de los contenedores pasen por ciertas verificaciones antes de ser desplegadas.

- ***Habilitar nodos de GKE protegidos.***
  Los nodos protegidos en GKE son una característica de seguridad que ofrece una capa adicional de defensa contra las amenazas de seguridad al asegurar el sistema operativo subyacente de los nodos del clúster.

- ***Habilitar nodos de GKE protegidos.***
  Se ha comentado anteriormente. Los nodos confidenciales en GKE permiten a los usuarios de Google Cloud ejecutar cargas de trabajo en GKE en un entorno donde la memoria de los nodos es encriptada añadiendo una capa adicional de protección.

- ***Encriptar Secrets en la capa de aplicación.***
  Esta opción da la capacidad de encriptar secretos (contraseñas, tokens, claves privadas…) a nivel de aplicación para proteger la información sensible.

- ***Habilitar Workload Identity.***
  Workload Identity es una forma de proporcionar a las aplicaciones en GKE una identidad de Google Cloud que les permite acceder a los recursos de Google Cloud Platform sin necesidad de utilizar claves de servicio estáticas.

- ***Habilita Grupos de Google para RBAC.***
  Implica el uso de grupos de Google para gestionar el acceso basado en roles (RBAC) dentro del clúster, permitiendo a los administradores asignar permisos a grupos enteros en lugar de usuarios individuales, facilitando la gestión de permisos.

- ***Auditoría de configuración.***
  Implica que el clúster tiene habilitada la auditoria de configuración, lo que significa que hay un registro de las configuraciones que han sido aplicadas permitiendo rastrear cambios por razones de seguridad.

- ***Workload vulnerability scanning.***
  Escanea las cargas de trabajo en busca de vulnerabilidades conocidas, por lo que identifica y remedia posibles vulnerabilidades ante de ser explotadas.

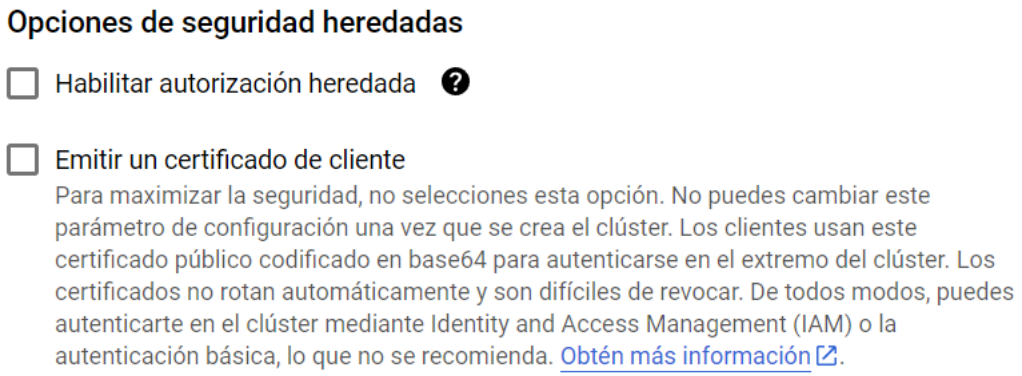

Figura 37. Opciones de seguridad heredadas.

Las opciones de seguridad heredadas se refieren a métodos de seguridad más antiguas que no son recomendables debido a la disponibilidad de otras opciones más seguras y modernas. Normalmente se mantienen disponibles para la compatibilidad con sistemas más antiguos.

4.2.8 Automatización del clúster.

En el apartado 8, hablaremos sobre la automatización de acciones para el mantenimiento, el ajuste de escala y el aprovisionamiento automático.

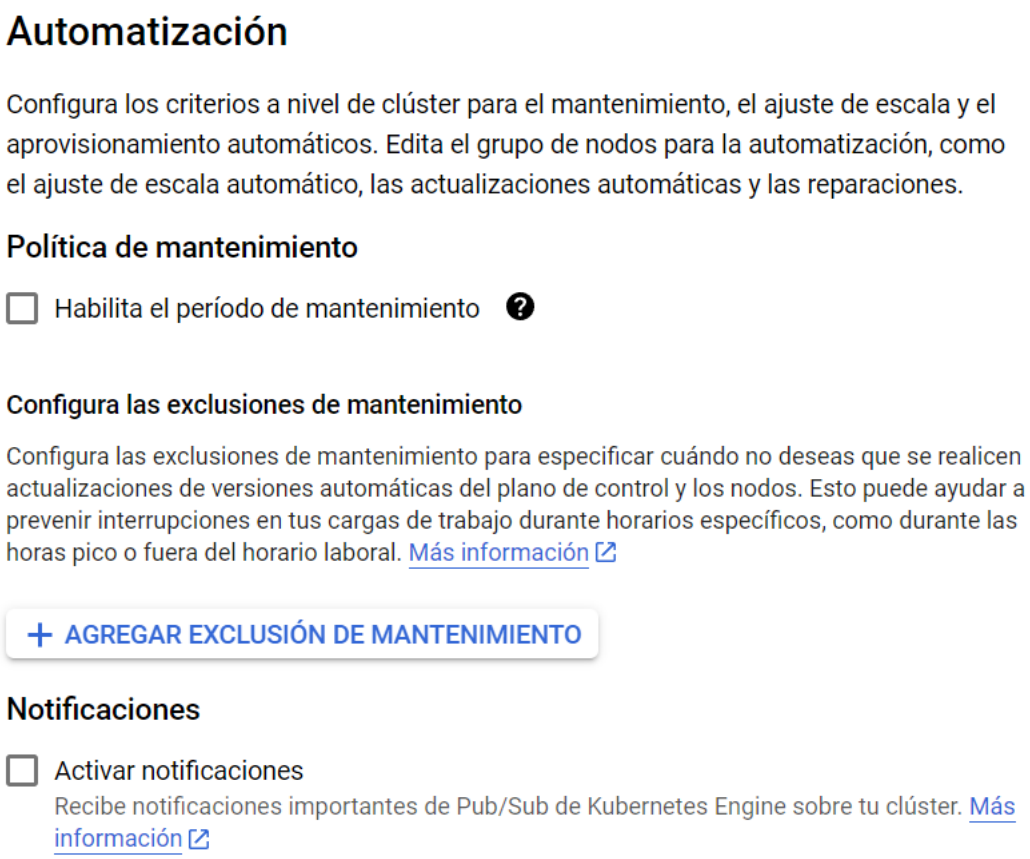

Figura 38. Automatización

- ***Habilita el período de mantenimiento.***
  Permite definir un período de tiempo o franja horaria durante el cual se llevan a cabo las operaciones de mantenimiento como actualizaciones y parches de forma programada.

- ***Activar notificaciones***
  Si esta activa, los usuarios recibirán notificaciones importantes sobre los Pub/Sub de Kubernetes Engine facilitando la monitorización y el poder dar respuesta a eventos importantes relacionados con el clúster.

Para finalizar, se comenta los ajustes de escalado automático de nodos en base a las cargas de trabajo recibidas por el clúster:

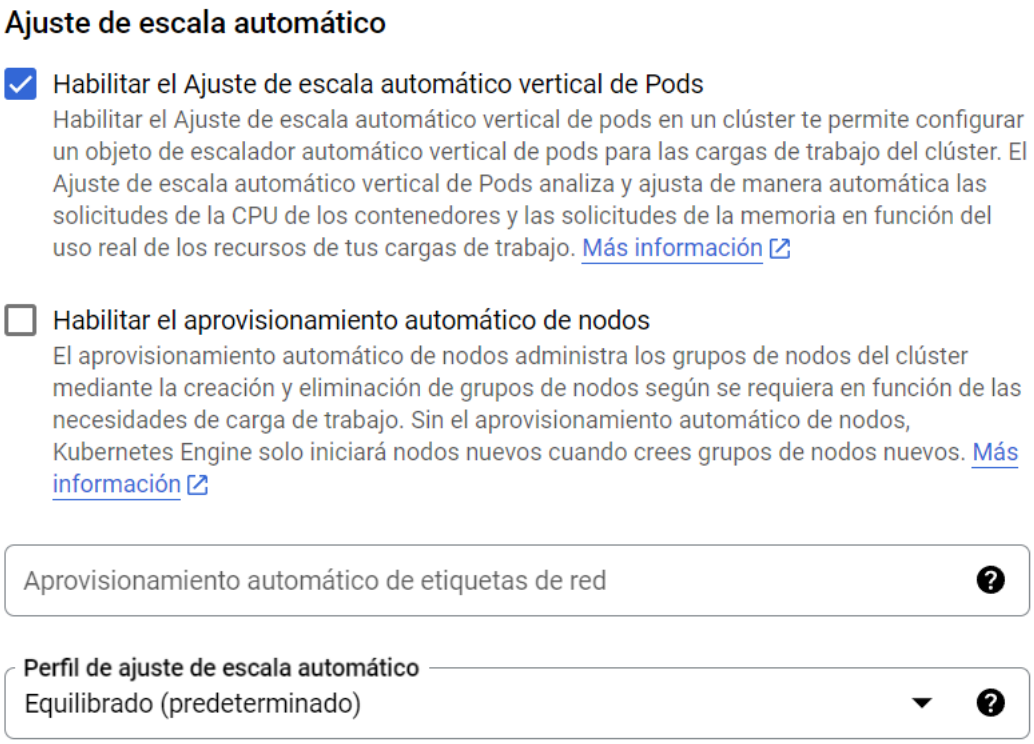

Figura 39. Escalado automático.

- ***Habilitar el ajuste de escala automático vertical de Pods.***
  Esta opción permite configurar el escalado automático de los recursos de los contenedores dentro del clúster. El sistema puede ajustar automáticamente la cantidad de CPU y memoria asignada a los pods, basándose en el uso actual y las solicitudes de recursos.

- ***Habilitar el aprovisionamiento automático de nodos.***
  Esta configuración administra la adición y eliminación de nodos de manera automática. Dependiendo de la demanda de recursos o la carga de trabajo, el sistema puede iniciar nuevos nodos o eliminar existentes para asegurar el buen funcionamiento de las aplicaciones en ejecución.

4.2.9 Características avanzadas.

El punto 9 vemos las características avanzadas respecto a la creación del clúster de Kubernetes:

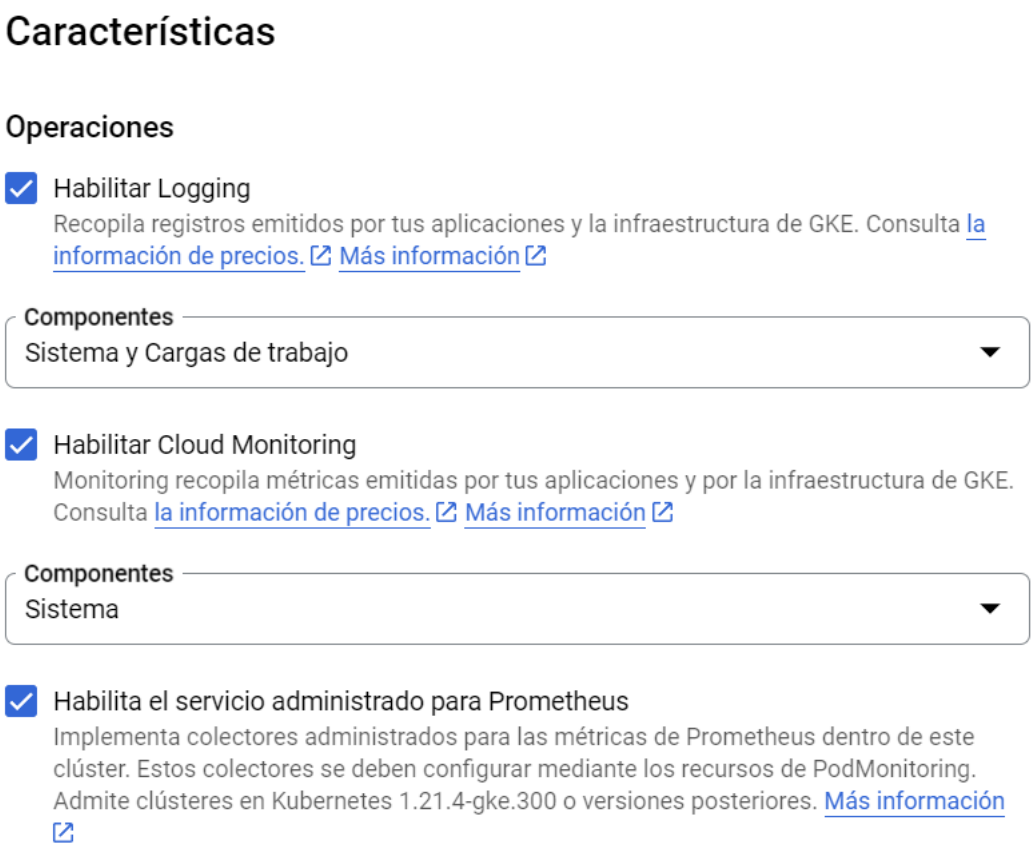

Figura 40. Características avanzadas.

- ***Habilitar Logging***
  Activa la recopilación de registros emitidos por las aplicaciones y la infraestructura propia de GKE. Los registros son vitales para el diagnóstico de problemas y el análisis de la actividad del sistema.

- ***Habilitar el Cloud Monitoring***
  Esta función activa la recopilación de métricas emitidas por las aplicaciones y por la infraestructura de GKE. El monitoreo es esencial para mantener la salud del sistema, realizar el escalado óptimo y responder proactivamente a los cambios en el comportamiento de la aplicación o a la carga de trabajo.

- ***Habilita el servicio administrado para Prometheus.***
  Esta opción está marcada y refiere a la implementación de colectores administrados para las métricas de Prometheus dentro del clúster. Prometheus es una herramienta de monitoreo y alerta de código abierto ampliamente utilizada en el ecosistema de Kubernetes.

4.2.10 Malla de servicios.

El "*Anthos Service Mesh*" es un servicio gestionado que proporciona capacidades de malla de servicios para microservicios. Una *malla de servicios* es una capa de infraestructura configurable que permite la comunicación entre distintos servicios de una aplicación, ofreciendo características como el descubrimiento de servicios, balanceo de carga, cifrado, autenticación y autorización. Esto se hace forma transparente, sin que los servicios individuales necesiten implementar estas cosas por sí mismos.

Figura 41. Malla de servicios.

La activación de este servicio en el clúster que estamos creando nos proporciona los siguientes beneficios:

- ***Comunicación administrada.***
  ASM gestiona y controla la forma que los servicios se comunican entre sí, lo que reduce la complejidad y el esfuerzo de configuración para los desarrolladores y administradores de sistemas.

- ***Monitorización.***
  ASM ofrece herramientas para monitorización para así entender el tráfico entre servicios, facilitando la identificación y la solución de problemas.

- ***Seguridad.***
  ASM permite la comunicación segura entre servicios ya que proporciona cifrado de tráfico y políticas de seguridad que se pueden aplicar de forma consistente.

4.2.11 Otras opciones.

Nos acercamos a la penúltima sección de la configuración el clúster con GKE accediendo a las opciones que no están dentro de ningún apartado anterior. Vamos a repasarlas:

Figura 42. Otras opciones, parte 1.

- ***Habilitar Cloud TPU.***
  Permita acelerar las cargas de trabajo de aprendizaje automático en el clúster utilizando las Cloud TPUs (Unidades de Procesamiento de Tensor), que son aceleradores de hardware específicos para tareas de aprendizaje profundo.

- ***Habilitar funciones alfa de Kubernetes en este clúster.***
  Permite el uso de características experimentales de Kubernetes que aún están en versión alfa, siendo estas versiones tempranas y que no pueden ser estables o cambiar en el futuro.

- ***Habilitar imputación de costos.***
  Proporciona una visión detallada del uso de recursos, desglosado por espacios de nombre y etiquetas dentro de Kubernetes facilitando la atribución del costo de los recursos a las entidades significativas permitiendo una facturación más detallada.

- ***Habilitar la copia de seguridad para GKE.***
  Crea copias de seguridad de los recursos y cargas de trabajo permitiendo una recuperación ante desastres. Los costos de este servicio se basan en la cantidad de datos a recuperar y el número de Pods que se protegen.

- ***Habilitar el controlador SCSI de Peristent Disk para Compute Engine.***
  Activa el controlador de la interfaz de *Almacenamiento de Contenedores* (CSI) para los persistent Disks de Google Compute Engine, proporcionando una gestión automática y nativa de los volúmenes de almacenamiento para los contenedores.

Figura 43. Otras opciones, parte 2.

- ***Habilitar el controlador CSI de Filestore.***
  Esta opción permite implementar y administrar automáticamente el controlador CSI (*Containter Storage Interface*) para FileStore en el clúster de Kubernetes permitiendo la conexión y gestión de volúmenes de almacenamiento en los contenedores de forma automática.

- ***Habilitar el controlador de CSI del FUSE de Cloud Storage.***
  FUSE (*Filesystem in Userspace*) es una interfaz que permite a los usuarios crear su propio sistema de archivos sin editar código del kernel. Esta opción permite la implementación y gestión automática del controlador que conecta el sistema de archivos del usuario con el almacenamiento de Cloud Storage como sistemas de archivo dentro de los contenedores.

- ***Habilitar la transmisión de imágenes.***
  Esta función permite que las cargas de trabajo se inicien sin la necesidad de esperar a que se descargue completamente la imagen del contenedor acelerando el inicio de los contenedores y la escalabilidad de las aplicaciones.

4.2.12 Registro del clúster en una flota.

Y con este punto llegamos a la sección final de la creación de un clúster GKE en Google Cloud. El último paso es el registro del clúster en una flota:

Figura 44. Registro del clúster en una flota.

Definimos *flota* como una agrupación o colección de clústeres que son administrados como un conjunto. Es útil para la administración centralizada de varios clústeres, permitiendo aplicar políticas y configuraciones de forma coherente a todos los clústeres registrados en una flota.

Para poder utilizar Anthos, el servicio de Google Cloud que permite la gestión moderna de aplicaciones es necesario registrar el clúster de Kubernetes en una flota de Anthos.

4.2.13 Activación y costo del clúster.

En forma de aclaración, he de comentar que el clúster que se crea para la realización del proyecto es en base a las prestaciones marcadas con un check (☑) en todas las opciones comentadas y visualizadas en las figuras anteriores.

Con estas opciones marcadas y con las prestaciones comentadas, el clúster nos cuesta mensualmente un importe de *274,98$* o *255,24€* en el momento de redacción de este texto:

Figura 45. Coste mensual aproximado final del clúster.

El coste entra, perfectamente, dentro de los 283€ gratuitos que proporciona Google al registrarse en la plataforma. Para la creación del clúster, pulsamos en el botón "*Crear*".

Al pulsar el botón, empieza la creación del clúster, tal y como se muestra en la figura siguiente:

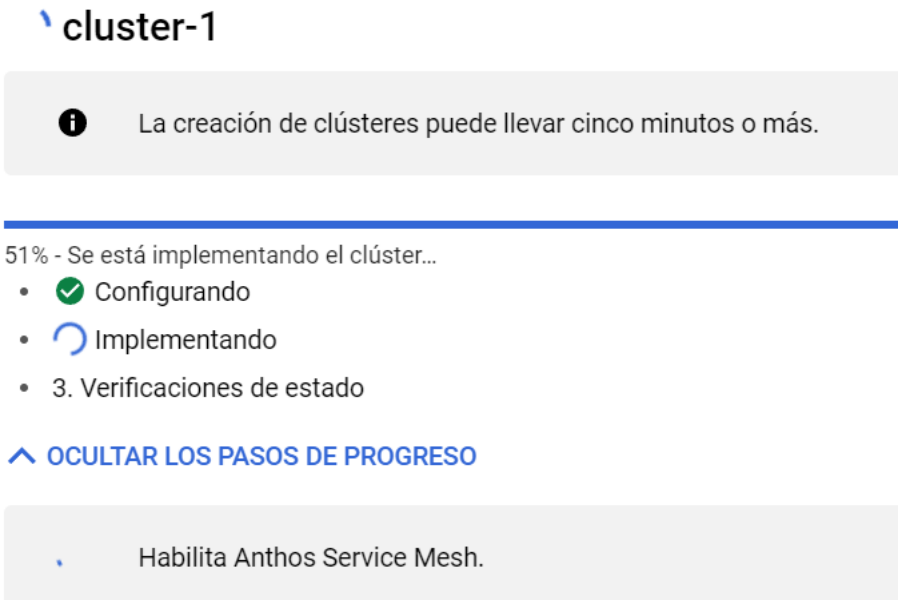

Figura 46. Proceso de creación del clúster.

Después de unos cinco minutos aproximadamente, el clúster de GKE ya estará disponible y operativo cómo se puede ver a continuación:

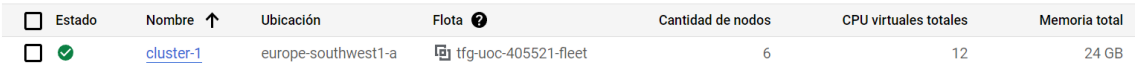

Figura 47. Clúster "*clúster-1*" activo.

Podemos apreciar que la ubicación es la región de Madrid (*europe-southwest1-a*) y la zona de disponibilidad "*a*". Que está registrado a la flota *tfg-uoc-405521-fleet* y que por último tiene el clúster 6 nodos (3 en la zona a y 3 más en la zona b) con unos recursos computacionales totales de:

- **12 vCPU's** (*2CPU's por VM x 6 VM's*)
- **24 GB de RAM** (*4GB de RAM por VM x 6 VM's*)

Por último y antes de finalizar este apartado, cuando se estaba creando el clúster y al registrar la flota en una malla de servicios de Anthos, aparece que debemos de habilitar Anthos para que registro de la flota sea efectivo. Esta gestión la haremos en el punto siguiente.

5. Activación y despliegue de una aplicación con Anthos.

En este capítulo, exploraremos la activación de Anthos, una plataforma de gestión y despliegue de aplicaciones moderna, ofrecida por Google Cloud, en nuestro recién creado clúster de Kubernetes.

5.1 Activación de Anthos para el despliegue de aplicaciones.

Primero de todo, debemos de habilitar el servicio Anthos en nuestra cuenta de Google Cloud. Para ello en la barra de búsqueda introducimos “Anthos” y aparecen varios resultados y debemos de seleccionar el siguiente:

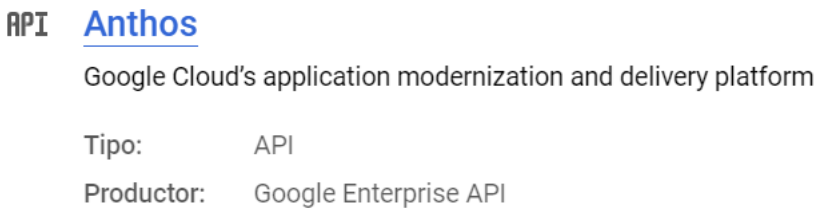

Figura 48. Selección de la API de Anthos.

Una vez hemos accedido, activamos la API haciendo clic en el botón de habilitar:

Figura 49. Activación del servicio en GCP “Anthos”.

Seguimos con la selección de la API de Anthos Config Management. Esta API permite utilizar el *Config Management* siendo esta una herramienta dentro de la plataforma Anthos que permite a los administradores de TI automatizar y aplicar configuraciones a través de sus clústeres de Kubernetes. La API de Anthos Config Management permite interactuar con esta herramienta para llevar a cabo tareas como:

- ***Configuración a gran escala.***
  Se puede usar a Anthos Config Management para crear políticas de configuración que se crean de manera consistente en todos los clústeres, ya sean dentro de Google Cloud o en entornos multi-nube u *on premise*.

- ***Sincronización de configuración.***
  La herramienta sincroniza las configuraciones desde un repositorio Git, permitiendo que las configuraciones sean versionadas y auditadas.

- ***Interfaz declarativa.***
  Utiliza archivos de configuración declarativos para definir el estado deseado de tus recursos y políticas en los clústeres de Kubernetes.

Seguimos con la siguiente API a activar. Anthos Audit es una herramienta que permite auditar y monitorear las operaciones dentro del entorno Anthos. La auditoría es un aspecto crítico de la gestión de infraestructuras y aplicaciones, proporcionando registros detallados de quien hizo qué y cuando. Esta información puede ser vital para el seguimiento de cambios, para la identificación de problemas o el cumplimiento de las regulaciones y/o seguridad.

Habilitar la Anthos Audit API permite:

- ***Recoger datos de la auditoría.***
  Recoger datos de forma automática que permiten saber sobre cómo se utilizan los recurso en Anthos y cómo se accede a ellos.

- ***Integración con herramientas de monitoreo y gestión de logs.***
  Permite la utilización de estos datos para integrarse con herramientas de terceros o de Google para analizar y visualizar los registros de la auditoria.

- ***Cumplimiento y regulaciones y seguridad.***
  Utiliza archivos de configuración declarativos para definir el estado deseado de tus recursos y políticas en los clústeres de Kubernetes.

Y, por último, vamos a activar la API de Anthos GKE que nos permite realizar operaciones cómo:

- Crear, configurar y gestionar clústeres de Kubernetes que se ejecutan en Anthos.
- Automatizar el despliegue y la gestión de aplicaciones en diferentes entornos nube.
- Utilizar las funcionalidades avanzadas de GKE en Anthos.

Una vez activadas todas las API’s, seguimos avanzando.

5.2 Seguridad del clúster antes del despliegue de aplicaciones.

Antes de desplegar alguna aplicación, debemos de ver en qué estado se encuentra el clúster de Kubernetes y realizar algunas acciones para verificar la seguridad. Para ello accedemos a Kubernetes Engine y a la sección de "*Descripción General*" detectando un clúster dentro de la flota (*tfg-uoc-405521-fllet*) en estado saludable:

Figura 50. Clúster creado en estado saludable.

Una vez verificado que el clúster está levantado y funcionando correctamente, seguimos con la verificación. Para ello debemos de habilitar la "*Containter Security API*" pulsando en el botón para realizar dicha acción:

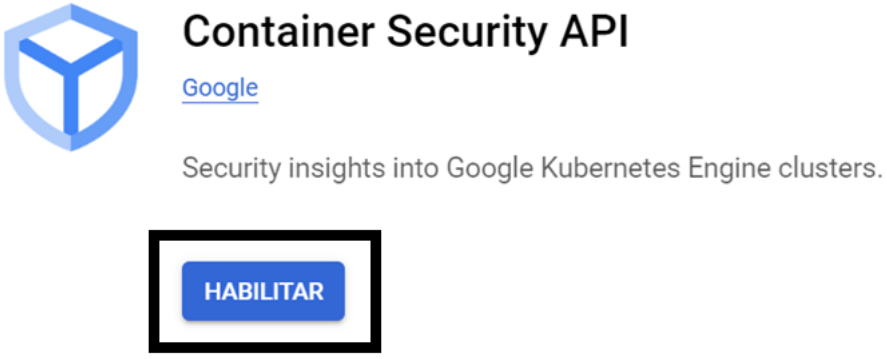

Figura 51. Activación de la API "Container Security".

Una vez habilitada la API, podremos ver si nuestro clúster tiene problemas de seguridad.

5.2.1 Problemas de seguridad encontrados.

Pasados unos segundos, Google Cloud ha analizado nuestro clúster obteniendo 12 problemas de seguridad de nivel alto:

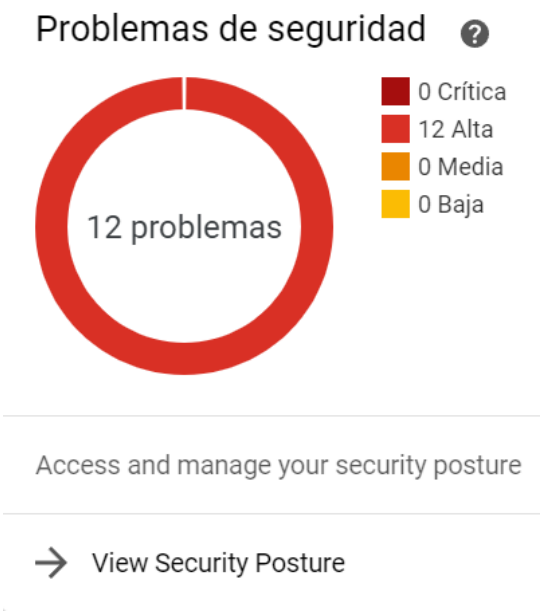

Figura 52. Problemas de seguridad de nuestro clúster.

Debemos de acceder en “View Security Posture” para que se nos cargue un dashboard que nos proporciona la posibilidad de analizar los problemas con más detalle:

Figura 53. Dashboard con los problemas de seguridad encontrados.

Los indicadores son los siguientes:

- ***Consideraciones.***
  Muestra un total de 12 problemas, ordenador por un rango de severidad: 0 críticos, 12 de alta prioridad, 0 de prioridad media y 0 de baja prioridad. Esto nos indica que los 12 problemas requieren de atención inmediata.

- ***Tipos.***
  Desglosa los problemas por categoría: 0 parámetros de configuración, 12 boletines de seguridad y 0 vulnerabilidades. Esta clasificación nos indica que todos los problemas actuales están relacionados con los boletines de seguridad, lo que podría implicar la necesidad de aplicar parches o actualizaciones de seguridad.

- ***Clústeres.***
  Indica que hay un clúster de GKE y que hay un recurso afectado.

- ***Cargas de trabajo.***
  Muestra una carga de trabajo con 0 recursos afectados y 1 no afectado, por lo que las cargas de trabajo no tienen impacto por los problemas identificados.

5.2.2 Solución a los problemas de seguridad encontrados.

Al indagar más respecto a los problemas de seguridad encontrados, verificamos que tenemos una actualización del canal utilizado tanto para los nodos como para el plano de control, tal como se puede observar en la siguiente imagen:

Figura 54. Dashboard con los problemas de seguridad encontrados.

Se procede a actualizar primero el plano de control y, después, habilitándose la opción, el pool de nodos. Una vez finalizado, accedemos al analizador de vulnerabilidades, seleccionando la opción "*set a Basic*" realizamos un nuevo escaneo de problemas a nivel de contenedor:

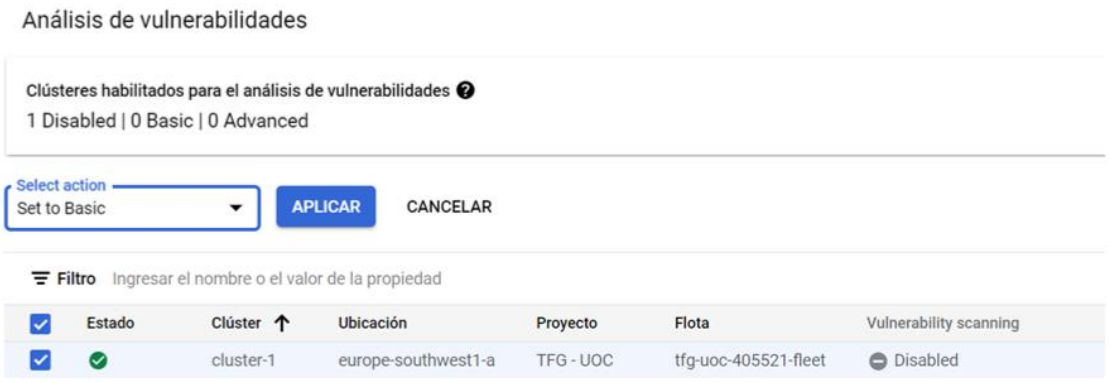

Figura 55. Análisis de vulnerabilidades.

Después de un buen rato y finalizado el análisis de vulnerabilidades, vemos que ya no nos aparece ningún problema de seguridad:

Figura 56. Problemas de seguridad solucionados.

5.2.3 Controlador de políticas.

Definimos "*controlador de políticas*" como una herramienta que ayuda a garantizar que los clústeres de Kubernetes cumplan con las políticas corporativas o de gobernanza definidas.

En nuestro caso, el sistema detecta que nuestro clúster no está dentro de política corporativa o de gobernanza y nos lo muestra de la siguiente forma:

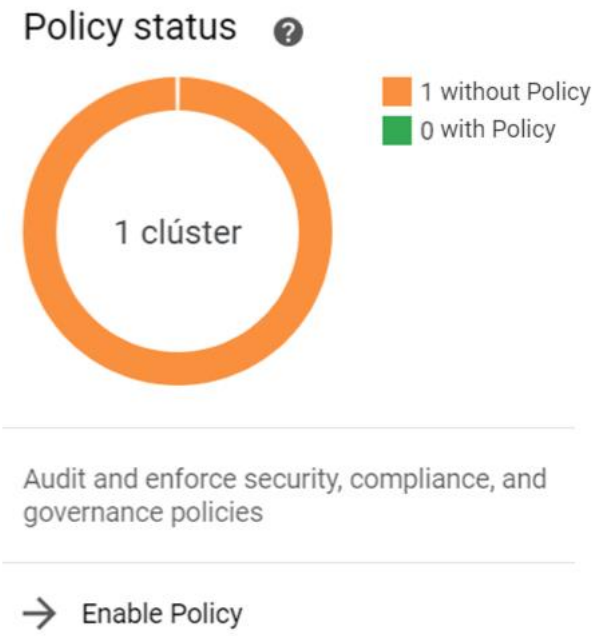

Figura 57. 1 clúster sin política de corporativa o de gobernanza.

Accedemos a "*Enable Policy*" y accedemos a la opción de instalar en nuestro clúster el controlador de políticas, tal y como se ve puede observar a continuación:

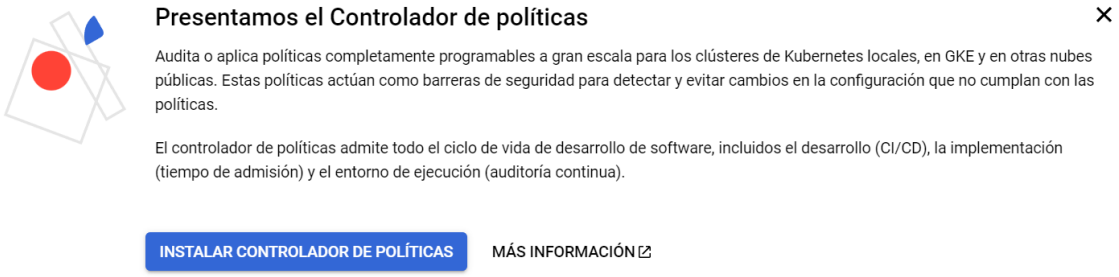

Figura 58. Opción de instalar el controlador de políticas.

Al presionar al botón "*Instalar controlador de políticas*" aparece una ventana donde nos muestra la posibilidad de instalarlo en la flota o sólo en un clúster en concreto:

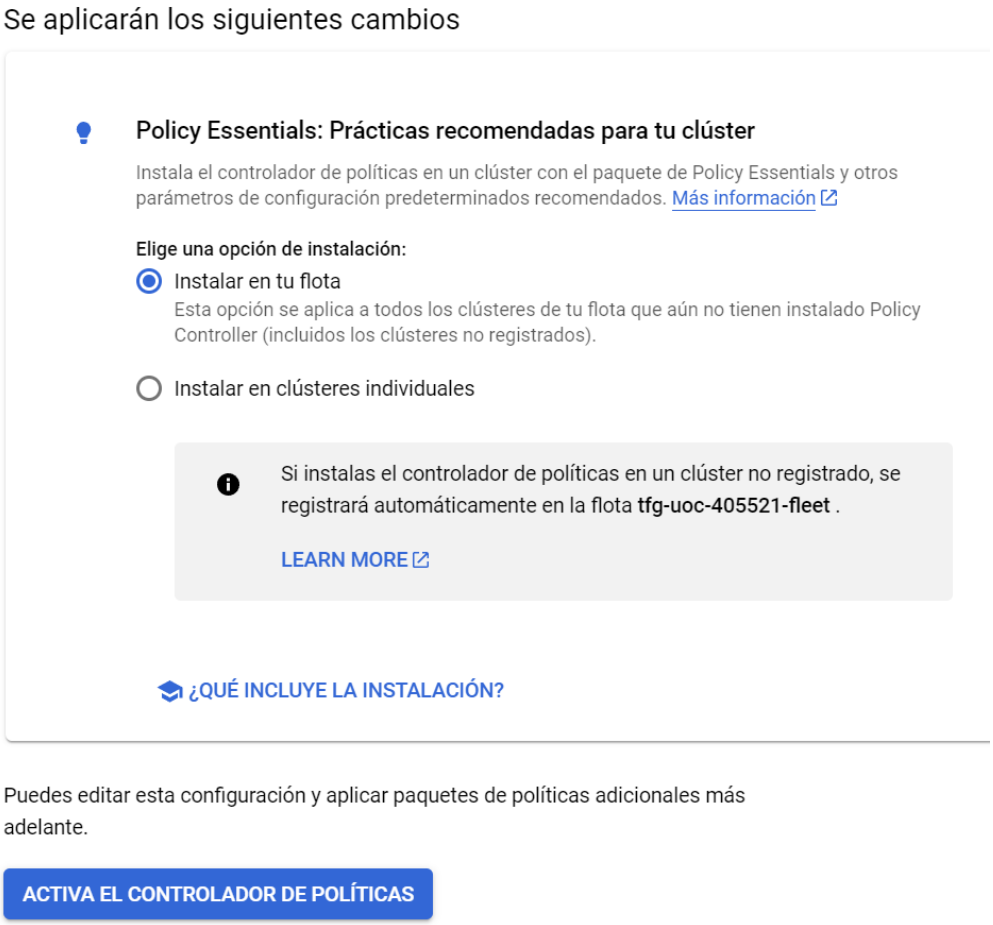

Figura 59. Opciones de instalación del controlador.

Elegimos en la flota y, acto seguid y después de esperar un par de minutos, tendremos que nuestro clúster tiene definida una política:

Figura 60 Clúster con política activada.

5.3 Despliegue de WordPress mediante GKE y Anthos.

En el punto anterior hemos dejado a punto nuestro clúster para poder trabajar con él, por lo que ahora, toca el despliegue de aplicaciones mediante Anthos a través del Marketplace de Google Cloud.

Para desplegar una aplicación, se debe ir a:

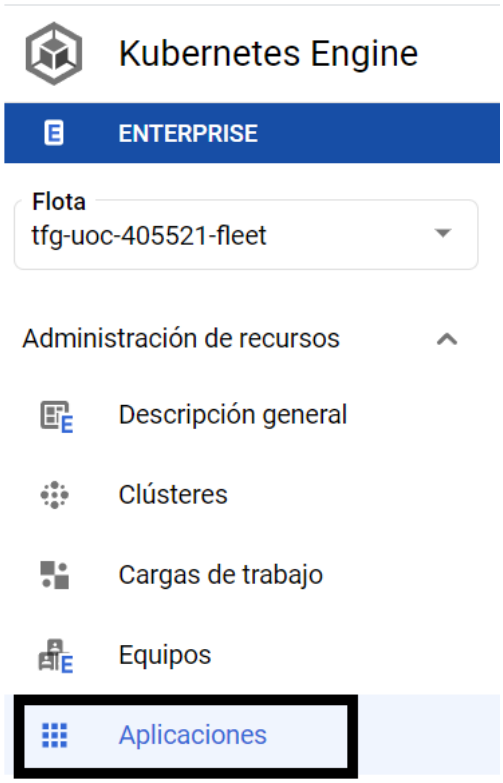

Figura 61. Sección "*aplicaciones*" en Kubernetes Engine.

Una vez dentro de la sección nos aparece el siguiente mensaje, por lo que debemos de hacer clic a "*Implementar desde el Marketplace*":

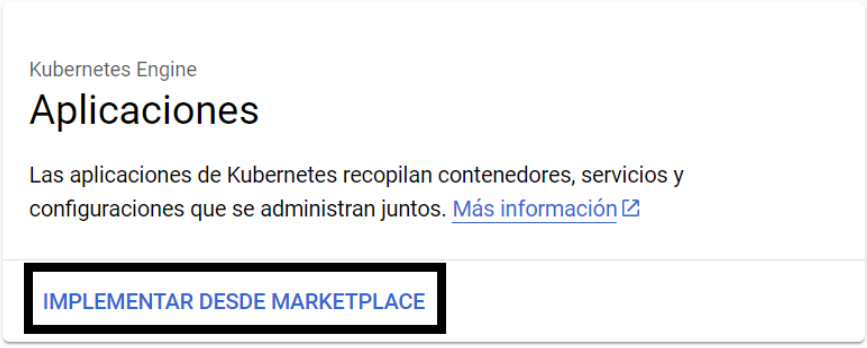

Figura 62. Sección "*aplicaciones*" en Kubernetes Engine.

Una vez se accede al Marketplace, tenemos aplicaciones divididas por categoría, precio o entorno de implementación:

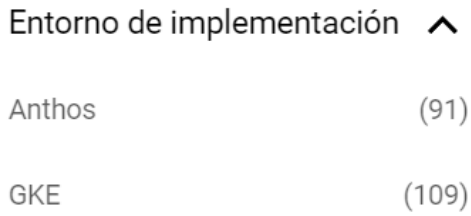

Figura 63. Las aplicaciones disponibles en los diferentes entornos.

Como se puede observar, hay más aplicaciones para despliegue en GKE que en Anthos, pero no eso no nos importa ya que la que queremos desplegar están en los dos entornos. Hacemos clic en "Anthos" para que nos liste las aplicaciones a desplegar y vemos que WordPress está disponible:

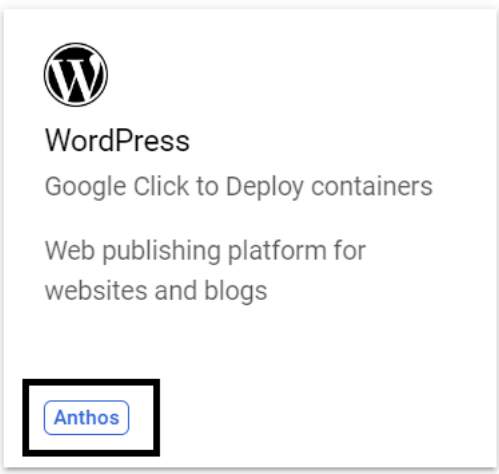

Figura 64. Despliegue de WordPress en Anthos.

Como se puede ver en la imagen anterior, aparece "*Anthos*" para informarnos que la aplicación se desplegará bajo este entorno. Hacemos clic y nos aparece una ventana en la que para poder desplegar la aplicación debemos de pulsar en "*configurar*":

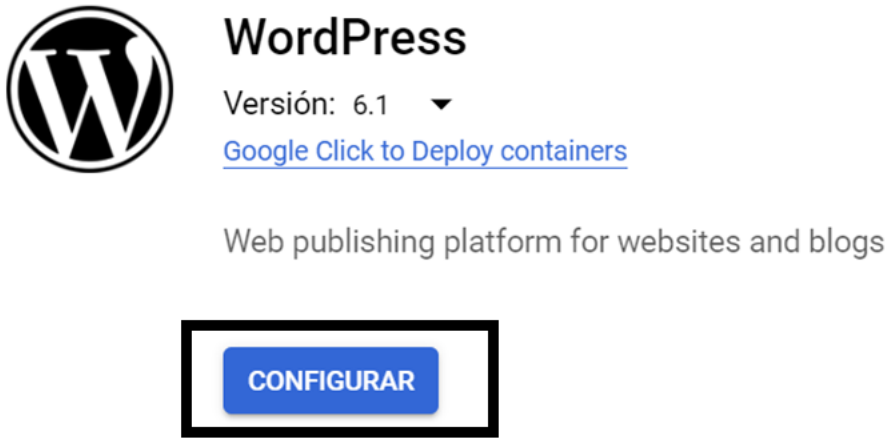

Figura 65. Despliegue de WordPress.

Una vez pulsado el botón de "*configurar*" nos aparece una ventana con las opciones de despliegue de la aplicación:

- ***Clúster de Kubernetes existente.***
  Seleccionamos el clúster creado para tal fin y llamado "*clúster-1*".

- ***Espacio de nombres.***
  Está configurado para usar el espacio de nombres "*default*", que es el espacio de nombres predeterminado em Kubernetes para recursos que no tienen un especio de nombres específico asignado.

- ***Nombre de la instancia de app.***
  Se nombre a esta instancia de WordPress "*wordpress-1*".

- ***StorageClass for WordPress Applicaction.***
  Se ha seleccionado "*premium-rwo*", lo que indica una clase de almacenamiento premium con *ReadWriteOnce* como modo de acceso, lo que significa que el volumen puede ser montado como lectura-escritura por un solo nodo.

- ***Storage size for presistent volumes in WordPress Application.***
  Se ha configurado un tamaño de almacenamiento de 5Gi, es decir, 5 Gibibytes (*poco más de 5GB*)

- ***StorageClass for MySQL Applicaction.***
  Al igual que para WordPress, se ha seleccionado "*premium-rwo*" para la base de datos.

- ***Storage size for presistent volumes in MySQL Application.***
  También se ha asignado un especio de almacenamiento de 5Gi para MySQL.

- ***WordPress admin email address.***
  He proporcionado mi correo de la UOC (*vbujan@uoc.edu*).

- ***Enable public IP access.***
  Permite el acceso a la aplicación desde una dirección IP pública.

- ***Enable Stackdriver Metrics Exporter.***
  Habilita la exportación de métricas a Stackdriver, una herramienta de monitoreo, registro y diagnóstico en la nube de Google.

  Todas las opciones descritas se pueden observar en la siguiente figura:

Una vez pulsamos el botón de "*Implementar*", Google Cloud y GKE empieza el despliegue de WordPress. Después de unos minutos podemos ya acceder al servicio activo en nuestro clúster de Kubernetes con Anthos. Para ello, se ha asignado una IP pública (34.36.173.90) y se nos proporciona el nombre de usuario y contraseña para editar el sistema de gestión de contenido (CMS):

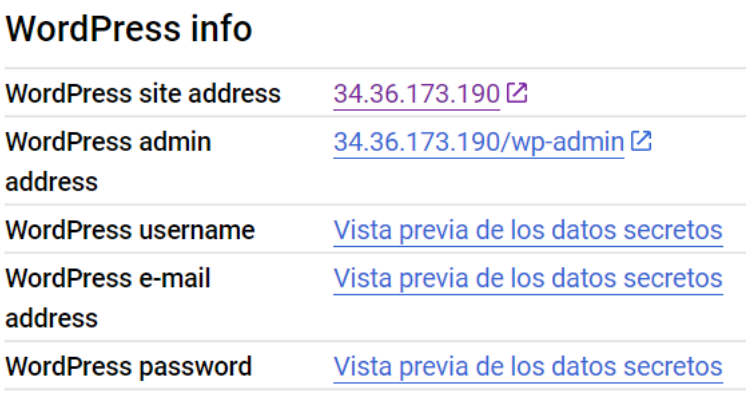

Figura 66. IP pública y datos de acceso al CMS.

Antes de acceder, vamos a comprobar que todos los componentes del WordPress se han desplegado correctamente:

Componentes

Tipo	Nombre ↑	Estado
Service	wordpress-1-apache-exporter-svc	OK
Secret	wordpress-1-deployer-config	OK
Stateful Set	wordpress-1-mysql	OK
Persistent Volume Claim	wordpress-1-mysql-pvc-wordpress-1-mysql-0	Bound
Secret	wordpress-1-mysql-secret	OK
Service	wordpress-1-mysql-svc	OK
Secret	wordpress-1-mysqld-exporter-secret	OK
Service	wordpress-1-mysqld-exporter-svc	OK
Secret	wordpress-1-tls	OK
Stateful Set	wordpress-1-wordpress	OK
Config Map	wordpress-1-wordpress-config	OK
Ingress	wordpress-1-wordpress-ingress	OK
Persistent Volume Claim	wordpress-1-wordpress-pvc-wordpress-1-wordpress-0	Bound
Secret	wordpress-1-wordpress-secret	OK
Service	wordpress-1-wordpress-svc	OK

Figura 67. Componentes desplegados de WordPress.

Como se puede apreciar, todos los componentes del Apache (servidor web), MySQL (servidor de base de datos) y el propio WordPress se han desplegado de forma satisfactoria y para comprobarlo, hacemos clic al enlace de la IP pública (34.36.173.190) para acceder al CMS:

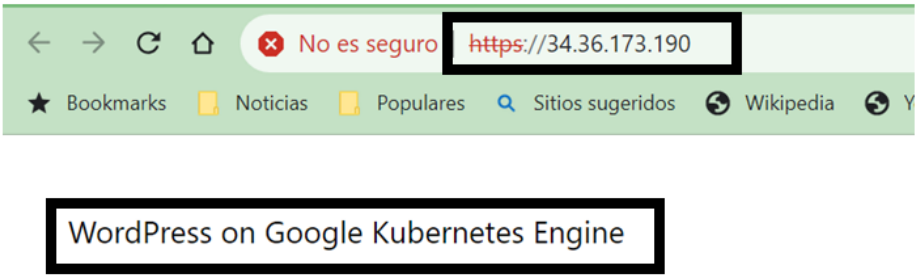

Figura 68. WordPress funcionando.

6. Creación del plan de recuperación ante desastres.

6.1 Plan de backup (copias de seguridad).

Realizar copias de seguridad regulares es una práctica esencial para la continuidad de negocio y la recuperación ante desastres. En GKE, esto podría incluir la copia de seguridad de:

- ***Configuraciones de clústeres.***
  Permite levantar rápidamente la misma configuración si fuera necesario.

- ***Datos de aplicaciones.***
  Hacer copias de seguridad para volúmenes persistentes que almacenan los datos de las aplicaciones asegura la no pérdida de la información.

- ***Objetos de Kubernetes.***
  Exportar y guardar los manifiestos de objetos como Deployments, Services y otros recursos permiten restaurar el estado deseado de las aplicaciones en Kubernetes.

Para crear copias de seguridad de un clúster GKE, debemos de seleccionar la opción "*Copia de seguridad para GKE*" y después, se selecciona "*Crear plan de creación de copias de seguridad*". Una vez se ha realizado clic, se debe de configurar las opciones de implantación:

- ***Clúster.***
  Se ha seleccionado el cluster que estamos utilizando "*cluster-1*".

- ***Nombre del plan de creación de copias de seguridad.***
  Se ha aceptado el nombre por defecto. Este es el identificador único para el plan de copias de seguridad.

- ***Región.***
  Indica la región geográfica en la que se almacenarán las copias de seguridad.

- ***Programación de la copia de seguridad.***
  Se utiliza la expresión CRON para definir el tiempo. He definido que se haga copia de seguridad cada día a las 03.00 AM con la expresión CRON "*0 3 * * ** ".

- ***Política de retención.***
  Se ha definido una política de borrado de las copias de seguridad de forma automática pasados los 7 días de antigüedad.

Vamos a realizar una copia de seguridad del WordPress de forma manual para tener un respaldo en caso de contingencia. Para ello, le damos click a "*Iniciar creación de una copia de seguridad*" y le asignamos de nombre al backup "backup-prueba" e iniciamos la copia de seguridad. Al cabo de un par de minutos finaliza satisfactoriamente como se puede comprobar a continuación:

Figura 69. Realización satisfactoria de la copia de seguridad.

6.2 Plan de restablecimiento.

Un plan de copias de seguridad y un plan de restablecimiento son dos componentes complementarios de una estrategia integral de gestión de datos y recuperación ante desastres.

El *plan de restablecimiento* tiene como objetivo restaurar los datos a un estado funcional después de un incidente. Este plan entra en acción después de que se produce una pérdida de datos o una interrupción del servicio. Define los pasos específicos para recuperar los datos de las copias de seguridad y restaurar los sistemas en su estado operativo.

A modo resumen, el plan de copias de seguridad es preventivo y asegura que los datos importantes se copien y almacenen de manera segura. Por otro lado, el plan de restablecimiento es reactivo y se utiliza cuando es necesario recurrir a esas copias de seguridad para restaurar la información y los sistemas después de un incidente. Ambos son necesarios para una estrategia robusta de protección de datos.

Para crear un plan de restablecimiento debemos ir a “Crear Plan de Restablecimiento” y darle clic para configurar las opciones de implantación:

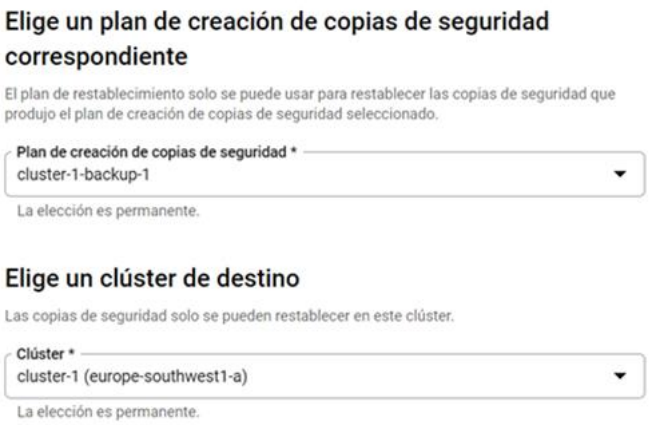

Figura 70. Crear un plan de restablecimiento.

Con ello, damos por finalizado este punto, que nos permite ante un desastre o una contingencia, recuperar el clúster de GKE y Anthos además de la aplicación desplegada volviendo a la normalidad en pocos minutos.

7. Conclusión y futuras direcciones.

En este Trabajo Final de Grado titulado "*Orquestación de Contenedores en un Entorno Linux con GKE, Athos y Google Cloud*", se ha dedicado una atención especial a entender como tecnologías en la orquestación de contenedores pueden optimizar la gestión de aplicaciones en un entorno nube. La elección de Google Kubernetes Engine (GKE), Anthos y Google Cloud, corriendo bajo un entorno Linux, no fue casual; estos representan la vanguardia de la tecnología para este tipo de servicios y ofrecen una combinación única de flexibilidad, robustez y eficiencia.

La exploración detallada de estas herramientas reveló su capacidad para, no sólo manejar con eficiencia la distribución y el escalado de aplicaciones, sino que también para integrar diversas operaciones y servicios en una única arquitectura nube. Este estudio demuestra que la integración GKE, Anthos y Google Cloud es capaz de ofrecer soluciones sólidas y eficientes, particularmente en arquitecturas de nuble complejas y distribuidas.

En el transcurso del proyecto, nos enfrentamos a desafíos inherentes a la creación, configuración y administración de un clúster de Kubernetes, así como la integración de Anthos. La superación de estos obstáculos no sólo requirió una planificación y una experimentación práctica, sino también un estudio de la bibliografía existente y que se detalla en el punto siguiente. Este proceso ha permitido, además de resolver los problemas técnicos, ganar una comprensión profunda de las implicaciones prácticas y teóricas de estas tecnologías en el mundo real.

Mirando hacia el futuro, la integración de GKE, Anthos y Google Cloud con tecnologías emergentes como la inteligencia artificial (IA) y el aprendizaje automático (Machine Learning, ML) presenta un panorama emocionante. Esta sinergia tiene el potencial de abrir nuevos caminos en la automatización de las decisiones y la eficiencia operativa en entornos nube, especialmente en los que respecta a la adaptabilidad y la respuesta en tiempo real a las necesidades cambiantes del mercado y del entorno operativo.

Además, este proyecto sienta las bases para investigaciones futuras, dónde se podrían realizar estudios comparativos con otras plataformas de orquestación y nube. Estos estudios serían cruciales para entender las diferencias de rendimiento, capacidad y flexibilidad entre las diversas herramientas disponibles en el mercado. Tal investigación podría iluminar las fortalezas y las debilidades de cada plataforma, además de ofrecer una guía sobre la mejor elección para diferentes escenarios y necesidades empresariales.

En una nota personal, el desarrollo de este proyecto ha sido una travesía enriquecedora y reveladora. No solo he obtenido una comprensión profunda de la orquestación de contenedores con GKE y como es el despliegue de una aplicación en un entorno nube, sino que también he apreciado las ventajas significativas que estas tecnologías ofrecen sobre los escenarios de virtualización tradicionales. Esta experiencia ha sido fundamental para mi crecimiento profesional en el campo de las tecnologías de la información y me ha dota de conocimientos y habilidades para un mejor entendimiento y desempeño en mi posición laboral.

8. Bibliografía

[1] Burns, B., Beda, J., & Hightower, K. (2020). "Kubernetes: Up & Running: Dive into the Future of Infrastructure". O'Reilly Media.

[2] Google Cloud. (2023). "Google Kubernetes Engine Documentation".

[3] Google Cloud. (2023). "Anthos Documentation".

[4] Smith, A., & Johnson, B. (2022). "Efficient Container Orchestration in Cloud Environments: A Comparative Study". Journal of Cloud Computing, 15(3), 123-145.

[5] Google Cloud. (2022). "Whitepaper on Advanced Container Management with GKE".

[6] Doe, J. (2021). "Optimizing Cloud Performance with Anthos".

[7] Miller, R. (2023). "Best Practices for Managing Containers in Linux Environments".

[8] Lee, C., & Kim, D. (2022). "Implementing a Scalable Cloud Architecture Using Google Kubernetes Engine: A Case Study". Enterprise Cloud Solutions, 10(2), 234-250.

[9] Kubernetes Community. (2023). "Discussions on Kubernetes Deployment Strategies".

[10] García, E. (2021). "Orchestration of Microservices in Cloud Environments Using Kubernetes".

[11] International Cloud Computing Standards Committee. (2022). "Standards for Container Orchestration in Cloud Computing".

[12] Smith, A., & Johnson, B. (2022). "Efficient Container Orchestration in Cloud Environments: A Comparative Study". Journal of Cloud Computing, 15(3), 123-145.

[13] Google Cloud. (2022). "Whitepaper on Advanced Container Management with GKE".

[14] Doe, J. (2021). "Optimizing Cloud Performance with Anthos".

[15] Coulouris, G., Dollimore, J., & Kindberg, T. (2019). "Distributed Systems: Concepts and Design". 6th Ed. Pearson Education.

[16] Linux Foundation. (2023). "Linux Kernel Documentation".

[17] Nguyen, T., & Schreiber, A. (2021). "A Study on the Performance of Docker vs. Kubernetes". Journal of Systems Architecture, 18(1), 77-88.

[18] Intel Corporation. (2022). "Optimizing Container Workloads on Intel Architecture".

[19] Rodríguez, M. (2022). "Advanced Techniques in Container Orchestration".

[20] O'Connor, L. (2023). "Challenges and Solutions in Cloud Container Management".

[21] Chen, X., & Zhao, Y. (2022). "Cloud Migration: From Legacy Systems to Containerized Solutions". Case Studies in Cloud Computing, 11(4), 442-460.

[22] Cloud Native Computing Foundation. (2023). "Kubernetes Deployment Strategies: Community Insights".

[23] Smith, K. (2022). "Container Orchestration in Multi-cloud Environments". Doctoral Dissertation, Massachusetts Institute of Technology.

9. Agradecimientos

Sin más dilación, quiero finalizar este trabajo final de grado agradeciendo a todos y a cada una de las personas que, durante la realización de estos estudios, he tenido la oportunidad y el inmenso placer de poder colaborar con ellas.

A mis padres, a ellos que son a quienes se lo debo todo en esta vida. Por ellos brindo tras finalizar mi quinto título universitario. Gracias por el apoyo, comprensión y por estar siempre a mi lado. Quizás siga estudiando, pero la rama de la tecnología no será una de ellas.

A mi novia Lorena, por entender que hay que estar unidos para enfrentarse a los retos de la vida. El desarrollo personal es uno de ellos, por lo que gracias por estar ahí y animarme siempre.

A todos mis consultores y profesores colaboradores, agradeceros las indicaciones y consejos recibidos a lo largo de todo este tiempo en el estudio del grado de ingeniería informática. He aprendido mucho y me llevo conmigo buenos momentos en los que disfrutado mucho en la realización de estos estudios.

A la gente que siempre ha creído en mí en mis diferentes trabajos. Gracias a todos por la oportunidad laboral que se me brinda en este año 2024. Eternamente agradecido por la confianza depositada en mí y espero estar a la altura de las expectativas.

Y como es costumbre en mis cuatro trabajos anteriores, no quería finalizar este sin incluir la frase que siempre finiquita el apartado de agradecimientos. La frase de Carlos Cortez, una frase que demuestra que, con esfuerzo, dedicación, constancia y voluntad cualquier objetivo es alcanzable:

"El éxito se alcanza convirtiendo cada paso en una meta y cada meta en un paso".

Gracias a todos y hasta siempre.

Víctor Buján Fernández