

Gestión de una arquitectura de microservicios con Istio

Autor

Raül de Arriba Garcia

rauldearriba@gmail.com

Abstract

El proyecto consiste en el estudio de unos casos de uso que nos permite hacer Istio junto con Kubernetes. Para poder realizar estas pruebas, se llevará a cabo la creación de un pequeño cluster virtualizado con Virtualbox y con Kubernetes como orquestrador de los contenedores Docker que contendrán los microservicios que gestionaremos con Istio. Todo esto con el objetivo de tener un entorno donde poder desplegar microservicios y gestionarlos de forma independiente.

Keywords— Infraestructura, PaaS, Kubernetes, Istio, Docker, Microservicio

Abstract

The project consists in the study of a use case that allows us to do Istio together with Kubernetes. In order to make these tests, the creation of a small cluster virtualized with Virtualbox will be carried out with Kubernetes as an orchestrator of the Docker containers that will contain the microservices that we will manage with Istio. All of this is made with the objective to have a infrastructure where you can deploy microservices and manage them independently.

Keywords— infrastructure, PaaS, Kubernetes, Istio, Docker, microservices

I. INTRODUCCIÓN

En los últimos años, se escucha por todos los medios de comunicación, y cada vez con más frecuencia, términos como Cloud [2], Big Data [3], Blockchain [4] o microservicios [5]. Estos son campos muy diferentes entre sí, pero todos ellos relacionados y con características en común: todos comparten que no pueden realizarse en un único ordenador personal, necesitan un cluster de máquinas para poder trabajar y realizar sus objetivos. No tenemos un ordenador suficientemente potente que por si solo opere con Petabytes de tamaño (Big Data), ni métodos de verificación únicos para validar los datos de una Blockchain por un único individuo.

Además, cada vez más las medianas y grandes empresas están dejando atrás la idea de tener un gran centro de procesamiento de datos (CPD) propio por motivos económicos. Disponer de tu propio CPD conlleva un gran gasto en electricidad, mantenimiento de los equipos informáticos, personal

de mantenimiento del CPD, mantenimiento y seguridad del edificio, dificultad para asegurar la alta disponibilidad y gran inversión en climatización del espacio.

Este cambio de mentalidad, se ha visto favorecido gracias a grandes empresas como Amazon con AWS, a Google con Google Cloud o a Microsoft con Azure, que han hecho posible llevar grandes empresas a la nube como es el caso de Apple y su iCloud que está parcialmente en la plataforma Amazon S3 [1].

Por lo comentado anteriormente, y el interés que ha despertado en mí este nuevo método de organización de las empresas, con este proyecto pretendo profundizar más en el campo y poder comprender a más bajo nivel el funcionamiento de estas infraestructuras para conocer su funcionamiento, ventajas e inconvenientes.

Por ese motivo, el presente documento tratará de mostrar todo proceso del proyecto. En la Sección

II se expondrán los objetivos del proyecto. En la Sección III se explicará la metodología utilizada para su desarrollo. En la Sección IV se mostrará la planificación del proyecto para terminarlo en los plazos definidos. En la Sección V se presenta el estudio de mercado de todas las tecnologías escogidas para el desarrollo del proyecto. En las Secciones VI y VII se presenta el escenario que se va a crear y los procesos instalación y configuración del sistema para poder conseguir una plataforma operable, gestionable y que microservicios y como se desplegan gracias a Kubernetes. En la Sección VIII se mostrara la parte del proyecto automatizada. En la sección IX se mostrará la solución de monitorización empleada para el proyecto. En la Sección X se mostara las funcionalidades utilizadas de GitHub y la estructuración del repositorio. En la Sección XI se presentará Istio y se mostrará como se gestionarán las comunicaciones entre los servicios desplegados en la plataforma para garantizar su correcto funcionamiento y poder conocer las intercomunicaciones de los servicios a partir de los casos de uso y en la Sección XII se expondrán reflexiones que habria que tener en consideración antes de llevar este proyecto a un estado de producción. Finalmente se mostaran en las Secciones XIII y XIV los resultados y conclusiones del proyecto.

II. OBJETIVOS

Este proyecto está planteado en objetivos acumulativos donde se tratara de ver un caso de uso inicial básico y progresivamente se amplía la profundidad del caso de uso a estudiar hasta cumplir el objetivo más complejo de los planteados.

- **Creación de una plataforma con kubernetes e Istio.** El primer objetivo consistirá en realizar un estudio de mercado entre las diferentes posibilidades que nos da el mercado actual en entornos cloud y realizar la instalación y configuración de las tecnologías necesarias para poder atacar los diferentes casos de uso.
- El primer caso de uso con Istio, **consistirá en limitar las comunicaciones entre los microservicios desplegados en el cluster.**

En un cluster, las comunicaciones entre los diferentes servicios son todas, con todas, como muestra la imagen izquierda de la figura 1. El segundo objetivo es conseguir limitar estas comunicaciones entre servicios utilizando la

tecnología Istio, como se muestra en la representación de la derecha de las comunicaciones del cluster.

- El tercer objetivo consistiría en **ampliar el caso de uso anterior aumentando en 2 el número de clusters y realizar comunicaciones entre ellos.**

Consistirá en ampliar el caso de uso y estudiar como se comporta Istio al limitar las comunicaciones en un entorno multicluster.

- El último caso a estudiar consistirá en tener **una serie de microservicios creados por nosotros y tratar de tener un diagrama de comunicaciones generado por Istio de forma automática.**

En este último caso buscaremos estudiar si Istio nos permite conseguir un diagrama de todas las comunicaciones dentro del cluster para llegar al punto de granularidad de, en el caso de detectar un problema, saber como se propagaría la comunicación.

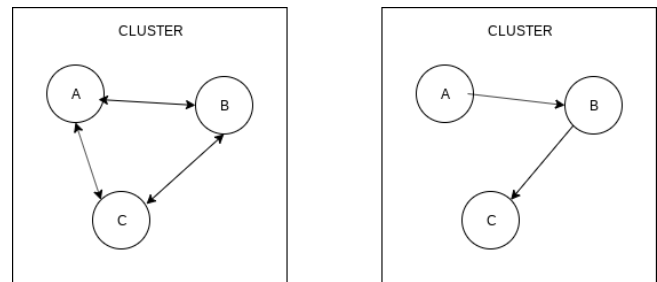


Fig. 1. Ejemplo de comunicaciones en un cluster (a la izquierda todos con todos y a la derecha el primer caso de uso)

III. METODOLOGÍA

Para este proyecto utilizaremos una metodología del modelo de cascada [6] por el hecho de que este proyecto esta pensado de forma iterativa. Empieza con la creación del entorno, un caso de uso poco complejo y a medida que se cumple el objetivo, se aumenta el nivel de granularidad de lo que Istio te permite lograr.

IV. PLANIFICACIÓN

La planificación de este proyecto la comenzamos realizando con una herramienta online, llamada monday.com, pero después de un tiempo de uso, descubrí que no es gratuita y tiene un coste mensual, por ese motivo trasasé toda la planificación a

canva.com. No es un software pensado exclusivamente para planificar, pero permite hacer diagramas de Gantt de manera fácil y muy visual. La planificación la he dividido en 4 fases: la primera fase de planificación y estudio de mercado, la segunda dedicada a la instalación y configuración del entorno base y el primer caso de estudio, la tercera donde introducimos microservicios propios y estudiamos la comunicación entre ellos, y finalmente una última fase donde aumentamos el número de clusters en 2 y estudiamos las comunicaciones entre ellos.

V. HERRAMIENTAS

En este apartado, mostaremos las herramientas y tecnologías utilizadas para llevar a cabo la realización de todo este proyecto después de estudiar las diferentes opciones disponibles. Dividiremos esta sección en herramientas organizativas y de control de versiones, en las herramientas para la plataforma y en las herramientas utilizadas dentro de la plataforma.

Las soluciones utilizadas para la parte organizativa y control de versiones son las siguientes:

- **canva.com:** Este software online que consiste en un marco para crear composiciones de imágenes lo utilizo para crear la planificación y el diagrama de Gantt por su fácil y rápido aprendizaje, toda la aplicación funciona arrastrando elementos al sitio deseado. Tiene la ventaja de ser muy fácil de compartir con otras personas de descargar sin marcas de agua y gratuito.
- **Github:** A nivel de control de versiones hay diferentes posibilidades como Bitbucket o Gitlab, pero finalmente me decanté por Github principalmente por la comodidad que me suponía al llevar bastante tiempo utilizándola a diario.

Github es una plataforma de desarrollo que te permite hostear código, mantener un control de versiones del mismo, mantener organizados tus proyectos, poder trabajar desde cualquier ordenador y compartir código con otros desarrolladores. Además, Github tiene funcionalidades menos conocidas como la sección de Issues y la wiki.

Por estas 2 funcionalidades, que no he utilizado hasta ahora, me decanté por ellas para poder investigar y aprender a utilizarlas para conseguir una buena documentación de este proyecto.

En el repositorio del proyecto en la wiki (<https://github.com/Radega1993/ProjectZero-TFG/wiki>) están todos los comandos y explicaciones de las etapas del proyecto.

El segundo bloque de herramientas son las utilizadas para la creación de la plataforma.

- **Virtualbox:** Esta ha sido la decisión más complicada de todas ya que es la base del proyecto, sobre esta elección se hará todo. Las opciones aquí eran muchas. Entre ellas he estudiado Amazon Web Services, Google Cloud, Microsoft Azure, IBM Cloud Private y RedHat OpenShift.

Todas estas opciones tienen el mismo problema, todas son de pago por uso.

Tienen un plan de prueba gratuito, pero como es un proyecto de investigación y aprendizaje no me quise arriesgar a no tener tiempo suficiente para desarrollar todo el proyecto. El segundo motivo de no escoger ninguna de las anteriores es por aprender a instalar y configurar un cluster desde 0. Por este segundo motivo se abren 2 opciones. La primera, crear un cluster de Raspberry Pi pero volvemos a tener el problema económico, cada una de ellas está entorno a los 40€ y para empezar el proyecto necesitaríamos mínimo 3 para 1 master y 2 nodos workers. Esto sin hacer un capacity planning[7] y tener una previsión de uso de todo los deployment que tendrá nuestra plataforma.

Finalmente la opción escogida ha sido virtualbox porque cumple todas las necesidades para este proyecto. Primero de todo es gratuito y Open Source. La facilidad de escalabilidad horizontal[8] al tener la posibilidad de crear tantos nodos como nuestro ordenador sea capaz de soportar.

- **Kubernetes:** Kubernetes[9] o k8s es un proyecto open source creado por Google. Es un orquestrador de contenedores.

La función principal de k8s es automatizar la implementación, el escalado y la administración de aplicaciones en contenedores.

- **Docker:** Se ha escogido docker como opción de contenerización de aplicaciones por los siguientes motivos: El primero es la integración con k8s, el segundo es el amplio soporte de la comunidad, la extensa documentación que hay

en internet, y finalmente, porque a nivel profesional, es la solución más utilizada actualmente como vemos en la figura 2, todo y que esta en declive desde que el 14 de noviembre de 2019 parte de la empresa fuera absorbida por Mirantis [11].

- **Python:** LO PONDRE SI FINALMENTE LO ACABO UTILIZANDO
- **Ansible:** Es una tecnología ampliamente utilizada en entornos cloud. Permite, de forma rápida y segura, automatizar aplicaciones e infraestructura. En nuestro proyecto lo utilizaremos para automatizar la creación y configuración del cluster, tanto de los nodos Masters como de los Workers. De esta forma, en caso de necesitar mas máquinas, se podran configurar, instalar y introducir al cluster de forma automatica [12].
- **Istio:** Es una solución que se integra junto a k8s y permite conectarse, proteger, controlar y observar los servicios que están funcionando dentro de la plataforma. Entre todas las funcionalidades que esta herramienta nos permite hacer, nosotros, nos vamos a centrar principalmente en el manejo de tráfico para estudiar diferentes casos de uso.

El tercer bloque de herramientas utilizadas en este proyecto son las aplicaciones que están corriendo dentro de la plataforma.

- **Prometheus:** Esta tecnología open-source de monitorización y alerta de eventos se encarga de registrar metricas a tiempo real mediante unos exporters y permite tener una vision del estado del sistema en todo momento.[13]
- **Grafana:** Es una una solución open-soure para analizar y monitorizar a través de dashboards la información recogida en una base de datos y mostrar estos graficos a tiempo real. Su integracion con prometheus es muy habitual y la utilizaremos como solucion de monitorización de la plataforma.
- **Kiali:** Esta tecnología propia de Istio te permite ver dashboard y graficos de los servicios del cluster a demas de ver y editar los yamls de configuración de Istio

VI. ESCENARIO PROPUESTO

En esta sección veremos el escenario que montamos y los procesos de instalación del mismo hasta conseguir un entorno operativo.

A. Prerequisitos

Para poder hacer este proyecto, necesitamos los siguientes requisitos: Un servidor, multiples servidores o un ordenador con 8 cores, 16GB de RAM y con capacidad para mantener minimo 3 maquinas virtuales(MV), un software de virtualización como VMWare o Virtualbox y la iso de Centos7 como base de sistema operativo para las MV.

B. Estado inicial

Como base donde alojar todo este proyecto, en mi caso, utilizaré un ordenador DELL XPS 15 con un sistema operativo con un procesador Intel I7 de 7 generación y 16 GB de RAM. Para la parte de virtualización, decidí utilizar Virtualbox por ser un software gratuito y que cumpliera todos los requisitos para hacer las máquinas virtuales. Virtualbox nos permite configurar los cores y la RAM de cada maquina Virtual, y nos permite crear redes para comunicar las MV entre ellas.

VII. INSTALACIÓN

En este apartado, prodecemos a instalar y configurar el cluster de kubernetes e Istio sobre el escenario mostrado anteriormente.

Antes de poder empezar a instalar y aplicar la configuración de cada una de las máquinas, crearemos una imagen base que la clonaremos tantas veces como maquinas queramos en el cluster. Esto lo hacemos para asegurar que todas las máquinas que componen el cluster sean homogeneas entre ellas, serán la misma imagen base y con ello las mismas versiones de sistema operativo.

A. Máquina Base

Para instalar la máquina base, con la ayuda de virtualbox, en nuestra maquina host, crearemos una nueva maquina virtual i la configuraremos con una capacidad de 40GB de disco duro, le daremos 2 cores de cpu y 4 GB de RAM, que son requisito de kubernetes. Una vez creada la MV le instalaremos el sistema operativo Centos7 sin modo gráfico, ya que no es necesario, y ganaremos espacio en el disco que podremos utilizar con aplicaciones mas útiles para la plataforma. Una vez ya tengamos Centos7 completamente operativo en nuestra MV, procedemos a crear la red Nat con la que comunicaremos todas las máquinas del cluster. En las preferencias de red,



Fig. 2. comparación entre Docker[azul], LXD[rojo] y Podman[amarillo] (Google trends)

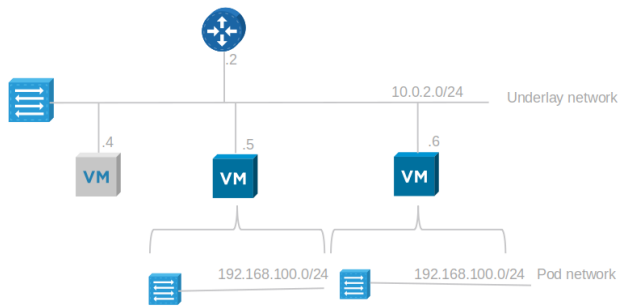


Fig. 3. Diagrama de red del cluster del proyecto

crearemos una nueva red NAT. Yo la he llamado TFGfarm y he decidido utilizar el rango privado de clase A 10.0.2.0/24.

B. Redireccionamiento de puertos

Este subapartado dentro de la instalación y configuración solo es necesario para trabajar en máquinas virtuales: Para acceder desde la máquina host al cluster vamos a necesitar hacer una redirección de puertos. Esto se debe a que la IP que tienen nuestras MV son privadas, asignadas por Virtualbox que actúa como DHCP. Si no hicieramos estos redireccionamientos de puertos no podrías acceder a determinados servicios desde nuestro host como por ejemplo SSH o servidores web como Apache o Nginx[16].

C. Master

Una vez tenemos creada y configurada la máquina base y la red Nat para nuestro entorno procedemos a clonar esta MV y le ponemos el nombre de "Master" para poder identificarla. Dentro de la MV master le asignamos la IP estatica que hemos definido en nuestro diagrama de red en la figura 3 y le definimos en el fichero de la ruta '/etc/sysconfig/network-scripts/ifcfg-enp0s3' la IP 10.0.2.4/24 y las demas

configuraciones de red como son la gateway, el DNS y el protocolo a estatico.

Se puede encontrar toda la configuración mas detallada en la wiki del proyecto en el apartado de instalación [17]. Definida una IP estatica para nuestro master y para todas las maquinas que compongan el cluster, porque necesitamos conocer a las máquinas para poder acceder y gestionarlasy, si las IPs cambiaran por el DHCP no seria viable gestionar esta clase de entornos. A continuación, para facilitar el acceso entre máquinas del cluster, definiremos en el /etc/hosts todas las maquinas de nuestro cluster por IP y nombre. Así, para el ser humano, será mas sencillo recordar o identificar una máquina llamada Worker1 que saber quién es 10.0.2.X.

Una vez terminada la configuración inicial del master, vamos a instalar los servicios necesarios para crear el cluster de k8s.

Definimos el SELinux en modo permisivo.

SELinux es un sistema de control de acceso obligatorio donde antes de una llamada al sistema, el kernel pregunta a SELinux si el proceso esta autorizado a realizar esa operación o no[18]. Los servicios que necesitamos instalar en el nodo master son:

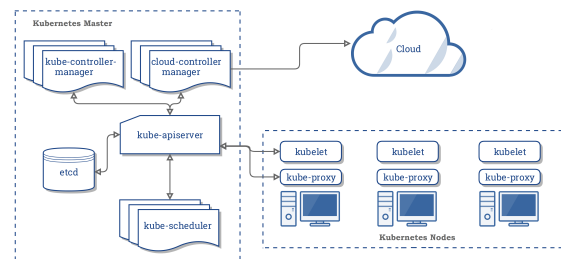


Fig. 4. componentes de un cluster k8s

- **kubelet** Se encuentra en cada nodo del cluster y su función principal es la de crear una comunicación con la API de kubernetes y informar

del estado de ejecución de cada nodo y de asegurar los contenedores que estan corriendo en cada pod.

- **kubeadm** Es una herramienta de kubernetes que nos permite desplegar de manera más sencilla un cluster de kubernetes. Esta herramienta contiene todas las instrucciones necesarias para crear un cluster.
- **kubectl** Es la herramienta de linea de comandos que nos permite controlar todo el cluster.
- **Docker** Es la herramienta de contenerización que he decidido utilizar para el proyecto, este se encarga de crear contenedores con la aplicación dentro y así permitir ser orquestrado por kubernetes.

Finalmente, antes de iniciar nuestro cluster de kubernetes, debemos desactivar el swap. Esto lo hacemos porque kubernetes tiene la posibilidad de limitar los recursos que puede consumir un pod dentro del sistema. Si no se deshabilita esta opción de la configuración de linux, si una aplicación supera el limite de memoria RAM definido, este acabara por llenarnos la memoria swap del sistema. Deshabilitamos el firewall para evitar problemas de conexión dentro del cluster y iniciamos el cluster con el comando `kubeadm`, con los parámetros de la IP de la API de kubernetes que se encuentra en el nodo master (como vemos en la figura 4) y el rango de IPs de los pods que serán las IPs que recibirán las aplicaciones que esten en ejecución dentro de la plataforma. Si todo se ha ejecutado correctamente, por pantalla nos mostrara un mensaje "successfully", un grupo de 3 comandos para copiar y dar permisos al fichero de configuración de k8s y el comando de `kubeadm join` con el token que deberemos guardar porque este se tendrá que ejecutarlo en los nodos workers para que formen parte del cluster.

En este punto ya tenemos nuestro cluster de kubernetes casi operativo. Solo faltaria desplegar nuestro primer pod que será Calico y el segundo que sera MetalLB. Calico es una solución open-source de red y seguridad. En la red, que se encargara de crearnos los tuneles de comunicación dentro de nuestro cluster en todas las aplicaciones que estemos corriendo dentro de la red de pods que hemos definido al iniciar el cluster. MetalLB es una solución Open-Source de balanceador de carga para clusters de kubernetes no desplegados en plataformas, como Amazon o Google, que te lo configuran

por defecto en la instalación. MetalLB nos asigna una IP del rango que le hemos configurado al servicio desplegado para poder acceder a él desde fuera del cluster.

D. Istio

Istio lo instalaremos en el nodo master del cluster y este lo veremos como un pod más corriendo en k8s en su namespace 'istio-system'. La instalación base la realizaremos descargando el script que nos proporcionan los desarrolladores de la solución. Este script se encargará de descargar todos los ficheros y directorios necesarios para poder lanzar istio en nuestro cluster. Una vez terminada la ejecución de este script ejecutaremos, con el comando `kubectl`, todos los ficheros YAML que se encuentran en la ruta 'istio-[version]/install/kubernetes'. A continuación debemos configurar la herramienta `istioctl` para poder utilizar todas las herramientas y con esto finalizaríamos la instalación completa de nuestro primer nodo del cluster, el nodo master.

E. Workers

Una vez tenemos el nodo Master operativo, debemos instalar, configurar y agregar los workers al cluster para que puedan correr aplicaciones. Primero de todo hemos de hacer igual que en el master. Clonaremos la imagen base y configuraremos la IP estática y añadimos los nodos al `/etc/hosts`. A continuación deshabilitaremos el SELinux, el firewall y el swap de la máquina y habilitamos el `ip forward` para permitir que la red de pods que habra en los workers y tengan salida hacia los demás nodos del cluster. Con el nodo configurado, instalamos `kubeadm` para que nos permita utilizar el comando de `kubeadm join` más adelante y `docker` que será la herramienta con la que contenerizaremos las aplicaciones. Ahora lanzamos el comando que guardamos cuando iniciamos el cluster en el nodo master y nuestro worker pasará a formar parte del cluster.

VIII. ANSIBLE

En esta sección, procedemos a mostrar la automatización disponible en la plataforma.

Para este proceso se ha utilizado Ansible por su rápida curva de aprendizaje, su gran potencial y que es una de las soluciones más utilizadas en el mundo. [12]

La estructura escogida para automatizar el proyecto ha sido la siguiente:

En la raíz encontramos un fichero donde están definidos todos los nodos del cluster y un fichero de setup que genera una llamada al playbook que realizará la automatización del proceso. Este playbook contiene la llamada a todos los roles necesarios para realizar la tarea de forma automática. Finalmente, encontramos un directorio "roles" donde está definida la implementación de cada tarea necesaria para su automatización.

IX. MONITORIZACIÓN

En este apartado se explicará la solución de monitorización decidida para la plataforma. Después de plantear el hacer nuestro propio sistema de "monitorización" básico con un script en python que comprobará cada minuto el estado de los nodos del cluster y de los pods que están desplegados en él, descarté esta opción y me decidí a utilizar la dupla de aplicaciones de monitorización estándares, que además se integran con Kubernetes e Istio a la perfección.

Por un lado tenemos Prometheus, software de recogida y procesamiento de métricas y alertas, donde hemos definido, entre otras, una alerta de ping a los nodos para saber en todo momento si perdemos la comunicación dentro del cluster.

Por el otro lado tenemos Grafana, esta aplicación recoge los datos de las métricas de Prometheus y con esos datos se crean dashboards que se actualizan a tiempo real. Esto nos permite ver el estado del sistema a cada momento.

Finalmente, disponemos de Kiali en la plataforma, que es una aplicación de Istio para monitorizar la red y poder ver y editar algunos ficheros de configuración a parte de crearlos automáticamente un grafo de comunicaciones de los servicios.

X. GITHUB

En este apartado se explica el uso que se le ha dado a Github durante la realización del proyecto.

Github es el software de control de versiones de Microsoft. Para el proyecto se creó un repositorio llamado "ProjectZero-TFG-". En este repositorio podemos encontrar lo siguiente:

Los ficheros YAML utilizados para el despliegue de las aplicaciones que corren en nuestro cluster, los ficheros ansible para la automatización de algunos

procesos del cluster y el paquete de Istio para su despliegue dentro del cluster.

Además, se ha utilizado 2 funcionalidades extras de la plataforma Github. La primera y más importante, ha sido la función de wiki[15], donde se ha realizado toda la documentación del proyecto en lenguaje markdown. La segunda la funcionalidad de "Issues" donde se han ido publicando diferentes problemas encontrados durante la realización del proyecto y su solución una vez encontrada.

XI. CASOS DE USO

En este apartado se explicarán los diferentes casos de uso realizados para estudiar el funcionamiento de Kubernetes y de Istio y poder ver todo su potencial.

A. caso 1

El primer caso de uso vamos a probar como funciona el enrutamiento proporcionado por Istio. [20]

Para desarrollar esta primera prueba de concepto, utilizaremos una aplicación que nos proporciona Istio que consiste en una plataforma para ver reviews de diferentes libros.

La aplicación tiene 6 microservicios. Una página principal donde se muestra toda la información, tres versiones diferentes de las reviews, una sin estrellas, otra con estrellas negras y una última con estrellas rojas, un microservicio con los detalles que contiene la información del libro y por último ratings que contiene la información del número de estrellas del libro.

Para poder probar el enrutamiento, hemos de conocer 3 conceptos de Istio.

- **Subset:** Sería más o menos como un grupo o namespace, es un subconjunto de pods de un mismo servicio.
- **DestinationRule:** Son las reglas que tiene que cumplir el tráfico enrutado.
- **VirtualService:** Son los enrutadores que se encargan de dirigir el tráfico hacia el destino y que cumplan las reglas definidas en las destination rules.

Una vez tenemos la aplicación funcionando en el cluster definiremos los subsets dentro de las DestinationRules y finalmente crearemos los virtual-services donde definiremos cada servicio que subset y regla tiene que cumplir.

Gracias a estas funcionalidades que nos ofrece Istio podemos enrutar el tráfico todo a diferentes microservicios, podemos enrutarlo por campos del header de la petición http (podemos redirigir el tráfico dependiendo del navegador o del usuario logeado) o podemos balancear la carga entre las diferentes versiones. [15]

B. caso 2

El segundo caso de uso consiste en tener 2 clusters y poder comunicar los servicios entre ellos. No se ha podido realizar este caso de uso por falta de recursos. Todo el proyecto se ha realizado sobre un ordenador personal y este no tenía potencia suficiente para desplegar 6 máquinas virtuales todas trabajando a la vez y cada una de ellas exponiendo servicios. El ordenador se queda bloqueado y no se ha podido probar esta funcionalidad.

C. caso 3

EXPLICAR CASO DE USO 3

XII. SIGUIENTES PASOS

En este apartado se expondrán algunas consideraciones que no se han podido realizar, pero que si un sistema como este se quisiera llevar a producción habría que tener presente antes de desplegarlo.

Lo primero serían los recursos, este entorno debería desplegarse en un cluster de servidores con memoria RAM, CPU y disco acorde con lo que deba soportar la plataforma.

Por otro lado, habría que disponer de 2 clusters idénticos pero en diferentes ubicaciones para garantizar la alta disponibilidad y así poder evitar pérdida de servicio y datos.

Se debería plantear una opción de registro de logs como elasticSearch y kibana para tener un control interno del cluster y poder detectar errores o fallas del sistema.

Finalmente, habría que añadir, a la parte de monitorización, una solución como alertmanager que te permita enviar las alertas de Prometheus por correo o algún canal de comunicación en caso de fallo para su rápida actuación en caso de ser necesaria.

A medida que el proyecto crece, surgirán nuevas necesidades para la plataforma, pero estas son las acciones necesarias más críticas detectadas a lo largo del proceso de realización de este proyecto.

XIII. RESULTADOS

En este apartado compararemos los objetivos propuestos en el proyecto con el trabajo realizado para conseguirlos.

Tal y como hemos visto en la sección de creación de la plataforma, este no es un proyecto de creación de una aplicación sino que se trata de un proyecto de investigación, de una prueba de concepto, sobre como se creaba una plataforma a partir de la tecnología kubernetes e Istio y el resultado ha sido satisfactorio ya que hemos conseguido un entorno operativo, todo y el gran trabajo de investigación y aprendizaje requerido, ya que son tecnologías que utilizar una gran cantidad de conceptos vistos durante el grado, pero que la interconexión entre ellos queda muy difuminada o no se ven durante el grado porque son especializaciones diferentes. Gracias a este proyecto he podido ver como se relacionan herramientas como git o docker más orientadas a ingeniería del software. Conceptos de tuneles SSH y configuraciones de red y seguridad entre las aplicaciones de la mención de Tecnología de la Información y las comunicaciones y plataformas cloud de sistemas distribuidos.

Respecto a los casos de uso algunos si han sido posibles de realizar y otros no se han podido realizar o completar. El primer caso a estudiar una vez el cluster estuviera listo, consistía en poder limitar la comunicación entre los servicios.

Este caso de uso sí lo hemos podido realizar de manera satisfactoria. Conseguimos enrutar el tráfico según nuestras necesidades, hacer un servicio inaccesible y limitar el tráfico por usuarios, hemos podido ver que el manejo de tráfico que nos ofrece Istio es muy potente y no muy complejo de configurar.

El segundo caso de uso, no se pudo realizar por falta de recursos para desplegar 2 clusters de kubernetes funcionando paralelamente, no se ha podido estudiar el tráfico entre servicios disponibles en clusters diferentes.

EXPLICAR 3º CASO

XIV. CONCLUSIÓN

En este apartado se exponen las conclusiones extraídas de realizar este proyecto de fin de grado.

Durante el proyecto, he podido aprender nuevas tecnologías y reforzar conocimientos de casi todos los ámbitos del grado por la gran cantidad

de temas que se han estudiado en este trabajo. Por una lado, he podido aprender más a fondo el funcionamiento y tecnologías empleadas en los sistemas Cloud, sobretodo Kubernetes e Istio, que han sido toda la base de este trabajo. Además, he podido entender mejor el funcionamiento de una red y sus componentes, ya que, la potencia de un cluster reside en las interconexiones de los nodos y en especial, con Kubernetes, donde la comunicación de los microservicios para que puedan entenderse entre ellos y poder realizar las funciones deseadas, es uno de sus grandes potenciales.

Por otro lado, he podido reforzar lenguajes de programación como python, herramientas de control de versiones como git y todas las funcionalidades que ofrece la plataforma Github y por último, aprender el funcionamiento de Docker y la contenerización de aplicaciones para su despliegue en cualquier entorno.

XV. AGRADECIMIENTOS

En primer lugar, agradecer a todos los profesores que he tenido a lo largo de mi paso por el grado, ya que sin ellos, este proyecto habria sido algo casi imposible de realizar. Mención especial a mi tutor Jordi Casas por su seguimiento y disponibilidad para atenderme siempre a pesar de mi difícil horario y a Porfidio Hernández por ayudarme a orientar alguna parte del proyecto.

En segundo lugar, agradecer a mi familia y pareja por apoyarme durante el proyecto y leerlo muchas veces para dar su opinión sobre el estado del artículo.

Finalmente, agradecer a los compañeros de trabajo que me ayudaron a no quedarme estancado en el proyecto y poder avanzar gracias a sus recomendaciones y consejos durante todo el trabajo.

REFERENCIAS

- [1] JAVIER PASTOR. "Apple paga más de 30 millones de dólares al mes a Amazon por usar AWS: iCloud es en buena parte la nube de Amazon"
<https://bit.ly/2MpsIJX> Accedido en: Setiembre, 2019.
- [2] RedHat. "¿Qué son las nubes?"
<https://red.ht/2oSw8vX> Accedido en: Noviembre, 2019.
- [3] Oracle. "Definición de big data"
<https://bit.ly/36GFaNC> Accedido en: Octubre, 2019.
- [4] Pradip Kumar Sharma ; Mu-Yen Chen ; Jong Hyuk Park. "A Software Defined Fog Node Based Distributed Blockchain Cloud Architecture for IoT"
<https://bit.ly/2N1TcNc> Accedido en: Octubre, 2019.

- [5] RedHat. "¿Qué son los microservicios?"
<https://red.ht/2qp6kbp> Accedido en: Noviembre, 2019.
- [6] OBS Business school. "Gestión de proyectos siguiendo el modelo de cascada"
<https://bit.ly/2ByxnmZ> Accedido en: Octubre, 2019.
- [7] Wikipedia. "Capacity planning"
<https://bit.ly/2PQdHmN> Accedido en: Octubre, 2019.
- [8] Oscar Blancarte. "Escalabilidad Horizontal y Vertical"
<https://bit.ly/2NREQmy> Accedido en: Noviembre, 2019.
- [9] Oscar kubernetes. "Orquestación de contenedores para producción"
<https://kubernetes.io/es/> Accedido en: Noviembre, 2019.
- [10] Istio. "What is Istio?"
<https://bit.ly/33n9smt> Accedido en: Noviembre, 2019.
- [11] Mirantis "Mirantis acquires Docker Enterprise"
<https://tcrn.ch/2PMv8UR> Accedido en: Diciembre, 2019.
- [12] Ansible. "Ansible oficial"
<https://www.ansible.com/> Accedido en: Diciembre, 2019.
- [13] prometheus. "prometheus oficial"
<https://prometheus.io/> Accedido en: Diciembre, 2019.
- [14] grafana. "grafana oficial"
<https://grafana.com/> Accedido en: Diciembre, 2019.
- [15] Wiki. "wiki"
<https://bit.ly/2ZY6BzB> Accedido en: Enero, 2019.
- [16] Wiki. "wiki ssh"
<https://bit.ly/397vU6H> Accedido en: Octubre, 2019.
- [17] Raúl. "wiki instalación"
<https://bit.ly/2SihQ4e> Accedido en: Octubre, 2019.
- [18] Debian. "Introducción a SELinux"
<https://bit.ly/34MYS0e> Accedido en: Noviembre, 2019.
- [19] calico. "Calico oficial"
<https://bit.ly/392o0vc> Accedido en: Noviembre, 2019.
- [20] Istio. "Bookinfo Application"
<https://bit.ly/39KmEpo> Accedido en: Enero, 2020.