

DESPLIEGUE DE UN CLÚSTER DE KUBERNETES EN LA NUBE

**TRABAJO FIN DE GRADO
ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED**



IES JUAN DE LA CIERVA, MADRID

EDUARDO GAYO LOPEZ Y ALBA MOREJÓN GARCÍA

ÍNDICE

ÍNDICE	3
1. Resumen	4
2. Introducción	5
2.1. Marco del Proyecto	5
2.2 Motivación	5
2.3 Objetivos	6
3. Estado del arte	6
3.1 Introducción	6
3.2 ¿Qué es la contenerización?	6
3.3 ¿Por qué contenerizar en vez de usar máquinas virtuales?	7
3.4 Containerd	8
3.5 Kubernetes	8
3.6 Arquitectura de Kubernetes	10
3.6.1 Pods	10
3.6.2 Nodos	11
3.6.3 Comunicación entre Nodos y el “Control Plane”	14
3.6.4 Controllers	16
3.6.5 Leases	17
3.6.6 Container Runtime Interface (CRI)	18
3.7 Importancia y relevancia de Kubernetes y la computación en la nube	18
3.8 Razones para elegir Kubernetes en la nube	20
3.9 Ventajas de utilizar Kubernetes en comparación con otras soluciones	20
3.10 CNI	23
3.11 Relevancia en un entorno laboral actual	23
3.12 Servicios en la nube ¿El futuro de la computación?	24
3.13 IBM Cloud	26
4. Descripción del proyecto	27
4.1 Alcance del proyecto	27
4.2 Metodología a seguir	27
4.3 Herramientas y tecnologías a utilizar	28
5. Despliegue de un clúster de Kubernetes en local	28
5.1 Configuración de las máquinas	28
5.2 Instalación de Kubernetes	32
5.3 Despliegue y configuración de Kubernetes	40
5.4 Despliegue de aplicaciones en Kubernetes	45
5.5 Pruebas de alta disponibilidad	51
6. Despliegue del clúster de Kubernetes en IBMCloud	56
6.1 Configuración inicial	57
6.2 Despliegue y pruebas con clúster de IBM Cloud	62
7. Conclusiones	70
8. Webgrafía	71

1. Resumen

<p>El siguiente proyecto es un acercamiento al software de Kubernetes, explicando paso a paso cómo funciona la contenerización, sus diferencias con el uso de máquinas virtuales y las ventajas que ofrece. Para poder realizar esta tarea, comenzaremos desplegando tres máquinas virtuales para crear un clúster de Kubernetes en local donde podamos realizar toda clase de pruebas con los nodos, pods, contenedores y distintos despliegues. Crearemos una red de contenedores para que Kubernetes pueda comunicarse en sus nodos y haremos pruebas de alta disponibilidad y replicación. Una vez hayamos interiorizado los conceptos y cómo funciona este software, haremos un despliegue del clúster en la nube, utilizando IBM Cloud como IaaS para nuestras máquinas y observando las diferencias que hay entre los dos despliegues.</p> <p>Palabras clave: Kubernetes, contenedores, Docker, Virtualización, IBM Cloud, Clúster.</p>	<p>The following project is an approach to the Kubernetes software, explaining step by step how containerisation works, its differences with the use of virtual machines and the advantages it offers. In order to perform this task, we will start by deploying three virtual machines to create a local Kubernetes cluster where we can perform all kinds of tests with nodes, pods, containers and different deployments. We will create a container network so that Kubernetes can communicate on its nodes and we will test high availability and replication. Once we have internalised the concepts and how this software works, we will deploy the cluster in the cloud, using IBM Cloud as IaaS for our machines and observing the differences between the two deployments.</p> <p>Keywords: Kubernetes, Containers, Docker, Virtualisation, IBM Cloud, Cluster</p>
---	---

2. Introducción

2.1. Marco del Proyecto

El siguiente trabajo de Fin de Grado busca un acercamiento a nuevas tecnologías en alza a día de hoy, destacando por encima de todas, “Kubernetes”. La idea es aprender a desplegar un clúster de contenedores escalables manejados por kubernetes tanto, en un entorno local controlado como en un proveedor de virtualización.

Se desplegará una aplicación web sencilla en el clúster para comprobar su funcionalidad y escalado, así como herramientas de monitorización del clúster como Kubernetes Dashboard para comprobar los recursos que va utilizando y comprender a fondo cómo funciona esta herramienta.

2.2 Motivación

Hoy en día Internet se ha vuelto una herramienta necesaria para el día a día. Desde trabajar en remoto o a través de aplicaciones web hasta hacer la compra online o leer el periódico, cada vez más personas solicitan servicios en línea, y las empresas deben estar preparadas para atender esta demanda constante y que va en aumento. Teniendo en cuenta que esta tendencia no solo no va a cambiar, sino que va en aumento, nos parece muy interesante estudiar cómo las empresas afrontan estos problemas y buscan soluciones ingeniosas para poder ofrecer más con menos.

La contenerización es fundamental en las empresas para ahorrar costes en materia de servidores y poder ofrecer escalabilidad a los recursos en línea. Se trata de una tecnología en auge cuyo uso será fundamental en los próximos años, por lo que resulta indispensable conocer sus ventajas y debilidades frente a otras alternativas más tradicionales como son las máquinas virtuales.

2.3 Objetivos

El objetivo de este proyecto es ilustrar paso a paso cómo desplegar un clúster de kubernetes en un entorno controlado a modo de prueba, para después lanzar otro clúster funcional en una herramienta de virtualización online como es IBM Cloud.

Para alcanzar estos objetivos, deberemos entender lo siguiente:

- Cómo funciona la contenerización.
- Cómo funciona un gestor de contenedores como es Kubernetes.
- Cómo desplegar servicios en IBM Cloud.

3. Estado del arte

3.1 Introducción

Para realizar el estado del arte de la tecnología a utilizar, debemos realizar un acercamiento a otras tecnologías que van de la mano de Kubernetes, como es el caso de la contenerización o las Container Network Interfaces, así como a la propia tecnología de Kubernetes.

3.2 ¿Qué es la contenerización?

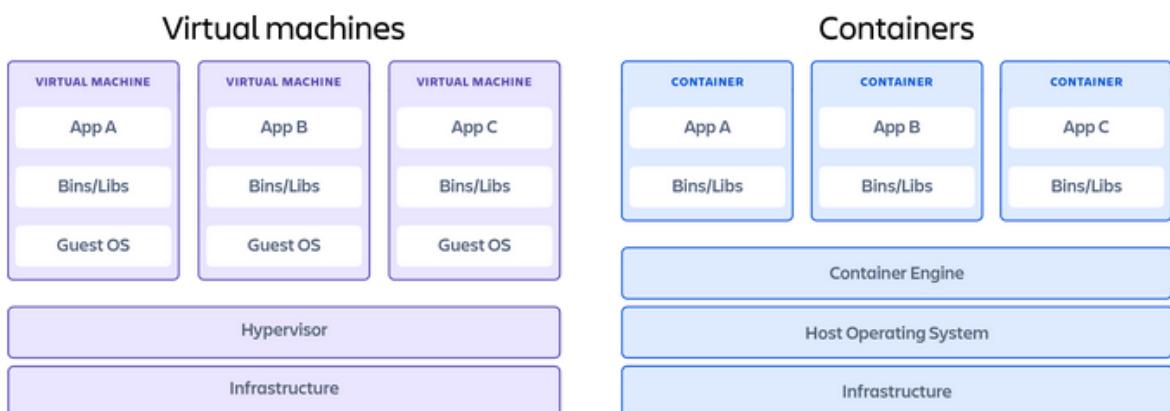
La contenerización consiste en el empaquetado de código software con solo las bibliotecas del sistema operativo y las dependencias necesarias para ejecutar el código, creando así un ejecutable ligero que se llama contenedor. Los contenedores son más portátiles y eficientes que las máquinas virtuales que hasta ahora se habían utilizado, convirtiéndose en la alternativa más utilizada frente a este servicio.

Hay que tener en cuenta que la contenerización permite desplegar aplicaciones de forma más rápida y segura que antes. Si pensamos en los métodos tradicionales de desarrollo de software, este, se desarrollaba en una plataforma (Windows, Linux, etc.) y, cuando se movía a una nueva ubicación con un S.O. diferente del utilizado en el desarrollo, solía provocar toda clase de errores y problemas. La contenerización soluciona este problema porque agrupa los archivos de configuración, las bibliotecas y las dependencias que son necesarias para la ejecución del software, lo que permite que se ejecute sin problemas en cualquier plataforma o nube.

El concepto de contenerización y aislamiento de procesos no es algo nuevo, existe desde hace décadas, pero fue la aparición del software de código abierto “Docker Engine” el que supuso una revolución en este campo, convirtiéndose en un estándar del sector de la informática.

3.3 ¿Por qué contenerizar en vez de usar máquinas virtuales?

El principal motivo es que los contenedores se denominan “ligeros”. Esto quiere decir que los contenedores comparten el kernel del sistema operativo de la máquina y no es necesario, como sí lo es en las máquinas virtuales, asociar sistemas operativos en la aplicación. Esto supone un ahorro de recursos que se pueden emplear en hacer nuestra aplicación más eficiente. Los contenedores son bastante más pequeños en capacidad que una máquina virtual y requieren menor tiempo para iniciarse, por lo que se puede ejecutar muchos más contenedores con los mismos recursos que emplearías en levantar una máquina virtual.



Como podemos observar en la imagen de arriba izquierda, si desplegamos una aplicación clusterizada en máquinas virtuales, cada una de estas necesita un S.O. propio para poder funcionar, lo que supone un gasto de recursos que no pueden ir a otro fin. Sin embargo, si observamos la imagen de arriba derecha, vemos que los contenedores no necesitan un S.O. para correr, únicamente con el “Container Engine” o software contenedor, ya puedes desplegar tantos contenedores como te permitan los recursos de tu máquina física. Esto supone un claro avance en una tecnología con respecto a la otra.

3.4 Containerd

Containerd es un motor de ejecución de contenedores estándar en el sector. Es una herramienta de código abierto que utiliza las características del núcleo del sistema operativo y mejora la gestión de contenedores mediante una capa de abstracción que gestiona cgroups, espacios de nombres, sistemas de archivos, capacidades de red y más.

Es un proyecto que nace en marzo de 2017 como una rama independiente de Docker y que es donado a la CNCF (Cloud Native Computing Foundation). Dos años después, en 2019, ya se consideraba como un proyecto “graduado” dentro de la CNCF, lo que significa que era muy adoptado y apoyado por la comunidad.

Es una aplicación perfecta para manejar cargas de trabajo tanto en despliegues a pequeña escala como a grandes entornos de nivel empresarial, siendo compatible con Kubernetes. Además garantiza la estandarización e interoperabilidad en los entornos de contenedores. A pesar de que existen otros motores de contenedores, como puede ser CRI-O o incluso Docker, debido a las razones mostradas anteriormente nos hemos decantado por Containerd para realizar nuestro despliegue.

3.5 Kubernetes

“Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Kubernetes facilita la automatización y la configuración declarativa. Tiene un ecosistema grande y en rápido crecimiento. El soporte, las herramientas y los servicios para Kubernetes están ampliamente disponibles”.¹

Esta es la definición que Kubernetes ofrece sobre su software, pero preguntémonos ¿Qué es Kubernetes?.

Lo primero que debemos saber es que el nombre Kubernetes proviene del griego y su significado es “timonel” o “piloto”. Esto ya nos indica que Kubernetes se va a encargar de “dirigir” algo, en concreto, contenedores.

¹ Kubernetes. (n.d.). ¿Qué es Kubernetes? Kubernetes Documentation. Recuperado el 3 de mayo de 2024, de <https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/>

Ya hemos hablado anteriormente de los contenedores, esta especie de “cajas” virtuales en la que se guardan las aplicaciones con todo aquello que necesita dicha aplicación para poder funcionar. Si solo contamos con uno o dos de estos contenedores, no hay ningún problema, podemos manejarlos con tranquilidad, pero ¿qué ocurre cuando contamos con decenas o cientos de contenedores? Supondría todo un dolor de cabeza para el administrador o administradores de sistemas estar revisando uno por uno todos los contenedores para comprobar que su funcionamiento es correcto, que si falla algún contenedor, otro ocupe su lugar para seguir ofreciendo el servicio, etc.

Todo esto puede ser manejado por Kubernetes de forma más sencilla. Kubernetes es un software que funciona como orquestador de contenedores, es decir se encarga de gestionar el número de contenedores que se necesitan según la demanda y donde deben ejecutarse. También se encarga de la escalabilidad. Si en un momento dado la aplicación está recibiendo una gran cantidad de peticiones y necesita más recursos para seguir dando un servicio óptimo, Kubernetes se encarga de crear más contenedores con esa aplicación de manera automática, clusterizando la aplicación y permitiendo que mantenga la alta disponibilidad. Kubernetes también se encarga de revisar los contenedores para reiniciarlos en caso de fallos o reemplazarlos automáticamente. Otra de las ayudas que ofrece Kubernetes es lo que se conoce como “despliegue continuo”. Esto nos permite actualizar la aplicación sin dejarla inactiva, ya que mientras en un nodo actualizamos nuestro código, otro nodo sigue manteniendo el servicio activo hasta que el nodo con la nueva actualización esté disponible, lo que permite una transición muy clara. El balanceo de carga es otro de los beneficios que nos propone Kubernetes, ya que se encarga automáticamente de distribuir el tráfico entrante entre todos aquellos contenedores que contengan la aplicación, así nos aseguramos que ninguno de los contenedores tenga un exceso de carga y pueda fallar.

Sin embargo, hay que tener en cuenta que Kubernetes no es una PaaS (Plataforma como servicio o Platform as a Service, en inglés) convencional. Como Kubernetes opera a nivel de contenedor y no a nivel de hardware, ofrece características de las PaaS como deployments, registro o balanceo de carga, sin embargo, Kubernetes no limita el tipo de aplicaciones que soporta. Si la aplicación puede correr en contenedores, puede correr en Kubernetes. Este software no se encarga de compilar la aplicación ni de hacer despliegues de código fuente, así como tampoco puede proveer servicios en capa de aplicación como Middleware, bases de datos o sistemas de almacenamiento. Puedes correr estas aplicaciones en Kubernetes, pero no puede ofrecer estos servicios por sí mismo Kubernetes tampoco te obliga ni te da

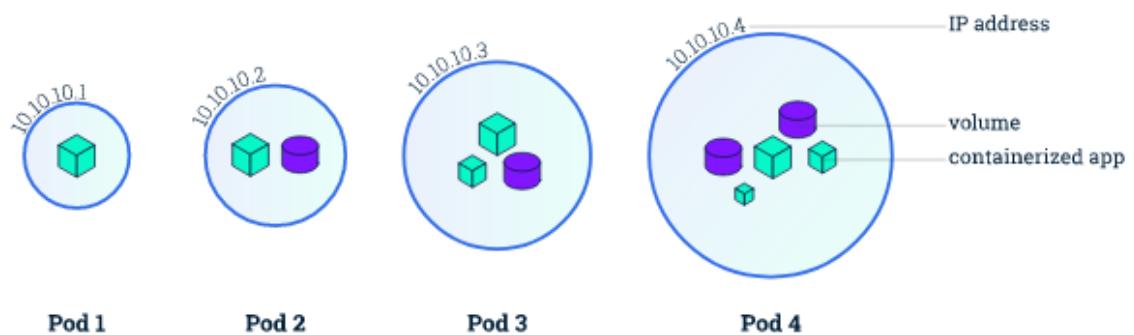
un lenguaje de configuración, te ofrece una API declarativa que puedes usar con cualquier forma de especificación declarativa.

3.6 Arquitectura de Kubernetes

A continuación vamos a ver en detalle cada uno de los elementos que componen Kubernetes y cómo interactúan entre ellos para ofrecer un entorno de contenedores fuerte y escalable.

3.6.1 Pods

Un pod es la unidad de computación desplegable más pequeña que se puede crear y gestionar en Kubernetes. Se trata de un grupo de uno o más contenedores, con un espacio y recursos de red compartidos y una especificación de cómo se deben ejecutar los contenedores. Los contenidos del pod están co-localizados y co-programados, ejecutándose en un contexto compartido. Además de contenedores de aplicaciones, un Pod puede contener contenedores de inicio que se ejecutan en el arranque del Pod, así como contenedores efímeros que pueden depurar un Pod en ejecución.



Los Pods en un clúster de Kubernetes se utilizan de dos formas principales:

- **Pods que ejecutan un único contenedor:** El modelo “un contenedor por Pod” es el más común en el uso de Kubernetes. Podemos pensar que el Pod es una “estructura” que envuelve al contenedor, gestionando Kubernetes los Pods en lugar de los contenedores directamente.
- **Pods que ejecutan múltiples contenedores:** Un pod puede encapsular varios contenedores que forman una aplicación y que se necesitan unos a otros para que la aplicación se ejecute de manera correcta.

3.6.2 Nodos

En Kubernetes un nodo es una máquina de trabajo. Un nodo puede ser una máquina física o virtual, dependiendo del tipo de clúster hayamos configurado. Cada nodo está gestionado por el componente master y contiene todos los servicios que necesita para ejecutar pods. Los servicios en un nodo incluyen:

- **Container Runtime:** Es un componente fundamental que permite a Kubernetes ejecutar los contenedores de una manera eficaz. Es el encargado de gestionar la ejecución y el ciclo de vida de los contenedores dentro del entorno de Kubernetes. Hay que tener en cuenta que Kubernetes admite tiempos de ejecución de contenedores como Containerd, CRI-O y cualquier otra implementación del CRI (Container Runtime Interface) de Kubernetes.
- **Kubelet:** Kubelet es un agente que se ejecuta en cada nodo del clúster. Es el encargado de que los contenedores se ejecuten en un Pod. Kubelet coge un conjunto de PodSpecs que se proporcionan a través de diferentes mecanismos y se asegura de que los contenedores descritos en esos PodSpecs se están ejecutando de forma correcta y no presentan ningún error. Hay que tener en cuenta que Kubelet no puede gestionar contenedores que no hayan sido creados por Kubernetes.
- **Kube-Proxy:** Se trata de un proxy de red que se ejecuta en cada nodo del clúster, encargándose de implementar parte del servicio Kubernetes. Kube-Proxy es el encargado de mantener las reglas de red en los nodos, lo que permite la comunicación por red entre los mismos desde sesiones de red dentro o fuera del clúster. Hay que tener en cuenta que Kube-Proxy utiliza la capa de filtrado de

paquetes del sistema operativo si esta existe y está disponible. En caso contrario, Kube-Proxy reenviará el tráfico por sí mismo.

El estado del nodo nos da bastante información sobre el nodo, destacando los siguientes campos:

- **Direcciones:** El uso de estos campos siempre dependerá del proveedor de servicios en la nube que utilicemos o la configuración de las máquinas locales donde se ejecute el nodo. Los campos que encontraremos serán:
 - **Hostname:** Es el nombre de la máquina huésped que aparece en el kernel del nodo.
 - **ExternalIP:** La dirección IP del nodo que está disponible fuera del clúster.
 - **InternalIP:** La dirección IP del nodo que solo se puede acceder desde dentro del clúster.
- **Estados:** Nos indica en qué estado se encuentra el nodo del clúster. Aquí podemos encontrar muchas opciones y estados distintos, pero vamos a destacar las siguientes:
 - **Ready:** Si se encuentra en “True” quiere decir que el nodo está en perfectas condiciones y listo para aceptar nuevos Pods. Si su estado es “False”, quiere decir que no es capaz de administrar nuevos Pods.
 - **PIDPressure:** Por defecto está en “False” y eso nos indica que no hay demasiados procesos en ejecución en el nodo. Si su estado es “True” quiere decir que hay demasiados procesos en el nodo.
 - **OutOfDisk:** Por defecto aparece en “False” y nos indica que hay espacio para añadir nuevos Pods al nodo. Si no hay suficiente espacio, pasará a estado “True”.
 - **NetworkUnavailable:** El valor “True” nos indicará que la configuración de red del nodo es incorrecta y por lo tanto no puede conectarse con el resto de nodos. Si la configuración es correcta, el estado será “False”.

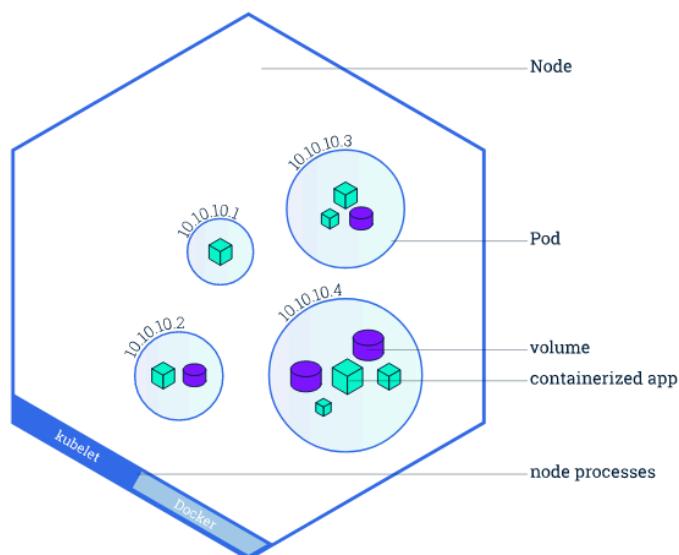
El estado de los nodos siempre se va a representar como un objeto JSON.

```
1  "conditions": [
2    {
3      "type": "Ready",
4      "status": "True"
5    }
6  ]
```

El ejemplo de arriba nos muestra cómo se describiría un nodo en buen estado.

- **Capacidad:** Describe los recursos que hay disponibles en el nodo seleccionado. Estos son la CPU, la memoria y el número máximo de Pods que pueden ser planificados dentro del nodo, lo que nos da una idea de cuanto podemos escalar nuestro servicio o cuantos servicios podemos tener en un nodo.
- **Información:** Esto nos da información general sobre el nodo, como puede ser la versión de kernel, la versión de Kubernetes que se está utilizando, la versión de Docker u otros sistemas de contenerización como Containerd y el nombre del sistema operativo.

Hay que tener en cuenta que los nodos no son creados por Kubernetes de forma inherente, al contrario que los Pods o los Services. Los nodos son creados de manera externa por proveedores de servicios en la nube como puede ser Amazon Web Services o creados por el administrador del sistema en la colección de máquinas, ya sean virtuales o físicas.



Esto quiere decir que cuando Kubernetes crea un nodo, lo que en realidad está creando es un objeto que representa al nodo y, después de ser creado, Kubernetes comprueba si es válido.

Un ejemplo de estos objetos que se crean para representar a los nodos es el siguiente:

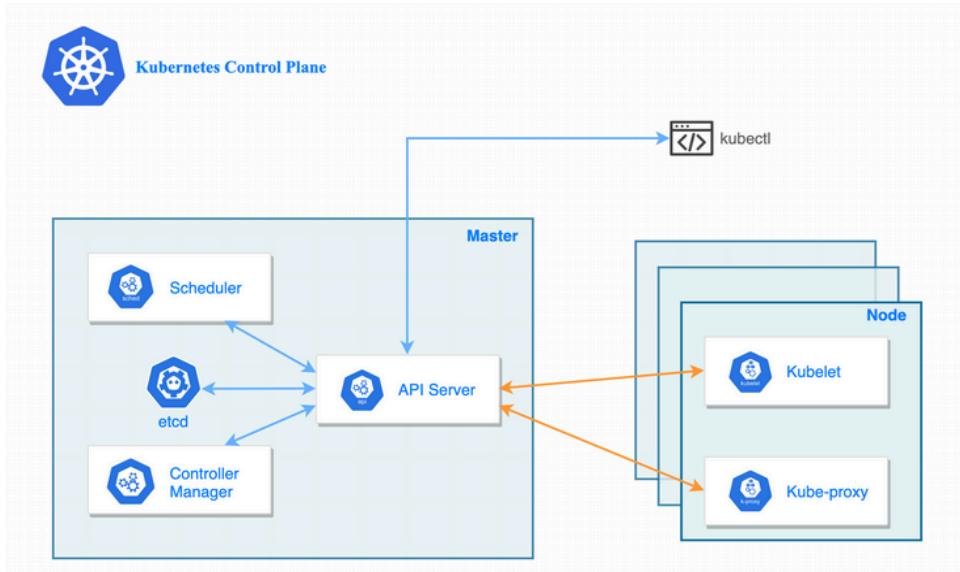
```
{  
  "kind": "Node",  
  "apiVersion": "v1",  
  "metadata": {  
    "name": "192.168.1.48",  
    "labels": {  
      "name": "k8s-master"  
    }  
  }  
}
```

Kubernetes crea un objeto “node” de manera interna y valida el mismo comprobando su estado en el campo “name” que se encuentra dentro del campo “metadata”. Si el nodo es funcional y todos los servicios que son necesarios se están ejecutando, Kubernetes decide que el nodo es apto para ser elegido y correr un Pod. Si estos parámetros no son correctos, Kubernetes ignora este nodo para cualquier actividad que se produzca en el clúster hasta que el nodo sea válido.

3.6.3 Comunicación entre Nodos y el “Control Plane”

Para estudiar la comunicación entre los nodos y el “Control Plane” o maestro, debemos saber que el patrón de API de Kubernetes es “Hub-and-spoke”². Todo uso de la API desde los nodos o los pods que se están ejecutando termina en el servidor central o hub. Ninguno de los otros componentes está diseñado para mostrar servicios al exterior, por lo que todo el tráfico pasa a través del nodo maestro. El servidor o nodo maestro está configurado para escuchar conexiones del exterior a través de un puerto seguro de HTTPS (por defecto, el 443). Debe de haber al menos una forma de autorización habilitada, sobre todo si se permiten solicitudes anónimas o tokens de cuentas de servicios. Los nodos del clúster deben contar con un certificado raíz público del clúster para poder conectarse de forma segura al servidor API con credenciales de cliente válidas.

² Arquitectura de red donde un punto central (hub) actúa como núcleo de control, mientras que los demás puntos (spokes) se conectan directamente al hub y no entre ellos.



Los pods que quieran conectarse al servidor API pueden hacerlo de forma segura aprovechando una cuenta de servicio para que Kubernetes inyecte automáticamente el certificado raíz público y un token de portador válido en el pod cuando se instancie. El namespace o servicio de Kubernetes se configura con una dirección IP virtual que se redirige al punto final HTTPS en el servidor API a través de Kube-Proxy. Hay que tener en cuenta que los componentes del control-plane también se comunican con el servidor API a través del puerto seguro.

Esto da como resultado el funcionamiento por defecto para las conexiones desde los nodos y pods que se ejecutan desde los nodos al control-plane son seguras por defecto y pueden ejecutarse en redes públicas o que no sean de confianza.

Existen dos vías de comunicación principales desde el control-plane o servidor API hasta los nodos. La primera es desde el servidor API al proceso kubelet que se ejecuta en cada nodo del clúster. La segunda es desde el servidor API a cualquier nodo, pod o servicio a través de la funcionalidad proxy que tiene el propio servidor API.

- **Del servidor API a kubelet:** Estas conexiones se utilizan para obtener registros de los pods, adjuntar pods en ejecución a través de kubectl y proporcionar la funcionalidad de reenvío de puertos del kubelet. Estas conexiones terminan en el endpoint HTTPS del kubelet. Por defecto, el servidor API no verifica el certificado de servidor del kubelet, lo que hace que la conexión sea vulnerable a ataques man-in-the-middle³, por lo que no es recomendable ejecutar este tipo de conexión en redes públicas o que no sean fiables.

³ Tipo de ciberataque donde un atacante intercepta y altera la comunicación entre dos partes que creen que están comunicándose entre sí.

- **Servidor API a nodos, pods y servicios:** Las conexiones desde el servidor API a un pod, servicio o nodo son conexiones HTTP simples por defecto, por lo que no están autenticadas ni cifradas. Sin embargo, estas conexiones también pueden ejecutarse a través del protocolo HTTPS seguro si ponemos https: por delante del nombre del nodo, pero hay que tener en cuenta que no validará el certificado proporcionado por el endpoint HTTPS ni proporcionará credenciales del cliente. Así que, aunque la conexión está cifrada, no tenemos garantía de integridad. Este tipo de conexiones no son nada seguras en redes públicas o aquellas que no sean de confianza.
- **Túnel SSH:** Kubernetes admite túneles SSH para proteger las comunicaciones entre el control-plane y los nodos. En esta configuración, el servidor API inicia una conexión SSH a cada nodo del clúster a través del puerto 22, pasando por ahí todo el tráfico destinado a un nodo, pod, servicio o kubelet. Este tipo de conexión garantiza que el tráfico no queda expuesto fuera de la red en la cual se están ejecutando los nodos.

3.6.4 Controllers

En Kubernetes, los controllers son bucles de control que observan el estado del clúster y realizan cambios cuando es necesario. Cada controller intenta acercar el estado actual del clúster al estado deseado.

Un controller rastrea al menos un tipo de recurso dentro de Kubernetes. Estos objetos tienen un campo llamado “spec” que representa el estado que deseamos que tenga dicho objeto, y el controller es el encargado de hacer que el estado actual del objeto se acerque a ese estado. Aunque el controller puede realizar esta acción por sí mismo, es más común que envíe mensajes al servidor API para solucionar estas diferencias.

Un ejemplo de controller a través del servidor API es el controller de Jobs. Un job es un recurso que Kubernetes utiliza cuando ejecuta uno o varios Pods para llevar a cabo una tarea y después finalizarla. Cuando el controller de jobs ve una nueva tarea, se hace cargo para que, en algún lugar de tu clúster que él decida, los kubelets en un conjunto de Nodos estén ejecutando el número correcto de Pods para realizar dicha tarea. Hay que tener en cuenta que el job controller no ejecuta ningún Pod ni ningún contenedor por si mismo, sino que le dice al servidor API que ejecute, elimine o cree Pods según sus necesidades.

Después de crear un nuevo job, el estado al que se desea llegar con ese job es “completed” o completado. Para ello, el job controller se encarga de que su estado se acerque lo máximo posible a ello, creando Pods que hagan las tareas necesarias para ese job, y que de esta manera se complete lo antes posible.

Sin embargo, otros controllers necesitan hacer cambios fuera del clúster, como por ejemplo un controller que se asegure que hay suficientes nodos en el clúster. Estos controllers que interactúan con el exterior encuentran el estado al que deben acercarse en el servidor API, comunicándose con el exterior para ajustar el estado actual. La capacidad de estos controllers es tal que incluso pueden escalar horizontalmente los nodos del clúster (siempre que haya nodos disponibles).

Con clústeres de Kubernetes, estos controllers pueden trabajar con herramientas de gestión de direcciones IP, servicios de almacenamiento, API de proveedores en la nube y otros servicios.

3.6.5 Leases

Es habitual que los sistemas distribuidos utilicen leases para bloquear los recursos compartidos y coordinar la actividad de todos los miembros del conjunto. Esto no es una excepción en Kubernetes, donde el concepto de leases está representado por objetos de tipo “lease” en el grupo de API “coordination.k8s.io”, el cual se utiliza para gestiones críticas del sistema de Kubernetes, como puede ser los heartbeats de los nodos y la elección de líder a nivel de componente.

- **Heartbeats del nodo:** Kubernetes utiliza la API lease para informar de los heartbeats del nodo kubelet al servidor de la API de Kubernetes. Debemos tener en cuenta que, para cada nodo, existe un objeto lease con un nombre que coincide en el namespace del kube-node-lease. En definitiva, cada heartbeat o latido del kubelet es una solicitud de actualización para ese objeto lease del que hablábamos anteriormente, actualizando el campo “spec.renewTime” del objeto lease. El control-plane de Kubernetes utiliza la marca de tiempo que queda registrada en ese campo para determinar si el nodo que está consultando está operativo o no.

- **Elección del líder:** Kubernetes también va a valerse de estos leases para asegurar que únicamente una instancia de un componente se está ejecutando en un momento determinado. Esto lo puede utilizar componentes del control-plane como “kube-controller-manager” o “kube-scheduler” en configuraciones de alta disponibilidad, donde únicamente una instancia del componente se está ejecutando mientras el resto de instancias están activas y a la espera.
- **Identidad del API server:** Cada kube-APIServer también utiliza la API lease para informar al resto del sistema de su condición de APIServer. Esto nos proporciona un mecanismo para que los clientes puedan averiguar cuántas instancias de Kubernetes APIServer están funcionando en el control-plane de Kubernetes.

3.6.6 Container Runtime Interface (CRI)

Se trata del protocolo principal para la comunicación entre el kubelet y el sistema de contenedores. El container runtime interface o CRI es un plugin de Kubernetes que permite a kubelet utilizar una gran variedad de sistemas de contenedores o “container runtime” como Docker, Containerd o Podman sin necesidad de volver a compilar los componentes del clúster. Es fundamental para el buen funcionamiento de un clúster de Kubernetes que haya un container runtime en ejecución en cada uno de los nodos que tenemos en el clúster, para que kubelet pueda iniciar y gestionar los pods que tendrán uno o varios contenedores en su interior.

3.7 Importancia y relevancia de Kubernetes y la computación en la nube

En el panorama tecnológico actual, donde la tecnología está en constante crecimiento, la computación en la nube y los Kubernetes tienen un papel importante en la infraestructura de TI en las organizaciones modernas.

Kubernetes ha emergido como una herramienta indispensable para la gestión eficiente de aplicaciones y servicios. Permite escalar aplicaciones automáticamente, lo cual es crucial para empresas con variaciones en el tráfico. Además, su flexibilidad permite desplegar las aplicaciones en entornos variados como la nube pública, instalaciones locales y entornos híbridos, ofreciendo libertad de elegir la mejor estrategia de despliegue.

Kubernetes simplifica muchas tareas mediante la automatización de procesos como el actualización de nuevas versiones, la supervisión de la salud de los servicios y la recuperación ante fallos. Esta automatización, no solo reduce la carga de trabajo, sino que también minimiza el riesgo de errores, mejorando así la fiabilidad y la eficiencia operativa. Además, la portabilidad que ofrece permite que las aplicaciones se ejecuten de manera consistente en diferentes entornos, lo que es valioso en estrategias híbridas y multi-nube.

La computación en la nube proporciona a las empresas una plataforma flexible y escalable para acceder a recursos informáticos a través de internet, eliminando la necesidad de gestionar infraestructuras físicas. Además, permite responder ágilmente, ajustando sus recursos según las necesidades o las condiciones de mercado, útil para negocios con variaciones en la demanda o en crecimiento.

También reduce las grandes inversiones en hardware y mantenimiento, ya que las empresas solo pagan por los recursos que utilizan. Los proveedores de servicios en la nube ofrecen altos niveles de disponibilidad y opciones de recuperación ante desastres, garantizando que los datos y aplicaciones estén siempre accesibles y protegidos. Las plataformas invierten considerablemente en infraestructura de seguridad y medidas de cumplimiento normativo, ofreciendo niveles de protección que muchas organizaciones no podrían lograr por sí mismas.

En resumen, Kubernetes y la computación en la nube son fundamentales en el entorno empresarial moderno. Juntos, permiten a las organizaciones ser más ágiles, eficientes y capaces de innovar rápidamente.

Kubernetes tiene la capacidad de automatizar la gestión de aplicaciones contenerizadas, ofrecer escalabilidad y flexibilidad, todo ello mejora la eficiencia operativa y reduce errores. Además, se ha enriquecido con una amplia gama de herramientas complementarias como Helm y Prometheus, que extienden sus capacidades y cuenta con características de seguridad que ayudan a proteger los sistemas contra amenazas.

Por su parte, la computación en la nube proporciona elasticidad, reducción de costos y acceso a tecnologías avanzadas, facilitando la innovación y permitiendo a las empresas responder rápidamente a las demandas del mercado.

Juntas, estas tecnologías no solo optimizan la gestión y el despliegue de aplicaciones, sino que también impulsan la transformación digital, permitiendo a las organizaciones competir y prosperar en un entorno en constante evolución.

3.8 Razones para elegir Kubernetes en la nube

Debido a todo lo mencionado anteriormente, utilizar Kubernetes en la nube representa una decisión que puede impulsar significativamente la eficiencia, la innovación y la competitividad de una empresa en el entorno empresarial actual.

En primer lugar, Kubernetes en la nube ofrece una escalabilidad incomparable. Al aprovechar la infraestructura de los proveedores en la nube, las empresas pueden escalar sus aplicaciones automáticamente. Esto garantiza que tengan un rendimiento óptimo incluso durante picos de tráfico más altos, ofreciendo mejor experiencia para los usuarios.

Además, la flexibilidad y la portabilidad de Kubernetes en la nube permiten a las empresas adaptarse a las cambiantes condiciones del mercado y las demandas de los clientes. Kubernetes hace que las aplicaciones puedan ejecutarse en cualquier entorno de nube, eliminando las restricciones de la infraestructura local. Esto facilita la migración de aplicaciones entre diferentes proveedores o la implementación de estrategias híbridas, proporcionando una mayor agilidad y flexibilidad.

Además, Kubernetes en la nube simplifica y automatiza muchas tareas operativas, lo que reduce la carga de trabajo y libera recursos. Automatiza gran parte del ciclo de vida de las aplicaciones, desde el despliegue y la gestión de aplicaciones hasta la supervisión y el escalado, lo que permite a las empresas ser más eficientes y centrarse en innovación y desarrollo.

3.9 Ventajas de utilizar Kubernetes en comparación con otras soluciones

Kubernetes es una plataforma para la orquestación de contenedores, pero existen alternativas que se adaptan a diferentes necesidades y preferencias. Algunas alternativas:

Docker Swarm, esta herramienta es de configuración sencilla, integrado con Docker, fácil de usar y buena para clústeres pequeños. Pero tiene menos funciones que Kubernetes y es menos adecuada para cargas de trabajo complejas.



Apache Mesos con Marathon, es altamente escalable, se puede trabajar con o sin contenedores y es fuerte en entornos multi-nube. Aunque su configuración es más compleja y tiene menos enfoque en contenedores.



Amazon ECS (Elastic Container Service), al ser un servicio de Amazon está integrado con AWS, es fácil de usar y no es necesario el plano de control. Pero tiene menos flexibilidad y características, además de tener dependencia de AWS.



Nomad, es una herramienta que se caracteriza por la simplicidad y la ligereza, soporta varios tipos de cargas de trabajo además de contenedores, es de fácil integración con HashiCorp. Aunque tiene menos características integradas que Kubernetes.



OpenShift, tiene mayor seguridad, es amigable para desarrolladores ya que tiene soporte robusto. Lo malo de este programa es que tiene mayor complejidad y mayor costo.



Rancher, esta herramienta simplifica el despliegue de Kubernetes, la gestión de múltiples clústeres y es buena para entornos híbridos y multi-nube. Los contras de esta herramienta es que al estar construida sobre Kubernetes, no se puede usar con otras herramientas..



Google Cloud Run, utiliza el servicio en la nube, no es necesario gestionar infraestructura y tiene buena integración con Google Cloud. Pero está limitado a contenedores sin estado y tiene dependencia con Google Cloud Platform.



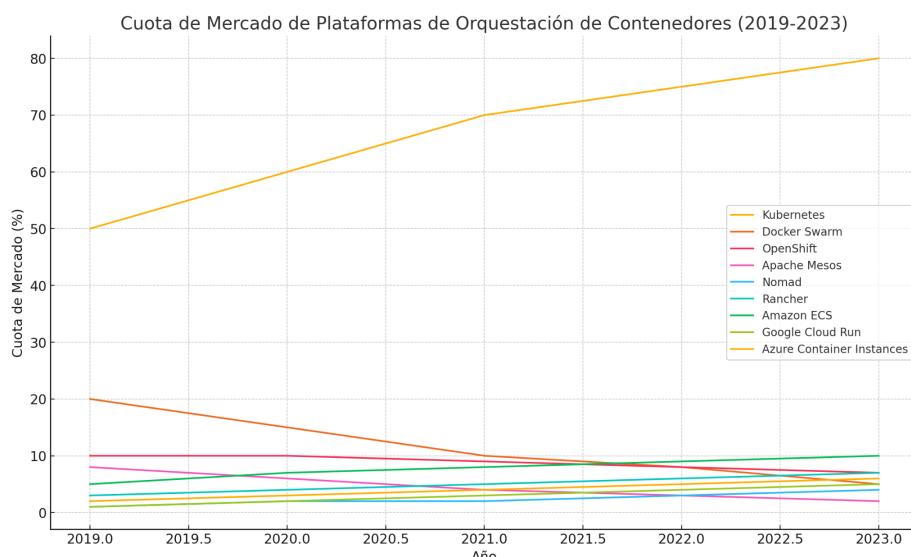
Cuando se trata de elegir la mejor opción para implementar un clúster en una empresa, Kubernetes destaca como la elección más sobresaliente. En primer lugar, su capacidad de escalabilidad es insuperable, permite que las aplicaciones se escalen de manera automática, adaptándose fácilmente a las necesidades cambiantes de una empresa. En contraste, otras alternativas pueden no tener esta flexibilidad y presentar dificultades para manejar cargas de trabajo variables.

Además, la portabilidad de Kubernetes es un punto a favor indiscutible, su capacidad para ejecutarse en una variedad de entornos, desde la nube pública hasta instalaciones locales. Proporciona a las empresas libertad para elegir la infraestructura que mejor se adapte a sus necesidades específicas. Otras opciones pueden estar limitadas a ciertos entornos o proveedores en la nube, que puede resultar en una dependencia no deseada o dificultades para migrar datos.

La automatización y gestión de recursos de Kubernetes son también áreas en las que destaca claramente. La implementación, actualización y escalado de aplicaciones se pueden automatizar, reduciendo la carga de trabajo y permitiendo una gestión más eficaz. Kubernetes ofrece una gestión de recursos más avanzada y una mayor capacidad para optimizar el uso de los recursos disponibles, lo que resulta en una mayor eficiencia operativa y un menor costo total de propiedad.

En resumen, “Kubernetes se ha convertido desde su lanzamiento en 2014 en el orquestador de contenedores más popular del mundo. Su principal punto diferencial es el uso del concepto de pods, una forma muy eficiente de agrupar contenedores y facilitar la comunicación entre microservicios.”⁴

Además, destaca por su incomparable escalabilidad, automatización y gestión de recursos, lo que garantiza la flexibilidad para ejecutarse en diferentes entornos y optimizar las operaciones. Todo ello lo convierte en la elección ideal para empresas de cualquier tamaño que buscan una solución de orquestación de contenedores confiable y robusta.



⁴ Asum Cloud. (n.d.). *¿Cuál es el mejor orquestador de contenedores? Descúbrelo ahora.* Recuperado de <https://ausum.cloud/kubernetes-swarm-mesos-mejor-orquestador-contenedores/#swarm>

3.10 CNI

El Container Network Interface, o CNI, es una identificación estándar y un conjunto de bibliotecas desarrolladas por el proyecto de contenedores Cloud Native Computing Foundation. La finalidad del CNI es configurar interfaces de red en los contenedores y eliminarlas cuando dichos contenedores son eliminados por el administrador. Entre las funciones del CNI se encuentran la configuración de red de contenedores, la interoperabilidad entre herramientas y orquestadores de contenedores, los plugins de red y la gestión de vida de la red.

Existen distintos CNI que podemos utilizar, como Flannel, Weave o Calico. Nosotros vamos a centrarnos en este último. Podríamos definir Calico como “un proyecto de comunidad de código abierto que proporciona redes para los contenedores y las máquinas virtuales”⁵.

Calico es un proyecto basado en la capa 3 o de red del modelo OSI que utiliza BGP⁶ para crear tablas de direccionamiento las cuales facilitan la comunicación entre los nodos. Calico crea una red de capa 3 y va asignando direcciones IP completamente direccionables a cada pod que se encuentra dentro del nodo. Para ello, divide una gran red CIDR⁷ en pequeños bloques de direcciones IP y va asignando uno o más bloques a cada nodo según las necesidades del mismo. Para direccionar el tráfico del pod, Calico utiliza todo lo que el sistema le pueda proporcionar, como las tablas de enrutamiento locales o las iptables que haya configuradas en cada nodo.

3.11 Relevancia en un entorno laboral actual

Implementar Kubernetes en la nube es una estrategia poderosa para optimizar la gestión de aplicaciones. Esta plataforma permite escalar automáticamente los recursos según la demanda, asegurando alta disponibilidad y eficiencia operativa. Imagina tener la capacidad de desplegar, gestionar y optimizar tus aplicaciones de manera eficiente, reduciendo costos operativos y mejorando la resiliencia de tu infraestructura. Con Kubernetes en la nube, se pueden ejecutar aplicaciones en cualquier entorno, ya sea en la nube pública, privada o híbrida, garantizando su continuidad sin interrupciones.

⁵ IBM. (n.d.). *Calico*. IBM Cloud Private. Recuperado el 10 de mayo de 2024, de <https://www.ibm.com/docs/es/cloud-private/3.2.x?topic=ins-calico>

⁶ Border Gateway Protocol. Protocolo estandar para intercambio de información de enrutamiento entre sistemas autónomos.

⁷ Classes Inter-Domain Routing. Es un método de asignación de IPs que mejora la eficiencia en el uso de las mismas.

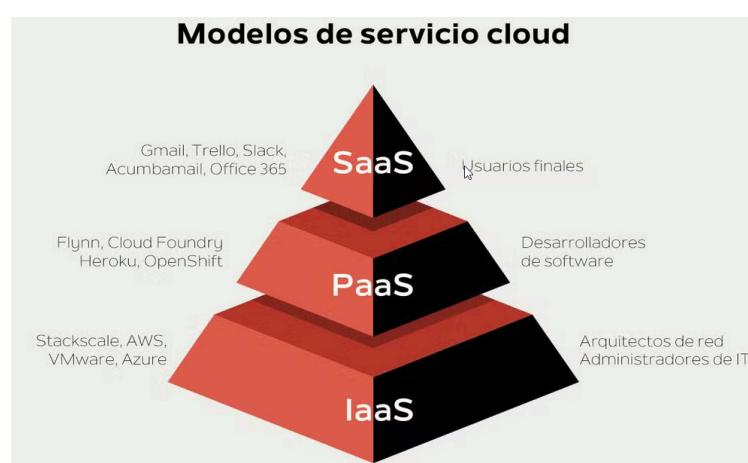
Kubernetes en la nube también ofrece una potente automatización que libera a tu equipo de tareas repetitivas, permitiéndoles centrarse en la innovación. Con características integradas de balanceo de carga, monitorización y seguridad, tus aplicaciones no solo serán más eficientes sino también más seguras y confiables. Las actualizaciones continuas sin tiempo de inactividad y la capacidad de revertir cambios problemáticos aseguran que tu empresa siempre esté a la vanguardia.

3.12 Servicios en la nube ¿El futuro de la computación?

Hoy en día la nube es un concepto que está muy presente en nuestras vidas. Casi todo el mundo tiene un servicio de almacenamiento en la nube y se jacta de sus beneficios frente al sistema local tradicional pero... ¿Sabemos acaso lo que es la nube?

Los servicios en la nube son plataformas, infraestructuras o software que proveedores externos ponen a disposición de los usuarios a través de internet. Existen distintos tipos de servicios en la nube y estos son:

- **Plataforma como servicio (PaaS):** Esto ofrece a los desarrolladores una plataforma para crear aplicaciones sin tener que preocuparse de la infraestructura. Google App Engine o Heroku son dos buenos ejemplos de este tipo de servicios.
- **Software como servicio (SaaS):** Uno de los servicios más conocidos gracias a Gmail u Office 365, este servicio distribuye aplicaciones a través de internet sin tener que instalar dicho software en el PC local.
- **Infraestructura como servicio (IaaS):** Este tipo de servicio proporciona al usuario recursos de almacenamiento, computación y redes bajo demanda. Existen una gran cantidad de empresas que ofrecen estos servicios, como puede ser Amazon Web Services, Google Cloud Platform, Microsoft Azure o IBM Cloud.



Los servicios en la nube ofrecen una gran cantidad de ventajas respecto a la computación tradicional. Por ejemplo, en la escalabilidad de servicios, ya que si en algún momento tienes un pico excepcional de peticiones a tu servicio, puedes escalarlo de manera cómoda sin tener que invertir en hardware adicional que luego no utilices, ya que puedes aumentar o reducir recursos a demanda.

Además es perfecto para empresas que están empezando a desarrollar su actividad y no quieren realizar una inversión demasiado grande, ya que puedes pagar estos servicios por el uso que le das e ir creciendo poco a poco.

La accesibilidad es otro de sus beneficios, ya que al permitir la conexión remota desde cualquier lugar con internet, facilita que el trabajo pueda resolverse independientemente de la localización en la que se encuentre uno.

Hay que tener en cuenta también que los proveedores de la nube cuentan con fuertes medidas de seguridad ante ciberataques y varias opciones de recuperación ante un desastre, lo que supone una protección muy eficiente de aquellos datos que sean sensibles y asegura la alta disponibilidad del negocio que se está llevando a cabo.

No podemos olvidar que los servicios en la nube incluyen actualizaciones automáticas para poder tener acceso a las últimas funcionalidades que se hayan desplegado y las mejoras de seguridad más avanzadas.

Sin embargo, no todo son ventajas en los servicios en la nube. Si creas toda una estructura de negocio alrededor de un proveedor de servicios, acabas siendo dependiente de este, lo que puede suponer un aumento de costes a largo plazo y el hecho de depender de dicho proveedor. Si él sufre un fallo en su sistema, no podrá ofrecer tu servicio con normalidad.

En definitiva, la computación en la nube presenta muchas ventajas pero también inconvenientes a tener en cuenta. A pesar de ello, se trata de una tendencia actual que sin duda crecerá con el paso de los años y cambiará la forma en que operan las empresas y compiten en el mercado global.

3.13 IBM Cloud

IBM Cloud es una plataforma integral de computación en la nube que ofrece una amplia gama de servicios, incluyendo las mencionadas en el punto anterior: infraestructura como servicio, plataforma como servicio y software como servicio. Proporciona opciones flexibles de infraestructura (como máquinas virtuales), junto con avanzadas soluciones de redes y almacenamiento. Su soporte para bases de datos y análisis avanzado permite obtener información a partir de grandes volúmenes de datos en tiempo real.

Se utilizará el servicio IBM Kubernetes Service (IKS), para desplegar el clúster Kubernetes altamente disponible. IKS ofrece capacidades integradas de alta disponibilidad y escalabilidad, lo que facilita la administración del clúster y reduce la carga operativa.

Se crearán varios nodos del clúster Kubernetes dentro de IBM Cloud, para garantizar la redundancia y la resistencia a fallos. Esto permitirá que la aplicación siga estando disponible incluso en caso de fallos, lo que nos va a proporcionar alta disponibilidad.

IBM Cloud destaca entre sus competidores por su estrecha integración con Watson AI, ofreciendo herramientas avanzadas de inteligencia artificial y aprendizaje automático que permiten a las empresas crear aplicaciones inteligentes y obtener valiosos conocimientos de sus datos. Además, la plataforma se centra en la seguridad y el cumplimiento normativo al proporcionar el cifrado y las certificaciones globales, para proteger los datos y cumplir con los requisitos legales. Otra ventaja importante es su capacidad para manejar entornos multicloud e híbridos, facilitando la integración de infraestructuras locales con servicios en la nube. La plataforma también lidera en tecnologías emergentes como blockchain y computación cuántica, ofreciendo herramientas avanzadas.

4. Descripción del proyecto

4.1 Alcance del proyecto

El alcance de este proyecto se centra en dos objetivos principales: la implementación de un cluster de Kubernetes en un entorno local y la posterior creación y despliegue de este cluster en un entorno de nube pública. En primer lugar, se abordará la configuración y despliegue de un cluster de Kubernetes en máquinas locales, incluyendo la instalación y configuración de herramientas necesarias como Containerd y kubeadm. Se realizarán pruebas de funcionamiento y ajustes de configuración para asegurar la estabilidad del clúster.

Posteriormente, se seleccionará un proveedor de servicios en la nube, como AWS, Google Cloud, IBM Cloud o Azure, y se procederá a la creación del clúster en la nube, manteniendo la configuración y estructura original. Se configurarán servicios adicionales en la nube, como balanceadores de carga, para optimizar el rendimiento y la alta disponibilidad del cluster.

Se documentarán detalladamente los pasos seguidos, proporcionando recomendaciones para futuros despliegues de clústers de Kubernetes en diferentes entornos. Este proyecto abarcará tanto los aspectos técnicos de la configuración y despliegue como las consideraciones prácticas para el despliegue y operación en diversos entornos.

4.2 Metodología a seguir

Para poder realizar este proyecto hemos seguido una metodología bastante clara. Lo primero ha sido estudiar en profundidad cómo funciona Kubernetes, su arquitectura y sus diferentes opciones. Una vez teníamos los conceptos claros, comenzamos a desplegar nuestro clúster a nivel local para poder practicar la instalación, el uso y el despliegue de aplicaciones en Kubernetes.

Empezamos con el despliegue en local ya que nos permitió empezar de nuevo varias veces por diversos errores que fuimos resolviendo. Una vez realizamos distintas pruebas con nuestro despliegue en local y vimos que la configuración y la instalación fueron correctas, pasamos a realizar el despliegue en un servidor en la nube, donde no contábamos con tanto tiempo para corregir errores, por lo que era necesario tener bien fijados los conceptos y haber obtenido experiencia con Kubernetes.

4.3 Herramientas y tecnologías a utilizar

Para la realización de este proyecto vamos a utilizar distintas herramientas para conseguir nuestros objetivos. Para desplegar nuestras máquinas virtuales utilizaremos Oracle VM VirtualBox, un hipervisor gratuito que nos permite diseñar máquinas virtuales con los recursos que necesitemos.

Los nodos del clúster local necesitan una red de contenedores para poder comunicarse entre ellos, por lo que será necesario utilizar Calico para este fin. Además Kubernetes necesita un sistema de contenedores sobre el que poder trabajar, por lo que escogeremos Containerd como software de contenerización.

Para poder realizar pruebas, debemos desplegar una aplicación. Hemos escogido Nginx, un servidor web de código abierto ya que nos permite un despliegue sencillo y una personalización ideal para ver este tipo de pruebas.

Por último, necesitaremos también una IaaS para poder realizar nuestro despliegue en la nube. En este apartado hemos optado por utilizar la plataforma IBM Cloud debido a que ofrece un crédito de 200 euros para poder trabajar con ello, lo que nos ha permitido hacer toda clase de pruebas y despliegues sin tener que desembolsar dinero.

5. Despliegue de un clúster de Kubernetes en local

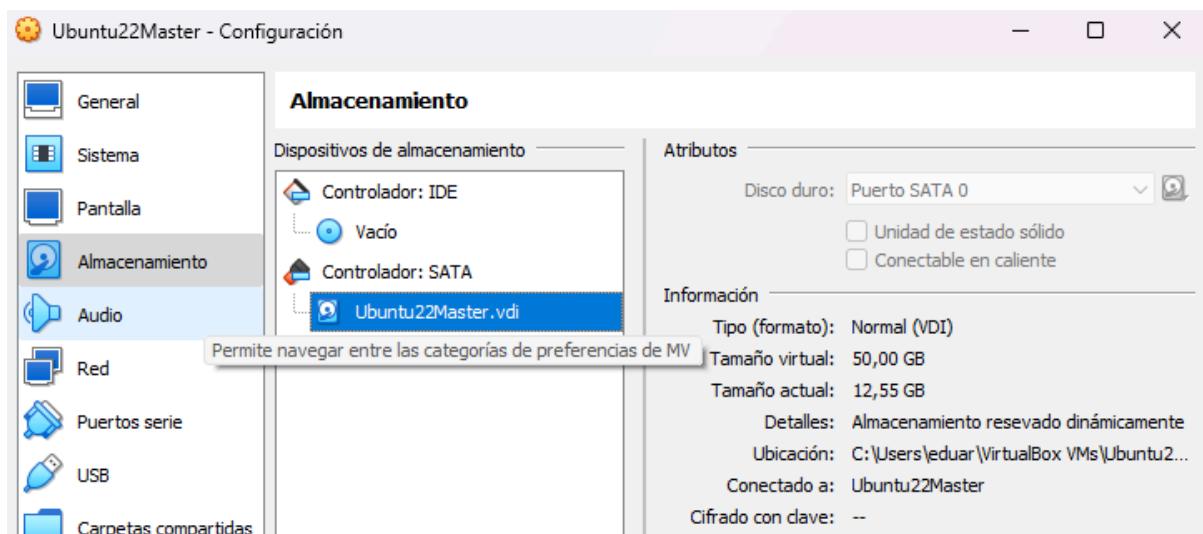
5.1 Configuración de las máquinas

A continuación vamos a mostrar las configuraciones que llevan las máquinas que vamos a estar utilizando.

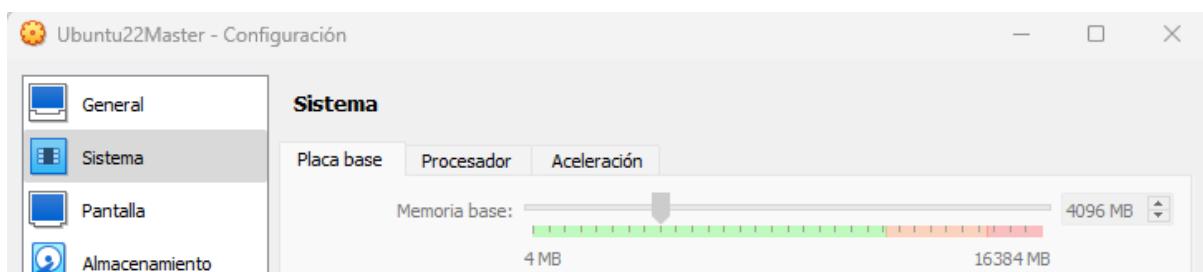
Configuración de las máquinas virtuales:

Sistema operativo: Como sistema operativo hemos optado por escoger un sistema operativo en base Linux, en concreto [Ubuntu-22.04.4](#) ya que se trata de una versión bastante estable de Ubuntu que soporta las configuraciones de Kubernetes sin problema.

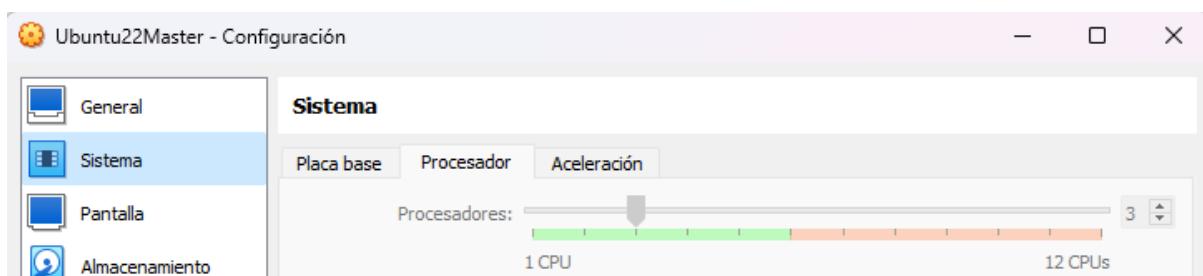
A cada máquina se le otorgará un disco duro virtual de 50 GB para poder manejar sin problema las configuraciones de Kubernetes,. Será un almacenamiento reservado dinámicamente para utilizar solamente el espacio que necesitamos.



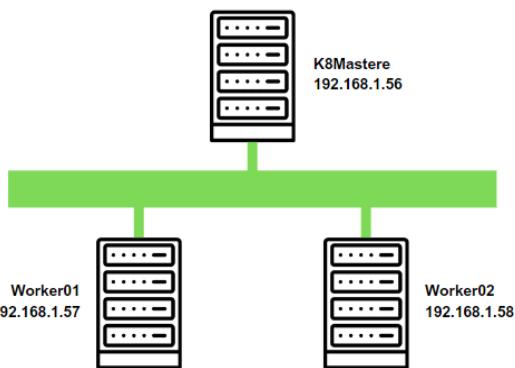
En lo concerniente a la memoria RAM, hemos optado por 4 GB para cada uno de los nodos, ya que se trata de la configuración mínima que recomienda Kubernetes.



El procesador de cada una de las máquinas contará con 3 núcleos para poder hacer frente a las peticiones de Kubernetes. Se trata otra vez de los recursos recomendados como mínimo para que Kubernetes pueda funcionar con normalidad.



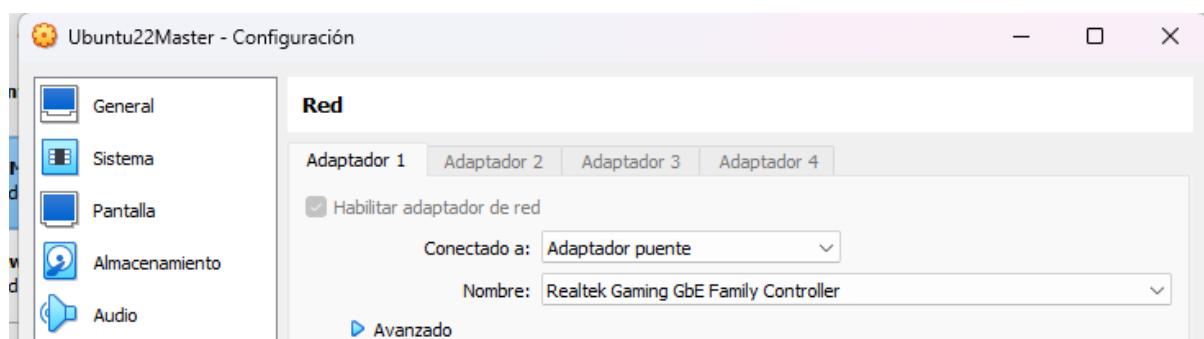
Las tres máquinas estarán conectadas a una red local (física) con un adaptador puente cada una de ellas, para poder tener salida a internet.



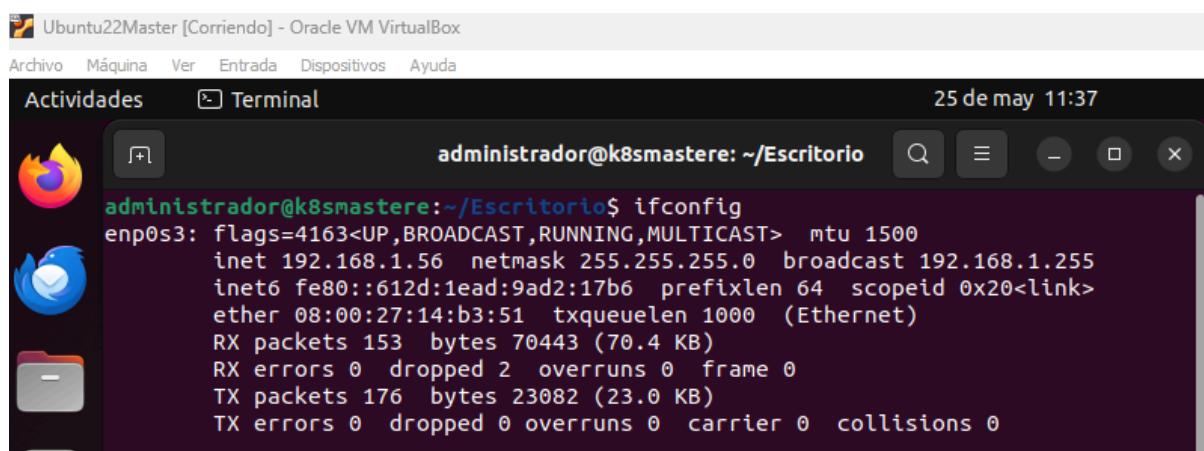
K8Mastere - 192.168.1.56

Worker01 - 192.168.1.57

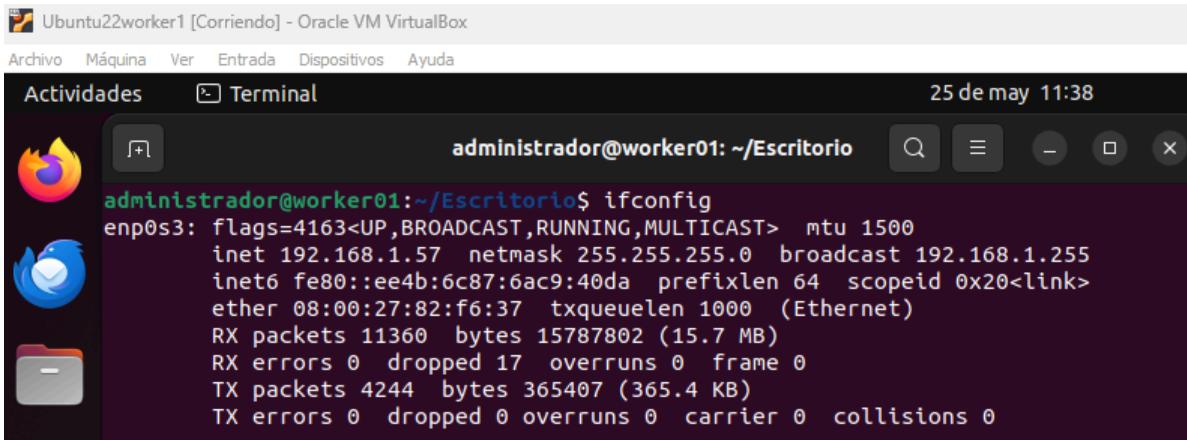
Worker02 - 192.168.1.58



K8Mastere:

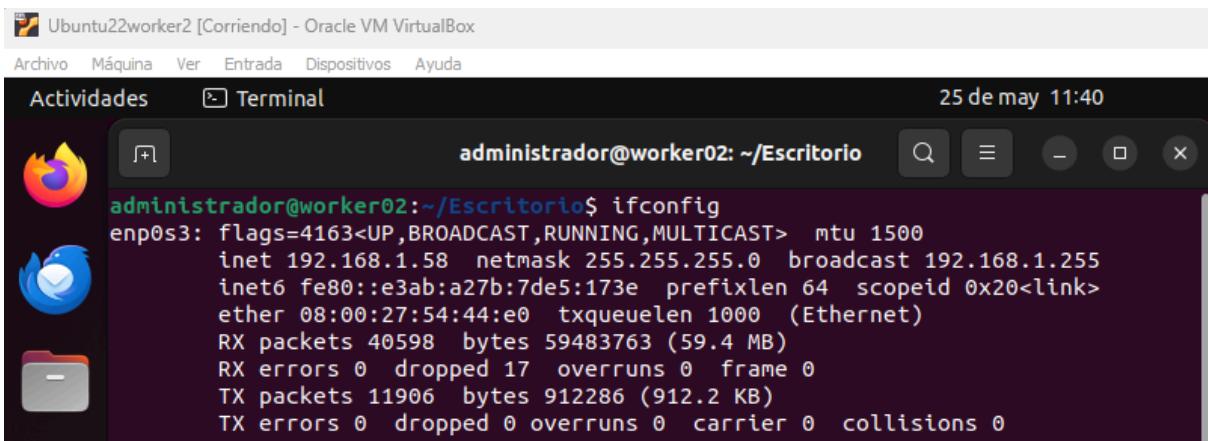


Worker1:



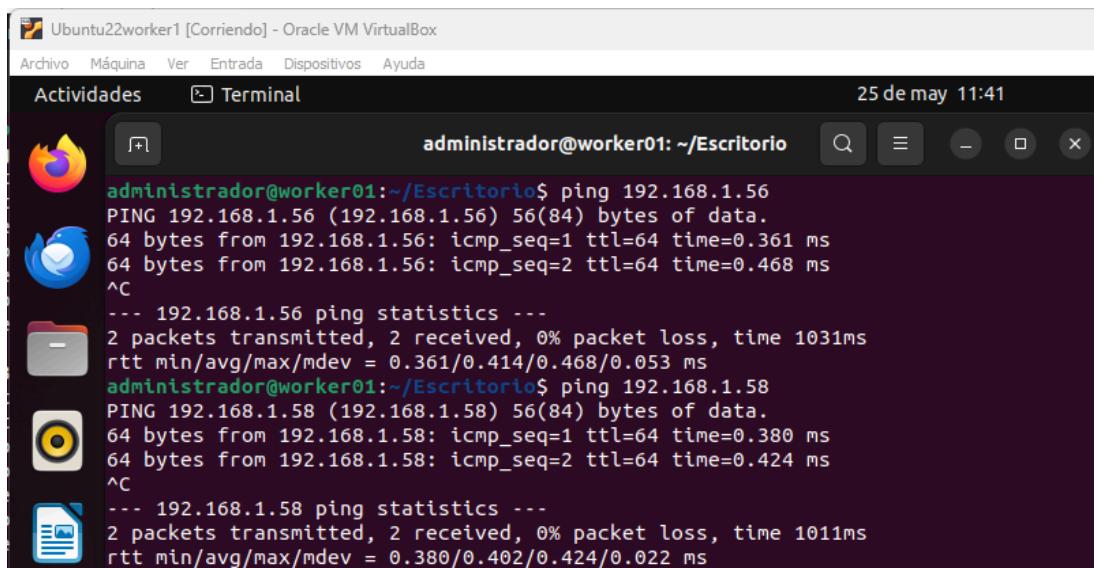
```
Ubuntu22worker1 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 11:38
administrador@worker01: ~/Escritorio
administrador@worker01:~/Escritorio$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.1.57 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::ee4b:6c87:6ac9:40da prefixlen 64 scopeid 0x20<link>
          ether 08:00:27:82:f6:37 txqueuelen 1000 (Ethernet)
            RX packets 11360 bytes 15787802 (15.7 MB)
            RX errors 0 dropped 17 overruns 0 frame 0
            TX packets 4244 bytes 365407 (365.4 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Worker2:



```
Ubuntu22worker2 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 11:40
administrador@worker02: ~/Escritorio
administrador@worker02:~/Escritorio$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.1.58 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::e3ab:a27b:7de5:173e prefixlen 64 scopeid 0x20<link>
          ether 08:00:27:54:44:e0 txqueuelen 1000 (Ethernet)
            RX packets 40598 bytes 59483763 (59.4 MB)
            RX errors 0 dropped 17 overruns 0 frame 0
            TX packets 11906 bytes 912286 (912.2 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Una vez hemos configurado las direcciones IP fijas en los nodos, comprobamos que se ven a través de nuestra red local y que pueden comunicarse mediante el comando ping.



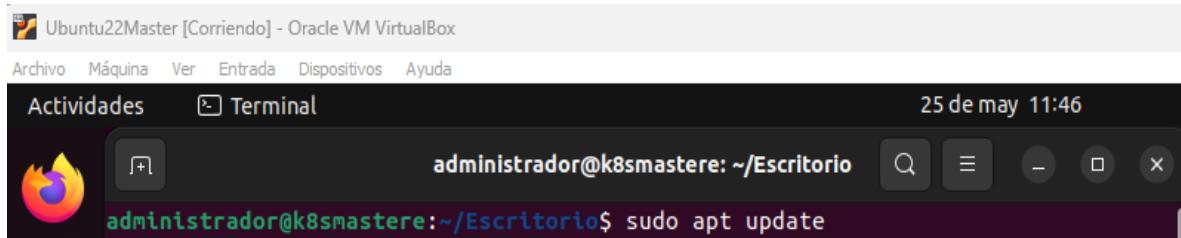
```
Ubuntu22worker1 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 11:41
administrador@worker01: ~/Escritorio
administrador@worker01:~/Escritorio$ ping 192.168.1.56
PING 192.168.1.56 (192.168.1.56) 56(84) bytes of data.
64 bytes from 192.168.1.56: icmp_seq=1 ttl=64 time=0.361 ms
64 bytes from 192.168.1.56: icmp_seq=2 ttl=64 time=0.468 ms
^C
--- 192.168.1.56 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.361/0.414/0.468/0.053 ms
administrador@worker01:~/Escritorio$ ping 192.168.1.58
PING 192.168.1.58 (192.168.1.58) 56(84) bytes of data.
64 bytes from 192.168.1.58: icmp_seq=1 ttl=64 time=0.380 ms
64 bytes from 192.168.1.58: icmp_seq=2 ttl=64 time=0.424 ms
^C
--- 192.168.1.58 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.380/0.402/0.424/0.022 ms
```

5.2 Instalación de Kubernetes

A partir de este punto, las configuraciones que hagan se harán tanto en el nodo principal (conocido como k8smastere) como en los nodos secundarios (conocidos como worker01 y worker02). Lo primero que hacemos es realizar una actualización de las máquinas con los siguientes comandos.

None

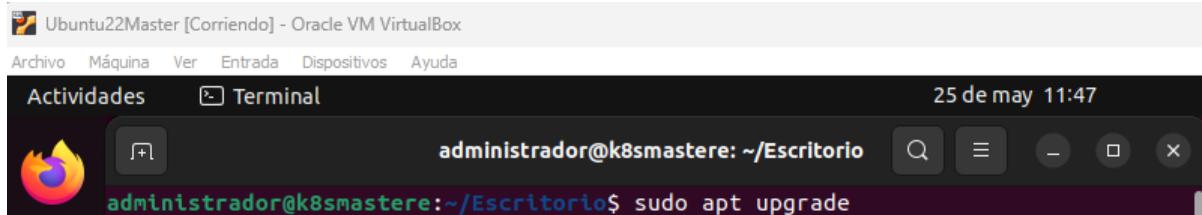
```
sudo apt update
```



The screenshot shows a terminal window titled 'Ubuntu22Master [Corriendo] - Oracle VM VirtualBox'. The window has a dark theme with white text. At the top, there's a menu bar with options like Archivo, Máquina, Ver, Entrada, Dispositivos, and Ayuda. Below the menu is a toolbar with icons for Actividades, Terminal, and a date/time indicator (25 de may 11:46). The terminal itself has a title bar showing 'administrador@k8smastere: ~/Escritorio'. The command 'sudo apt update' is typed into the terminal and is visible in green text at the bottom.

None

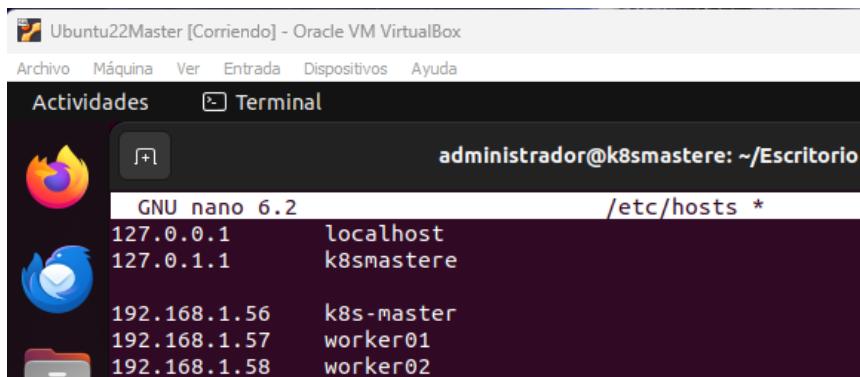
```
sudo apt upgrade
```



This screenshot is similar to the previous one, showing the same terminal window environment. The command 'sudo apt upgrade' is shown in the terminal window, indicating the continuation of the system update process.

Comenzamos con el proceso para crear los kubernetes en servicio local.

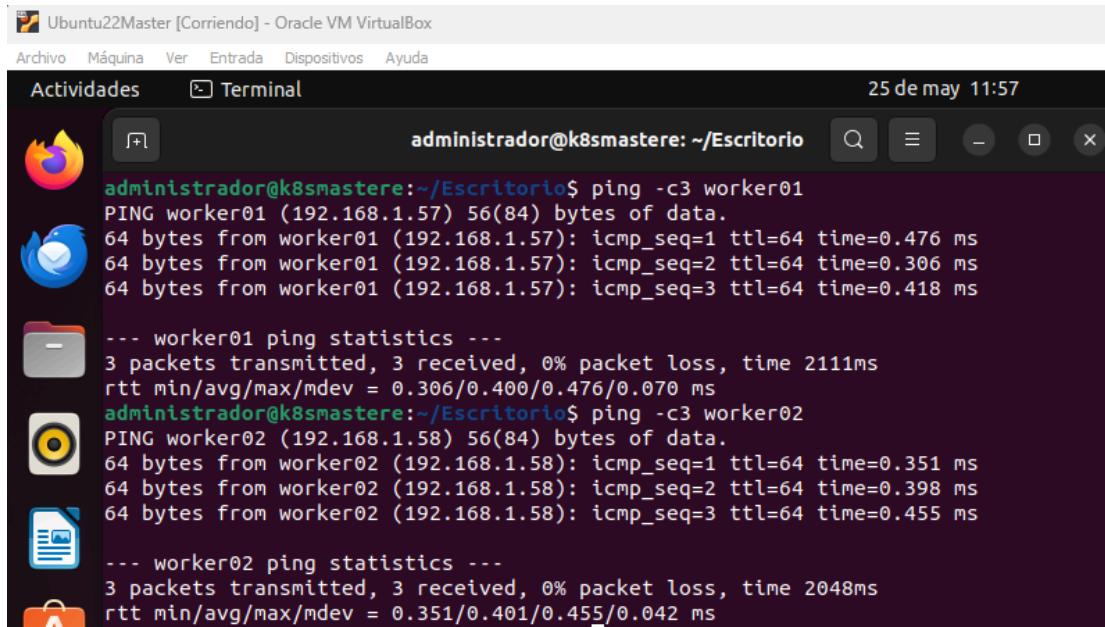
En todas las máquinas actualizamos el archivo /etc/hosts para añadir los nombres y las ips de las máquinas y poder localizarlas correctamente en red.



The screenshot shows a terminal window with the title 'GNU nano 6.2 /etc/hosts *'. The file contains the following entries:

IP	Nombre
127.0.0.1	localhost
127.0.1.1	k8smastere
192.168.1.56	k8s-master
192.168.1.57	worker01
192.168.1.58	worker02

Hacemos la prueba desde el nodo master de que las máquinas tienen visibilidad en la red por su nombre.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "administrador@k8smastere: ~/Escritorio". The terminal content shows the output of several ping commands:

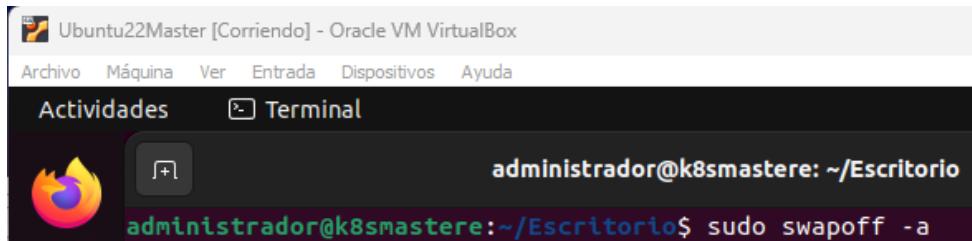
```
administrador@k8smastere:~/Escritorio$ ping -c3 worker01
PING worker01 (192.168.1.57) 56(84) bytes of data.
64 bytes from worker01 (192.168.1.57): icmp_seq=1 ttl=64 time=0.476 ms
64 bytes from worker01 (192.168.1.57): icmp_seq=2 ttl=64 time=0.306 ms
64 bytes from worker01 (192.168.1.57): icmp_seq=3 ttl=64 time=0.418 ms

--- worker01 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2111ms
rtt min/avg/max/mdev = 0.306/0.400/0.476/0.070 ms

administrador@k8smastere:~/Escritorio$ ping -c3 worker02
PING worker02 (192.168.1.58) 56(84) bytes of data.
64 bytes from worker02 (192.168.1.58): icmp_seq=1 ttl=64 time=0.351 ms
64 bytes from worker02 (192.168.1.58): icmp_seq=2 ttl=64 time=0.398 ms
64 bytes from worker02 (192.168.1.58): icmp_seq=3 ttl=64 time=0.455 ms

--- worker02 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2048ms
rtt min/avg/max/mdev = 0.351/0.401/0.455/0.042 ms
```

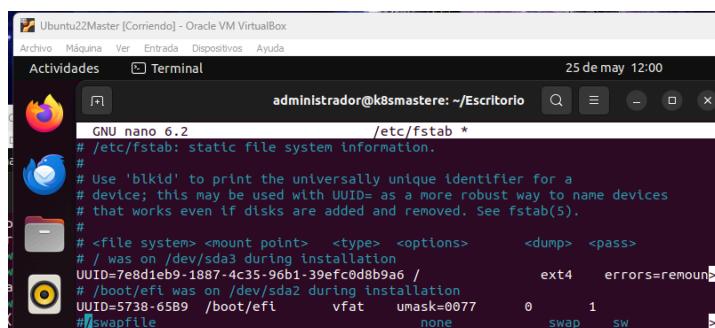
A continuación procederemos a deshabilitar la memoria swap en nuestras máquinas. Esto se hace porque Kubernetes asume que las aplicaciones que despleguemos en un futuro tienen acceso directo a la memoria física, buscando un rendimiento predecible. El uso de la memoria swap puede darnos latencias en el intercambio de memoria en el disco, por lo que es mejor deshabilitar dicha memoria para ganar un rendimiento consistente, una gestión de recursos precisa y una mayor escalabilidad en nuestro clúster.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "administrador@k8smastere: ~/Escritorio". The terminal content shows the command "sudo swapoff -a" being run:

```
administrador@k8smastere:~/Escritorio$ sudo swapoff -a
```

Editamos el fichero /etc/fstab y comentamos la última línea donde viene la palabra swapfile.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "administrador@k8smastere: ~/Escritorio". The terminal content shows the /etc/fstab file being edited with the nano editor. The swapfile line is commented out:

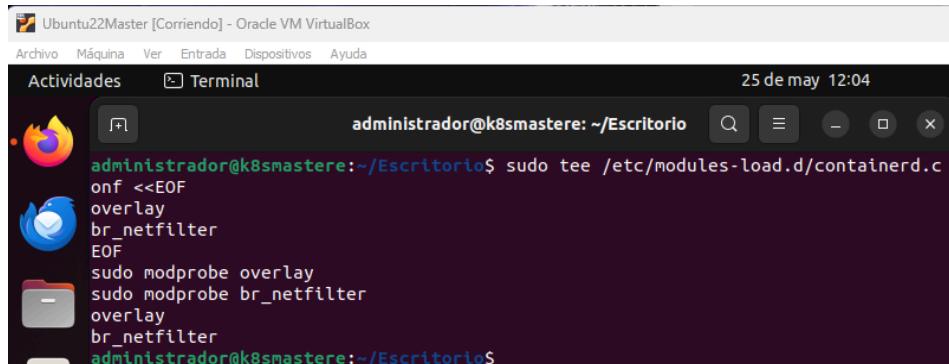
```
GNU nano 6.2          /etc/fstab *
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options>      <dump>   <pass>
# / was on /dev/sda3 during installation
UUID=7e8d1eb9-1887-4c35-96b1-39efc0db9a6 /          ext4    errors=remount
# /boot/efi was on /dev/sda2 during installation
UUID=5738-65B9 /boot/efi    vfat   umask=0077    0      1
#swapfile
```

Lo siguiente será agregar algunos parámetros al kernel para que kubernetes funcione correctamente.

None

```
sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
```

Esto se realiza para cargar los módulos “overlay” y “br_netfilter” en el kernel del sistema. Overlay es un módulo que soporta sistemas de archivos OverlayFS, los cuales son utilizados por los contenedores para capas de imágenes, mientras que br_netfilter permite filtrar el tráfico en puentes red, lo que es necesario para que nuestros contenedores se comuniquen entre sí.



None

```
sudo tee /etc/sysctl.d/99-kubernetes-cri.conf <<EOF
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF
```

```

actividades Terminal 25 de may 12:05
administrador@k8smastere: ~/Escritorio
administrador@k8smastere:~/Escritorio$ sudo tee /etc/sysctl.d/99-kubernetes-cri.conf <<EOF
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1

```

En cambio, estos comandos ponen varias configuraciones de red para nuestro CRI. `net.bridge.bridge-nf-call-iptables` permite la llamada de iptables para paquetes entrantes en interfaces de puente, lo que nos permite el enrutamiento de red dentro de los contenedores y los contenedores y el mundo exterior. `net.bridge.bridge-nf-call-ip6tables` hace lo mismo que el anterior, pero para protocolos IPv6, mientras que `net.ipv4.ip_forward` lo que hace es permitir el reenvío de paquetes IP entre interfaces de red en el kernel del sistema.

A continuación reiniciamos los parámetros del kernel para guardar estos cambios.

None

```
sudo sysctl --system
```

```

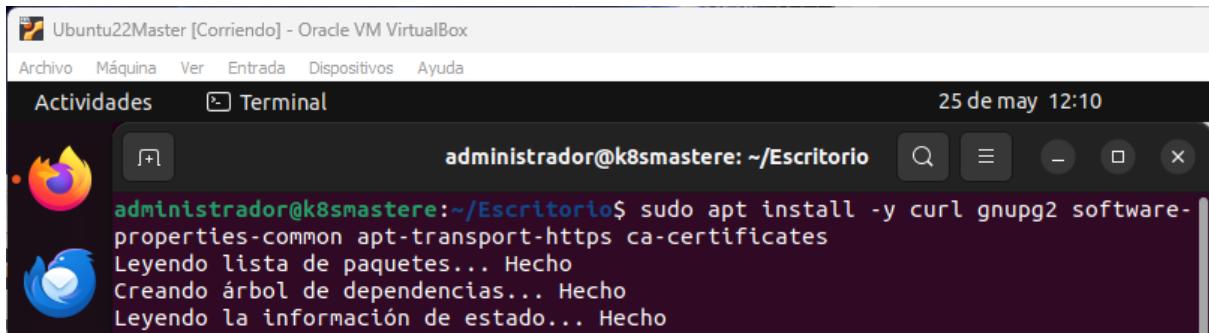
actividades Terminal 25 de may 12:07
administrador@k8smastere:~/Escritorio
administrador@k8smastere:~/Escritorio$ sudo sysctl --system
* Aplicando /etc/sysctl.d/10-console-messages.conf...
kernel.printk = 4 4 1 7
* Aplicando /etc/sysctl.d/10-ipv6-privacy.conf...
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
* Aplicando /etc/sysctl.d/10-kernel-hardening.conf...
kernel.kptr_restrict = 1
* Aplicando /etc/sysctl.d/10-magic-sysrq.conf...
kernel.sysrq = 176
* Aplicando /etc/sysctl.d/10-network-security.conf...
net.ipv4.conf.default.rp_filter = 2

```

Lo siguiente que hacemos es instalar curl. Curl nos permite realizar algunas funciones relacionadas con URL, como descargar contenidos de internet. Esto lo haremos para poder realizar descargas de ciertas herramientas que necesitaremos posteriormente.

None

```
sudo apt install -y curl gnupg2 software-properties-common
apt-transport-https ca-certificates
```

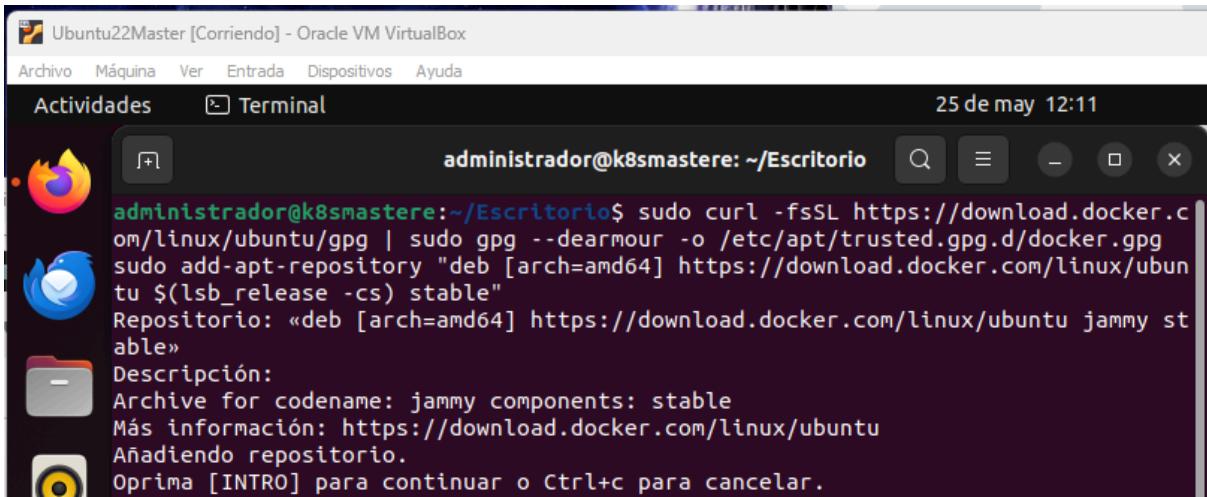


```
administrador@k8smastere:~/Escritorio$ sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
```

El siguiente paso es habilitar el repositorio de Containerd para poder descargarlo posteriormente.

None

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

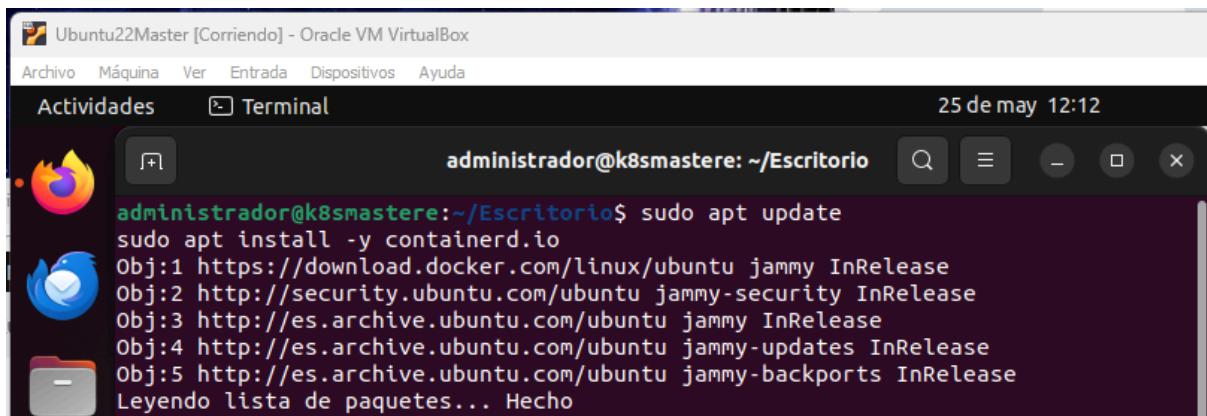


```
administrador@k8smastere:~/Escritorio$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: «deb [arch=amd64] https://download.docker.com/linux/ubuntu jammy stable»
Descripción:
Archive for codename: jammy components: stable
Más información: https://download.docker.com/linux/ubuntu
Añadiendo repositorio.
Oprima [INTRO] para continuar o Ctrl+c para cancelar.
```

Una vez hemos habilitado dicho repositorio, procedemos a su instalación.

None

```
sudo apt update
sudo apt install -y containerd.io
```

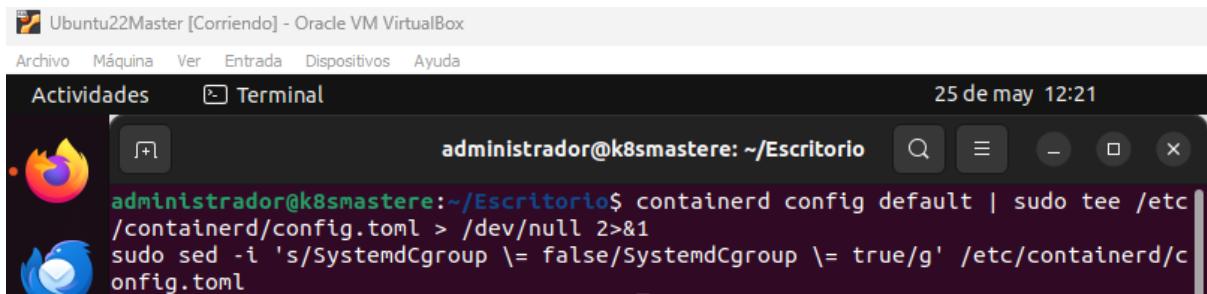


```
Ubuntu22Master [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 12:12
+ administrador@k8smastere: ~/Escritorio
administrador@k8smastere:~/Escritorio$ sudo apt update
sudo apt install -y containerd.io
Obj:1 https://download.docker.com/linux/ubuntu jammy InRelease
Obj:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Obj:4 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease
Obj:5 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease
Leyendo lista de paquetes... Hecho
```

A continuación configuraremos el Containerd con una configuración por defecto. Por supuesto, podemos alterar esta configuración para que se adapte a nuestras necesidades, pero en este caso no nos es necesario

None

```
containerd config default | sudo tee /etc/containerd/config.toml > /dev/null
2>&1
sudo sed -i 's/SystemdCgroup != false/SystemdCgroup != true/g'
/etc/containerd/config.toml
```

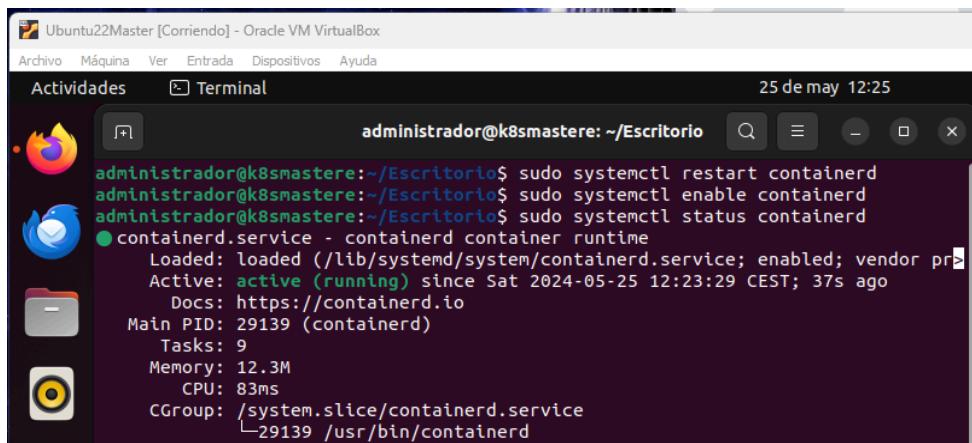


```
Ubuntu22Master [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 12:21
+ administrador@k8smastere: ~/Escritorio
administrador@k8smastere:~/Escritorio$ containerd config default | sudo tee /etc/
/containerd/config.toml > /dev/null 2>&1
sudo sed -i 's/SystemdCgroup != false/SystemdCgroup != true/g' /etc/containerd/c
onfig.toml
```

Para asegurarnos que Containerd utiliza el fichero de configuración que acabamos de crear, reiniciamos la aplicación, la activamos y comprobamos su estado para asegurarnos que no hay ningún error con Containerd.

None

```
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
```

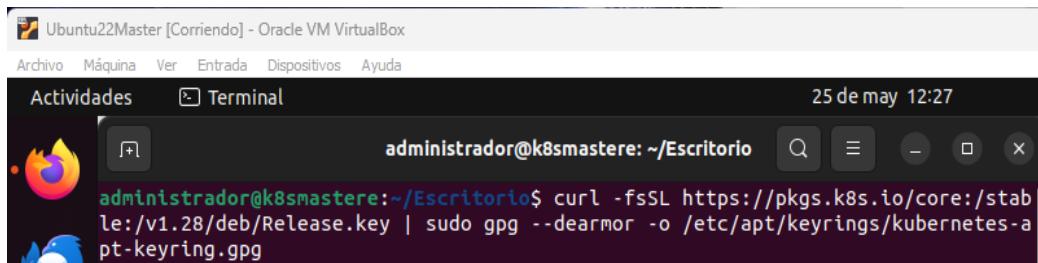


```
Ubuntu22Master [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 12:25
administrador@k8smastere: ~/Escritorio
administrador@k8smastere:~/Escritorio$ sudo systemctl restart containerd
administrador@k8smastere:~/Escritorio$ sudo systemctl enable containerd
administrador@k8smastere:~/Escritorio$ sudo systemctl status containerd
● containerd.service - containerd container runtime
   Loaded: loaded (/lib/systemd/system/containerd.service; enabled; vendor pr...
   Active: active (running) since Sat 2024-05-25 12:23:29 CEST; 37s ago
     Docs: https://containerd.io
      Main PID: 29139 (containerd)
        Tasks: 9
       Memory: 12.3M
          CPU: 83ms
        CGroup: /system.slice/containerd.service
                  └─29139 /usr/bin/containerd
```

Nuestro siguiente paso es agregar la llave GPG de Kubernetes.

None

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg
--dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

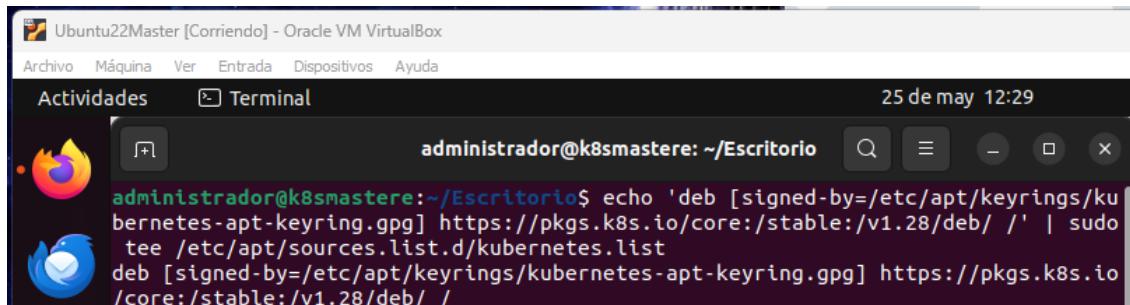


```
Ubuntu22Master [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 12:27
administrador@k8smastere: ~/Escritorio
administrador@k8smastere:~/Escritorio$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

A continuación agregamos el repositorio de Kubernetes.

None

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

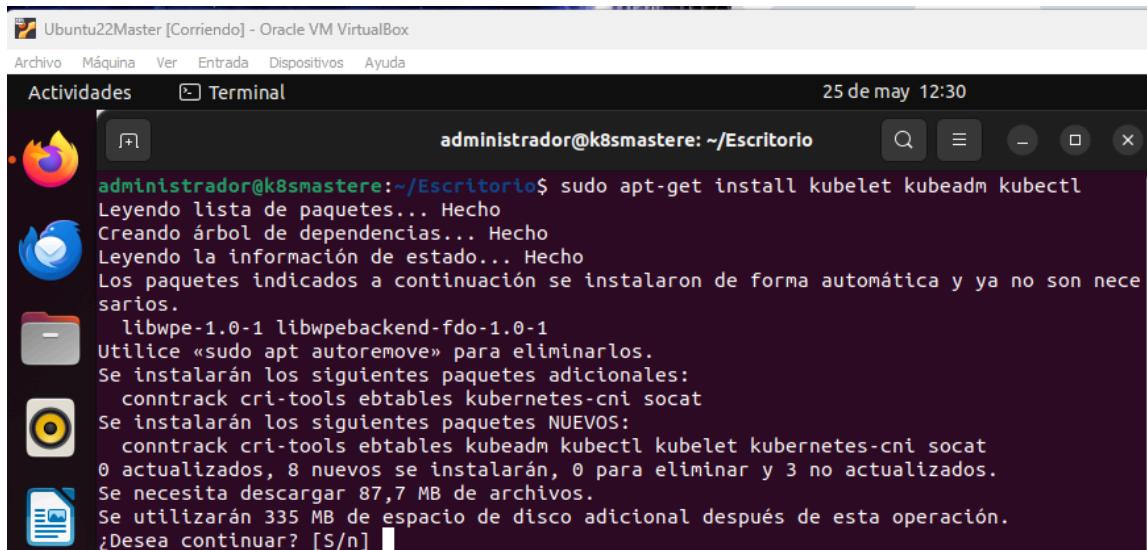


```
Ubuntu22Master [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 12:29
administrador@k8smastere: ~/Escritorio
administrador@k8smastere:~/Escritorio$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

Y por último instalamos kubelet, kubeadm y kubectl, aplicaciones necesarias para el funcionamiento de Kubernetes.

None

```
sudo apt-get install -y kubelet kubeadm kubectl
```

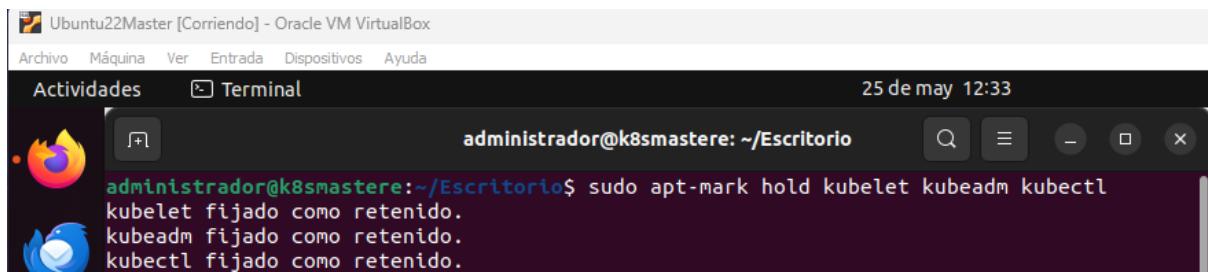


```
Ubuntu22Master [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 12:30
administrador@k8smastere: ~/Escritorio
administrador@k8smastere:~/Escritorio$ sudo apt-get install kubelet kubeadm kubectl
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libwpe-1.0-1 libwpebackend-fdo-1.0-1
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  conntrack cri-tools ebtables kubernetes-cni socat
Se instalarán los siguientes paquetes NUEVOS:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 actualizados, 8 nuevos se instalarán, 0 para eliminar y 3 no actualizados.
Se necesita descargar 87,7 MB de archivos.
Se utilizarán 335 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] ■
```

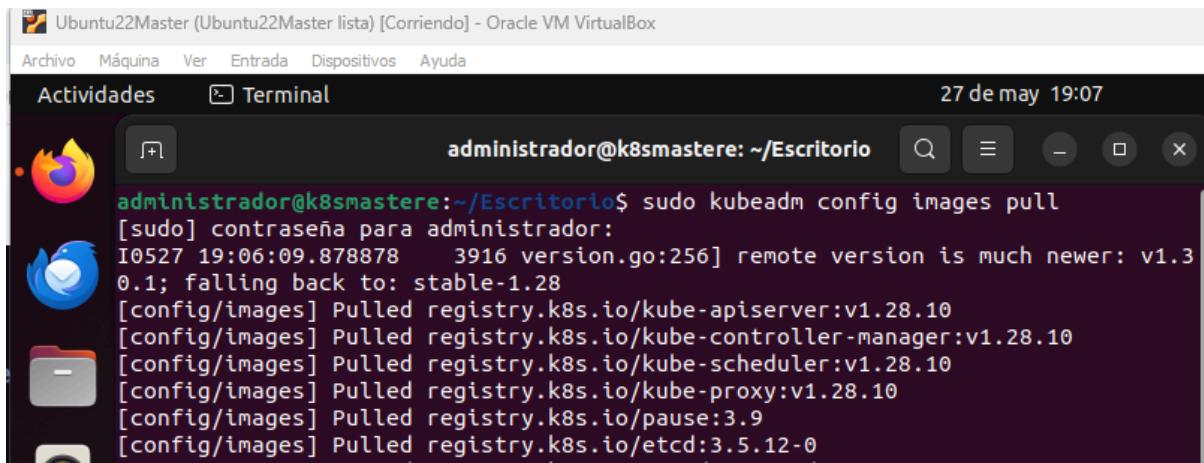
Desactivamos las actualizaciones automáticas para evitar problemas.

None

```
sudo apt-mark hold kubelet kubeadm kubectl
```



```
Ubuntu22Master [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de may 12:33
administrador@k8smastere: ~/Escritorio
administrador@k8smastere:~/Escritorio$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet fijado como retenido.
kubeadm fijado como retenido.
kubectl fijado como retenido.
```



```
administrador@k8smastere:~/Escritorio$ sudo kubeadm config images pull
[sudo] contraseña para administrador:
I0527 19:06:09.878878    3916 version.go:256] remote version is much newer: v1.3
0.1; falling back to: stable-1.28
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.28.10
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.28.10
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.28.10
[config/images] Pulled registry.k8s.io/kube-proxy:v1.28.10
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcfd:3.5.12-0
```

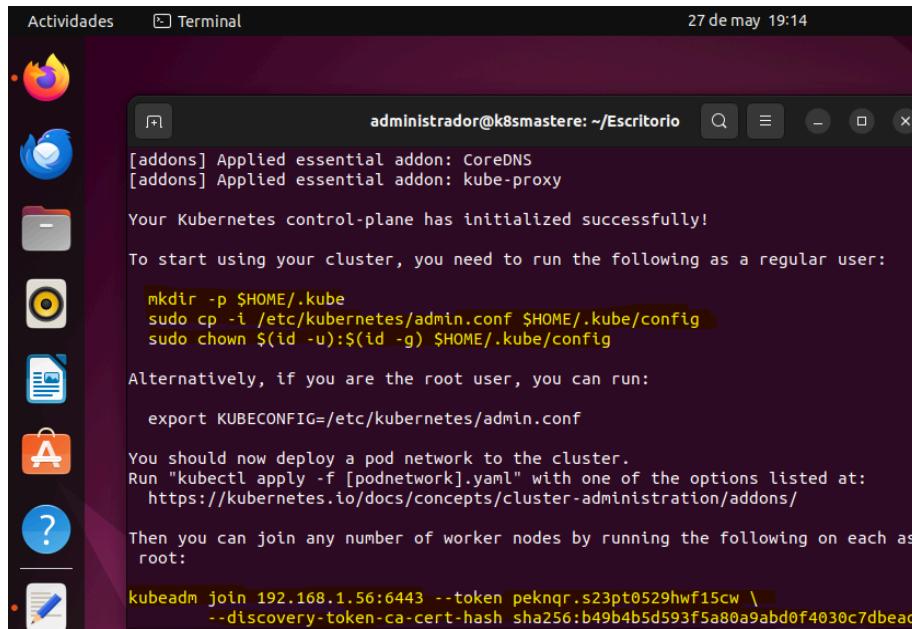
5.3 Despliegue y configuración de Kubernetes

El siguiente comando inicializará el clúster de Kubernetes usando kubeadm, en el que especificamos el rango de direcciones IP que se asignarán a los pods (CIDR). Además, se especifica la ruta del socket del runtime de contenedores (CRI) en este caso, se está utilizando containerd como runtime de contenedores. Especificamos la dirección IP que el servidor de la API (Máquina Master) utilizará para anunciarse a otros nodos del clúster, y por último se está configurando para la versión 1.28.9 de Kubernetes. Este comando debe ejecutarse únicamente en nuestro modo master, ya que es lo que hará que ese nodo funcione como maestro del clúster.

```
None
sudo      kubeadm      init      --pod-network-cidr=172.24.0.0/16
          --cri-socket=unix:///run/containerd/containerd.sock
          --apiserver-advertise-address=192.168.1.56 --kubernetes-version "1.28.9"
```

```
administrador@k8smastere:~/Escritorio$ sudo kubeadm init --pod-network-cidr=172.
24.0.0/16 --cri-socket=unix:///run/containerd/containerd.sock --apiserver-advert
ise-address=192.168.1.56 --kubernetes-version "1.28.9"
[init] Using Kubernetes version: v1.28.9
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
```

Es importante en este paso darle una red que no interfiera con los pods de Kubernetes. Nosotros le hemos asignado la 172.24.0.0/16 pero se puede utilizar cualquiera. Si todo se hace correctamente, veremos que se inicia y nos muestra lo siguiente.



The screenshot shows a terminal window titled "Terminal" with the command "administrador@k8smastere: ~/Escritorio". The window displays the output of a Kubernetes initialization script. It includes messages about CoreDNS and kube-proxy addons being applied, a success message for the control-plane initialization, instructions for regular users to copy the config file, and instructions for root users to set the KUBECONFIG environment variable. It also provides a link for deploying a pod network and instructions for joining worker nodes.

```
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

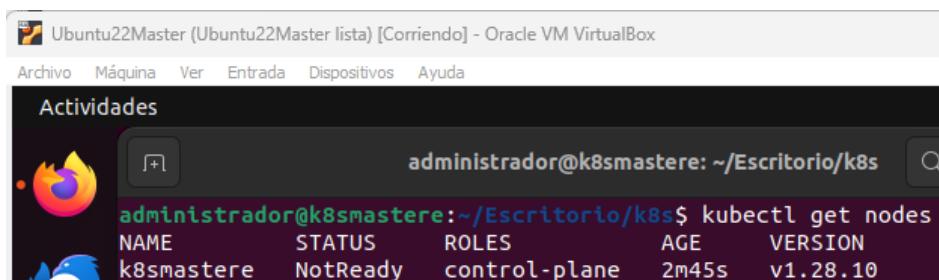
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as
root:

kubeadm join 192.168.1.56:6443 --token peknqr.s23pt0529hwf15cw \
--discovery-token-ca-cert-hash sha256:b49b4b5d593f5a80a9abd0f4030c7dbead
```

Como vemos, Kubernetes nos ayuda mucho con el proceso de instalación y configuración. Los primeros comandos en amarillo son para copiar el archivo de configuración por defecto de Kubernetes a una nueva carpeta llamada .kube y dándole los permisos del usuario root. Esto va a permitir que podamos interactuar con el clúster sin tener permisos de administrador con el usuario que deseemos. El segundo comando que se ve subrayado en amarillo lo utilizaremos más adelante para unir nodos a nuestro clúster, lo que facilita mucho todo el proceso.

Si una vez hemos iniciado kubernetes ejecutamos el comando get nodes, vemos nuestro Ubuntu-22-master como el único que hay.



The screenshot shows a terminal window titled "Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox" with the command "administrador@k8smastere: ~/Escritorio/k8s\$". The window displays the output of the "kubectl get nodes" command, which shows a single node named "k8smastere" with status "NotReady" and role "control-plane".

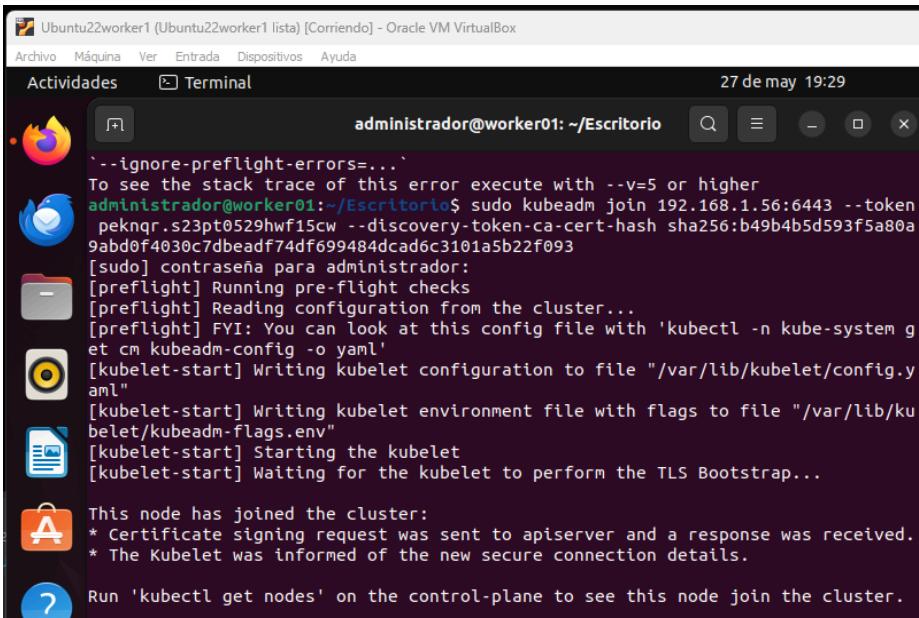
NAME	STATUS	ROLES	AGE	VERSION
k8smastere	NotReady	control-plane	2m45s	v1.28.10

El siguiente paso será añadir más nodos a nuestro clúster para que funcione como tal. Para ello, iremos a nuestro worker01 y worker02 y ejecutaremos el comando que nos ha dado antes el nodo master.

None

```
kubeadm join 192.168.1.56:6443 --token peknqr.s23pt0529hwf15cw \
-discovery-token-ca-cert-hash sha256:xxxxxx
```

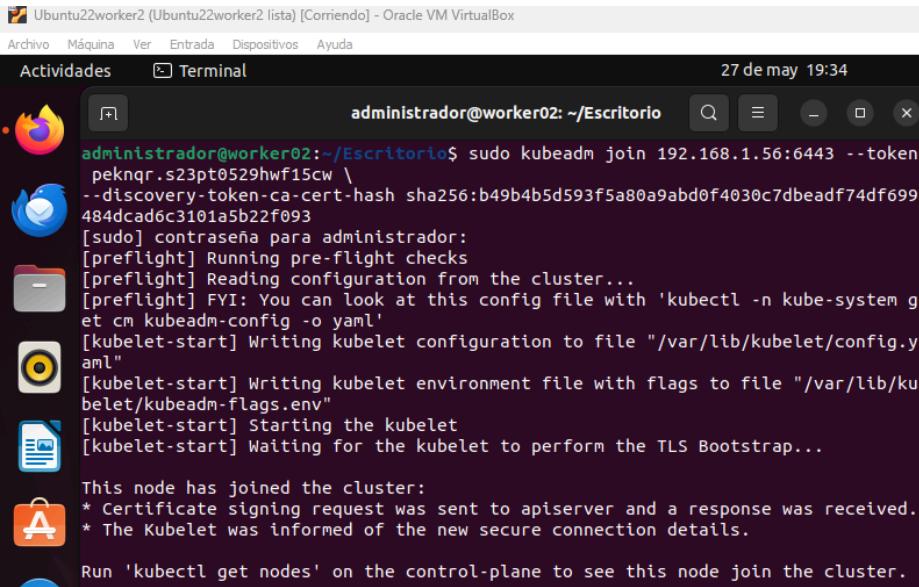
Worker01



```
--ignore-preflight-errors=...
To see the stack trace of this error execute with --v=5 or higher
administrador@worker01:~/Escritorio$ sudo kubeadm join 192.168.1.56:6443 --token
peknqr.s23pt0529hwf15cw --discovery-token-ca-cert-hash sha256:b49b4b5d593f5a80a
9abd0f4030c7dbeadf74df699484dcad6c3101a5b22f093
[sudo] contraseña para administrador:
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system g
et cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.y
aml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/ku
belet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Worker02



```
administrador@worker02:~/Escritorio$ sudo kubeadm join 192.168.1.56:6443 --token
peknqr.s23pt0529hwf15cw \
--discovery-token-ca-cert-hash sha256:b49b4b5d593f5a80a9abd0f4030c7dbeadf74df699
484dcad6c3101a5b22f093
[sudo] contraseña para administrador:
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system g
et cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.y
aml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/ku
belet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Una vez realizado, si volvemos a ejecutar el comando `kubectl get nodes` en el master, veremos nuestros nodos desplegados.

```

administrador@k8smastere:~/Escritorio$ kubectl get nodes
NAME      STATUS    ROLES   AGE     VERSION
k8smastere   NotReady control-plane 21m    v1.28.10
worker01     NotReady <none>    4m36s   v1.28.10
worker02     NotReady <none>    24s    v1.28.10

```

Sin embargo vemos que en el status pone NotReady, esto es debido a que debemos darles una interfaz de red de contenedores o CNI (Container Network Interface) para que los nodos puedan comunicarse entre ellos. Nosotros vamos a instalar Cálico Network Plugin, pero se pueden utilizar otras como Flannel.

Para ello lo primero que haremos será descargar de internet el .yaml de “tigera-operator”, el operador de cálico y redirigirlo a nuestro archivo tigera-operator.yaml

```

administrador@k8smastere:~/Escritorio/k8s$ curl -o https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/tigera-operator.yaml > tigera-operator.yaml

```

El siguiente paso será crear un pod en nuestro nodo master con este .yaml que acabamos de descargar para crear el servicio de Calico en nuestro clúster.

```

administrador@k8smastere:~/Escritorio/k8s$ kubectl create -f tigera-operator.yaml

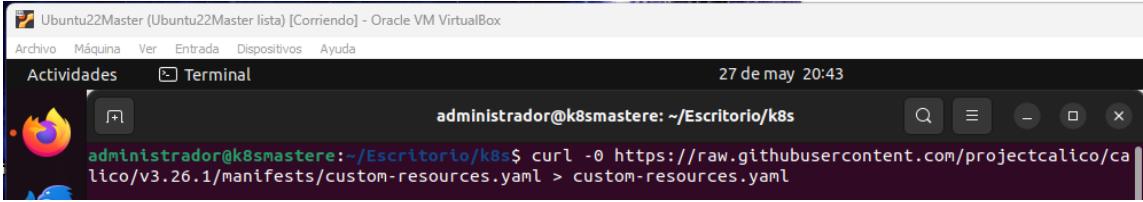
```

Una vez realizado este paso, debemos descargar ahora el .yaml que nos permite modificar los valores por defecto de tigera-operator para hacer que se adecuen a nuestras necesidades. Para ello lo primero es descargar de internet este .yaml y redigirlo al custom-resources.yaml.

None

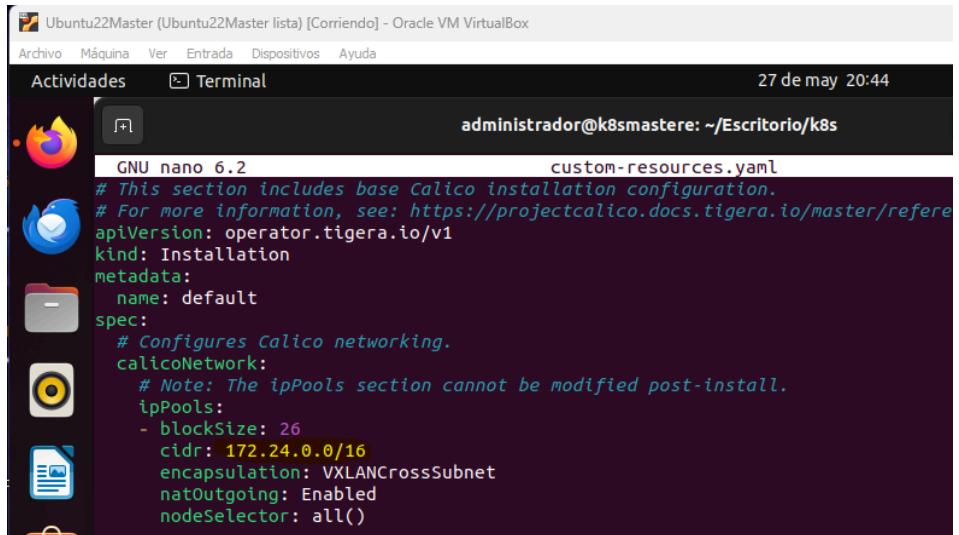
`curl -o`

<https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/custom-resources.yaml> > custom-resources.yaml



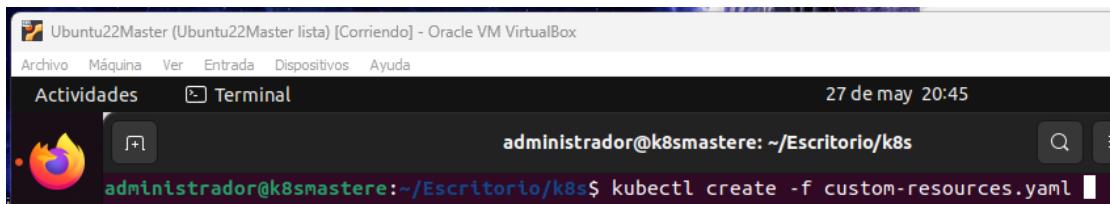
```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 27 de may 20:43
administrador@k8smastere:~/Escritorio/k8s$ curl -0 https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/custom-resources.yaml > custom-resources.yaml
```

Ahora que ya lo tenemos en nuestro master, debemos acceder al fichero con la herramienta nano y modificar el campo “cidr”. Por defecto, este campo viene en la red 192.168.0.0/16, pero nosotros debemos ajustarlo a la red que le hemos dado al principio al clúster, es decir la 172.24.0.0/16



```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 27 de may 20:44
administrador@k8smastere:~/Escritorio/k8s$ gnu nano 6.2
custom-resources.yaml
# This section includes base Calico installation configuration.
# For more information, see: https://projectcalico.docs.tigera.io/master/referenc...
apiVersion: operator.tigera.io/v1
kind: Installation
metadata:
  name: default
spec:
  # Configures Calico networking.
  calicoNetwork:
    # Note: The ipPools section cannot be modified post-install.
    ipPools:
      - blockSize: 26
        cidr: 172.24.0.0/16
        encapsulation: VXLANCrossSubnet
        natOutgoing: Enabled
        nodeSelector: all()
```

Una vez lo hemos modificado, es el momento de lanzar otro pod con nuestro custom-resources.yaml para que se apliquen nuestros ajustes al clúster.



```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 27 de may 20:45
administrador@k8smastere:~/Escritorio/k8s$ kubectl create -f custom-resources.yaml
```

Si volvemos a lanzar el comando para ver los nodos dentro del clúster, ya vemos como el estado de nuestros nodos es Ready.

None

`kubectl get nodes`

```

Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal
administrador@k8smastere: ~/Escritorio
administrador@k8smastere:~/Escritorio$ kubectl get nodes
NAME      STATUS   ROLES      AGE   VERSION
k8smastere   Ready    control-plane   31m   v1.28.10
worker01     Ready    <none>    14m   v1.28.10
worker02     Ready    <none>    10m   v1.28.10

```

Con esto nuestro clúster de Kubernetes está perfectamente operativo y listo para funcionar, por lo que el siguiente paso es desplegar aplicaciones para probarlo.

5.4 Despliegue de aplicaciones en Kubernetes

El siguiente paso en nuestro proyecto es desplegar algunas aplicaciones en Kubernetes y comprobar su funcionalidad. Para ello hemos decidido empezar con el Dashboard de Kubernetes. Se trata de una herramienta muy útil que nos ofrece una interfaz de usuario de nuestro cluster para monitorizar servicios o incluso desplegar nuevas aplicaciones.

El primer paso es desplegar el deployment del dashboard con un .yaml. En esta ocasión nosotros no generamos dicho .yaml, sino que lo lanzaremos directamente desde el Git de Kubernetes mediante el siguiente comando.

```

None
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml

```

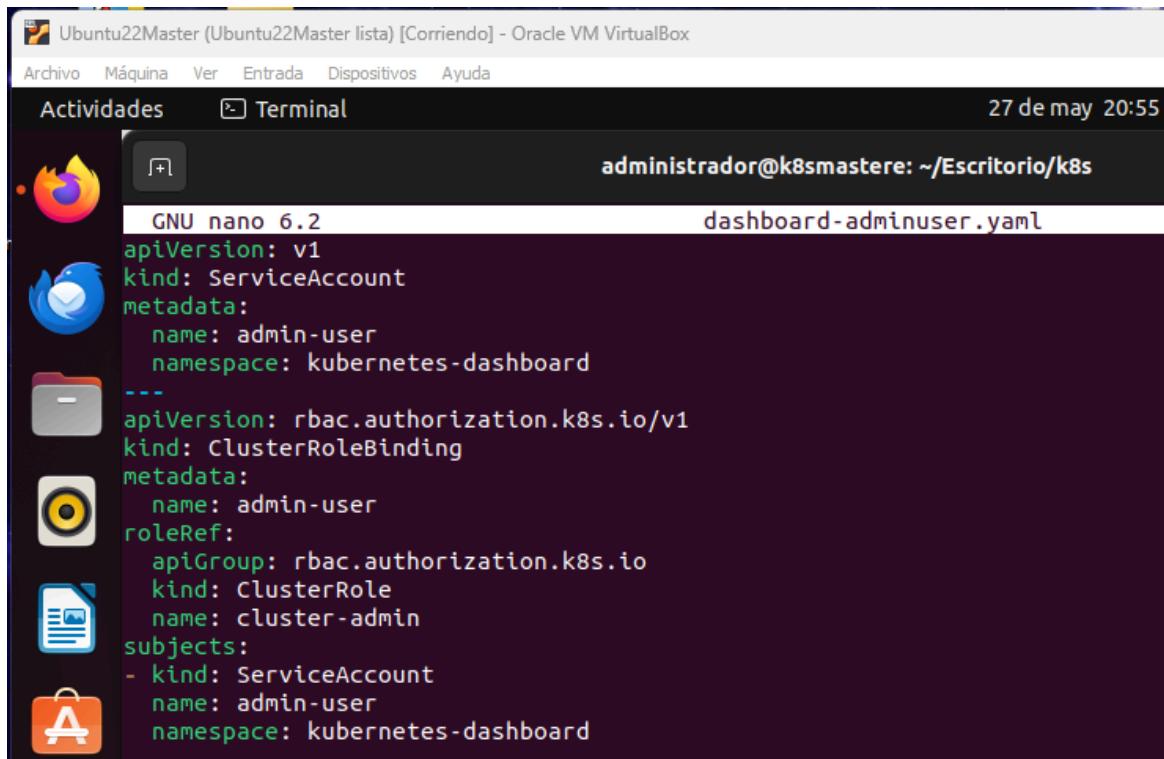
```

Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 27 de may 20:54
administrador@k8smastere: ~/Escritorio/k8s
administrador@k8smastere:~/Escritorio/k8s$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml

```

Esto lanza nuestro servicio, pero aún no podemos acceder a él, ya que vamos a necesitar ciertas credenciales para poder gestionar cluster desde el Dashboard. Por ello, primero debemos crear el servicio de cuenta y el rol de usuario.

Para crear estos servicios, creamos un archivo .yaml para definir un ServiceAccount y un ClusterRoleBinding que permita al usuario acceder al Dashboard.

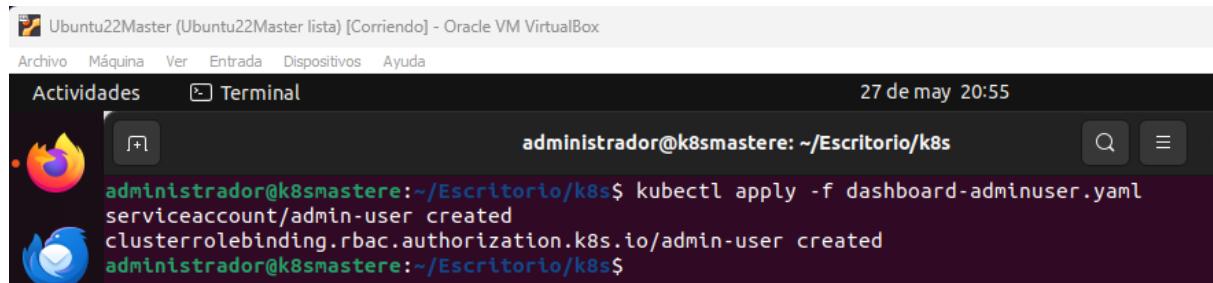


```
GNU nano 6.2
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

Una vez lo tenemos listo, procedemos a desplegarlo en nuestro clúster.

None

```
kubectl apply -f dashboard-adminuser.yaml
```

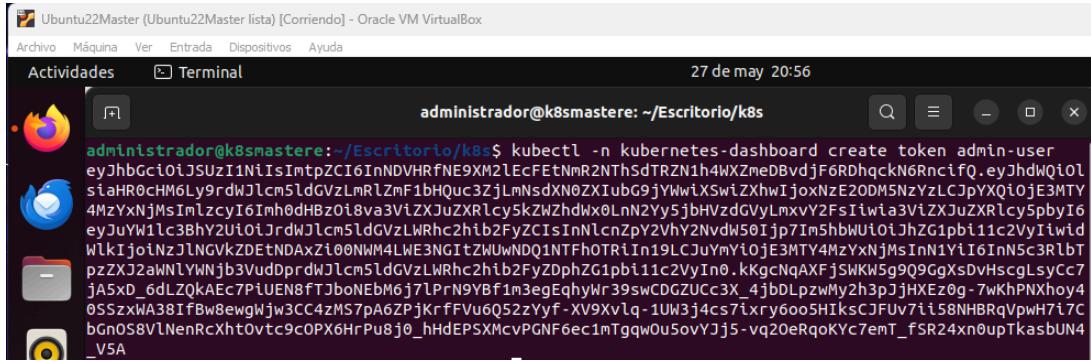


```
administrador@k8smastere:~/Escritorio/k8s$ kubectl apply -f dashboard-adminuser.yaml
serviceaccount/admin-user created
clusterrolebinding.rbac.authorization.k8s.io/admin-user created
administrador@k8smastere:~/Escritorio/k8s$
```

Con este nuevo servicio en marcha, podemos pedirle a Kubernetes que nos genere un token único de acceso para poder acceder al dashboard de Kubernetes

None

```
kubectl -n kubernetes-dashboard create token admin-user
```



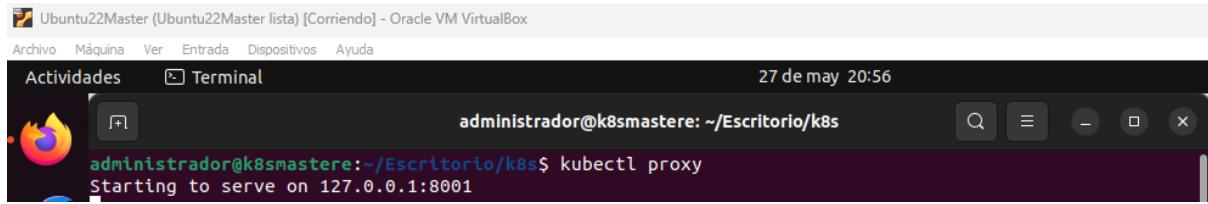
```

Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 27 de may 20:56
administrador@k8smastere: ~/Escritorio/k8s
administrador@k8smastere:~/Escritorio/k8s$ kubectl -n kubernetes-dashboard create token admin-user
eyJhbGciOiJSUzI1NiIsImtpZCI6InNDVHRfNE9XM2LecFetNmr2Nth5dTRZn1h4WXZmeDBvdjF6RDhqcKNGRncifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlc5ldGVzLmRlZmf1bHQuc3ZjLmNsdxN0ZXiubG9jYWwiXswizXhwIjoxNzE20DMSNzYzLCJpYXQiojE3MTY4MzYXNjMsImLzcyI6Imh0dHBzoi8va3VlZXJuXRlc5kZWhdWx0LnN2Yy5jbHVzdGvYlmxvY2FsIiwia3VlZXJuXRlc5pbvI6eyJuYW1lc3BhY2UiOijrdWJlc5ldGVzLWRhc2hib2FyZCISInNlcnPzY2VhY2NvdW50Ijp7Im5hbWUiOijhZG1pbii1c2VyIiwidWlkIjo1NzJ1NGVkdEtNDaxZl00NM4LWE3NGItZWuNDQ1NTfhOTRiIn19LCJuYmYlOjE3MTY4MzYxNjMsInN1iI6InN5c3RlbTpZzXj2aWNlyYNNjb3Vud0prdWJlc5ldGVzLWRhc2hib2FyZDphZG1pbii1c2VyIn0.kKqcNqAXFjSWKN5g909GgXsDvhscgLsyCc7jA5xd_6dLZQKAEC7PiUN8ftJboNEbM6j7lprN9YBf1m3egEghyWr39sCDGZUcc3X_4jbDlpzwMy2h3pJjhxEzog-7wKhPNXhoy4OSzxA38IfBw8ewgWjw3CC4zMS7pA6ZPjKrfFVu6Q52zyff-XV9Xvlq-1UW3j4cs7ixry6oo5HIksCJFUv7ii58NHBRqVpwH7i7CbGnOS8VLNenRcxht0vtc9c0PX6HrPu8j0_hDdEPSXMcvPGNF6ec1mTgqw0u5ovYJj5-vq20eRqoKYc7emT_fSR24xn0upTkasbUN4
_V5A

```

Para acceder al dashboard, primero debemos ejecutar el proxy de Kubernetes, lo que nos va a permitir acceder a la API de Kubernetes sin tener que preocuparnos de la autenticación y la seguridad del API, pues ya tenemos nuestro token de acceso.

None
kubectl proxy



```

Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 27 de may 20:56
administrador@k8smastere: ~/Escritorio/k8s
administrador@k8smastere:~/Escritorio/k8s$ kubectl proxy
Starting to serve on 127.0.0.1:8001

```

Y con esto ya podemos acceder mediante un navegador web a nuestro Dashboard de Kubernetes con el siguiente link.

None
<http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/>

The screenshot shows the Kubernetes Dashboard's 'Overview' page. On the left, a sidebar lists various resource types: Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Controladores de Replicación, Stateful Sets, Service, Ingresses, Ingress Classes, Services, Config Maps, Persistent Volume Claims, and Secrets. The main area has two sections: 'Cargas de trabajo' (Workloads) and 'Despliegues'. The 'Cargas de trabajo' section shows three green circles representing 'Despliegues' (Deployments) with counts of 1, 3, and 1 respectively. The 'Despliegues' section shows a table with one row for 'nginx-deployment', which is running 3 pods and was created 12 minutes ago.

Una vez dentro, podemos ver rápidamente varias funciones de nuestro clúster, como los servicios, los despliegues, los pods que se están ejecutando, etc. Esta interfaz incluso nos permite lanzar nuevos despliegues añadiendo únicamente el código que normalmente escribiríamos en nuestro archivo .yaml.

En este caso observamos cómo desplegamos una imagen de Nginx que va a tener 3 réplicas.

The screenshot shows the 'Crear' (Create) dialog box in the Kubernetes Dashboard. It has tabs for 'Crear desde entrada' (Create from input), 'Crear desde un fichero' (Create from file), and 'Crear desde un formulario' (Create from form). The 'Crear desde entrada' tab is selected. A text area contains a YAML template for a Deployment named 'nginx-deployment' with 3 replicas, selecting 'app: nginx' and using the 'nginx:latest' image. At the bottom are 'Cargar' (Load) and 'Cancelar' (Cancel) buttons.

```

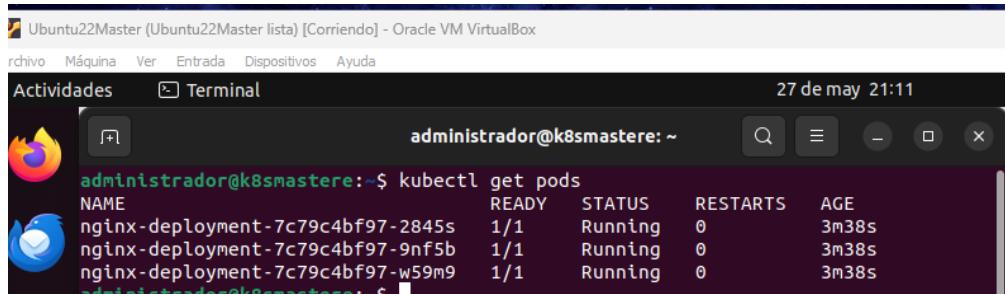
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: nginx-deployment
5   labels:
6     app: nginx
7 spec:
8   replicas: 3
9   selector:
10    matchLabels:
11      app: nginx
12   template:
13     metadata:
14       labels:
15         app: nginx
16     spec:
17       containers:
18         - name: nginx
19           image: nginx:latest
20         ports:
21           - containerPort: 80

```

Si ejecutamos:

```
None  
kubectl get pods
```

Vemos que las réplicas están corriendo, cada una en un pod distinto que ha creado Kubernetes.

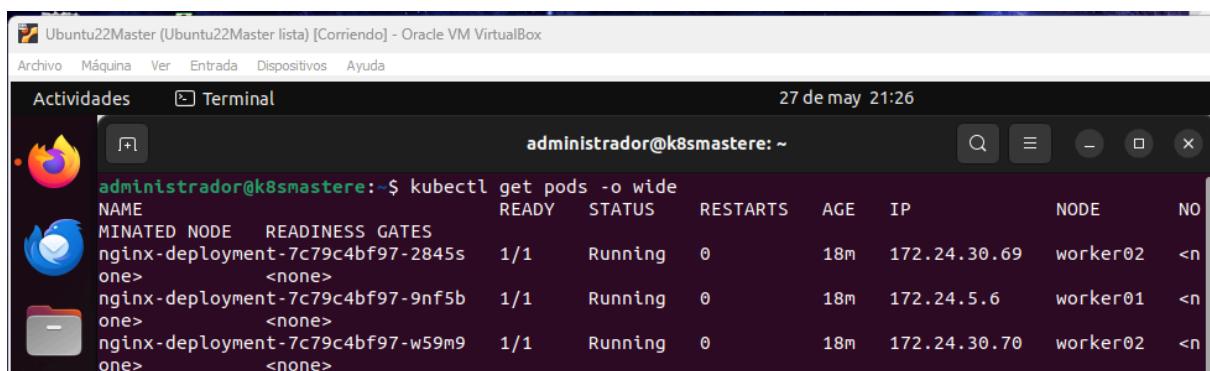


```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox  
Archivo Máquina Ver Entrada Dispositivos Ayuda  
Actividades Terminal 27 de may 21:11  
administrador@k8smastere: ~  
+ administrador@k8smastere:~$ kubectl get pods  
NAME READY STATUS RESTARTS AGE  
nginx-deployment-7c79c4bf97-2845s 1/1 Running 0 3m38s  
nginx-deployment-7c79c4bf97-9nf5b 1/1 Running 0 3m38s  
nginx-deployment-7c79c4bf97-w59m9 1/1 Running 0 3m38s  
administrador@k8smastere: ~
```

Podemos observar más detalles de estos pods creados si ejecutamos el comando:

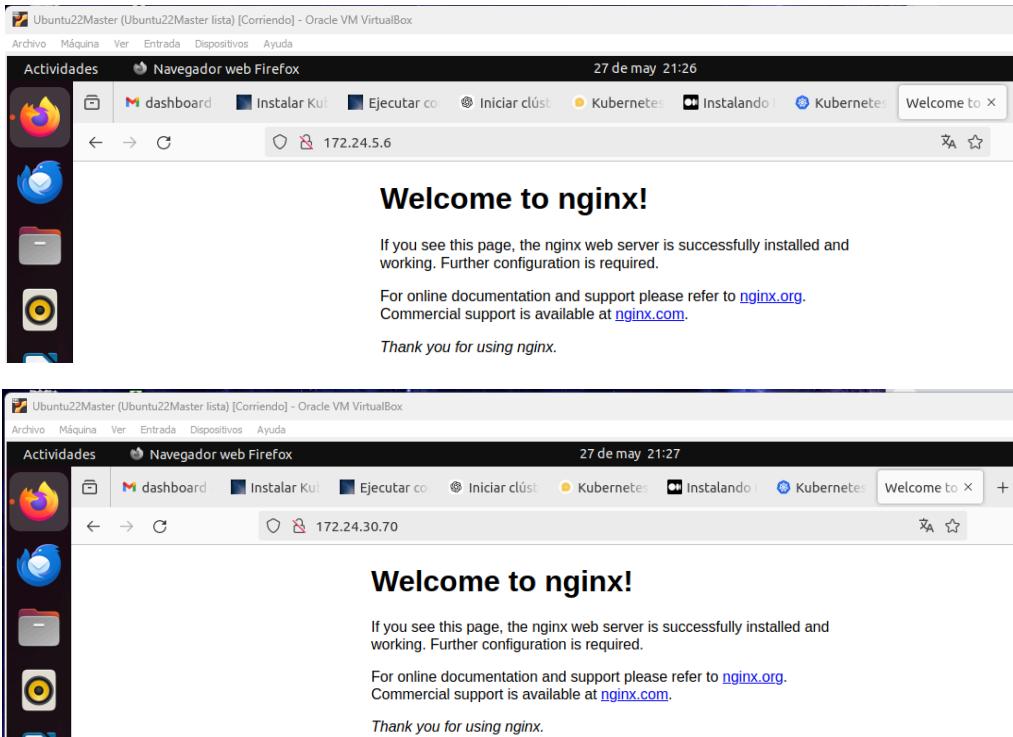
```
None  
kubectl get pods -o wide
```

Esto nos muestra la dirección IP que se ha asignado a cada pod, siempre dentro del rango de red CIDR que le dimos al principio de la instalación y que especificamos en nuestra red virtual Calico. También vemos como Kubernetes, de manera automática, realiza un balanceo de la carga ejecutando cada pod en un nodo distinto.



```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox  
Archivo Máquina Ver Entrada Dispositivos Ayuda  
Actividades Terminal 27 de may 21:26  
administrador@k8smastere: ~  
+ administrador@k8smastere:~$ kubectl get pods -o wide  
NAME MINATED NODE READINESS GATES READY STATUS RESTARTS AGE IP NODE NO  
nginx-deployment-7c79c4bf97-2845s one> <none> 1/1 Running 0 18m 172.24.30.69 worker02 <n  
one> <none>  
nginx-deployment-7c79c4bf97-9nf5b one> <none> 1/1 Running 0 18m 172.24.5.6 worker01 <n  
one> <none>  
nginx-deployment-7c79c4bf97-w59m9 one> <none> 1/1 Running 0 18m 172.24.30.70 worker02 <n  
one> <none>
```

Si hacemos una búsqueda en nuestro navegador de esas direcciones IP que nos muestran los nodos, observamos como ambas nos muestran la página por defecto que ofrece Nginx.



También podemos observar toda esa información desde nuestro Dashboard, donde se nos muestra el nombre del despliegue de Nginx y los pods que están corriendo en ese momento.

Nombre	Imágenes	Etiquetas	Pods
nginx-deployment	nginx:latest	app: nginx	3 / 3

Nombre	Imágenes	Etiquetas	Nodo	Estado	Reinicios	Utilización de CPU (núcleos)	Utiliza memo
nginx-deployment-7c79c4bf5	nginx:latest	app: nginx pod-template-has h: 7c79c4bf97	worker02	Running	0	-	-
nginx-deployment-7c79c4bf5	nginx:latest	app: nginx pod-template-has h: 7c79c4bf97	worker01	Running	0	-	-
nginx-deployment-7c79c4bf5-w59m9	nginx:latest	app: nginx pod-template-has h: 7c79c4bf97	worker02	Running	0	-	-

5.5 Pruebas de alta disponibilidad

Ya hemos comprobado como Kubernetes despliega aplicaciones y realiza un balanceo de la carga de los pods creados. El siguiente paso es comprobar la alta disponibilidad que nos ofrece dicho cluster en caso de que algún nodo de problemas.

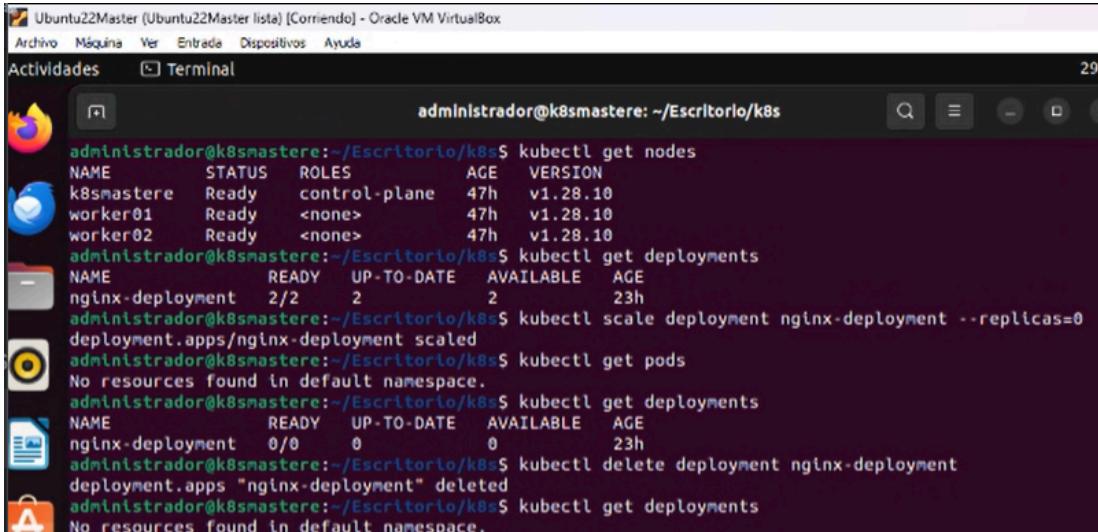
Lo primero que debemos hacer es eliminar el deployment de Nginx que hemos generado en el Dashboard de Kubernetes para lanzar un nuevo despliegue modificado a nuestro gusto.

Para ello utilizaremos los siguientes comandos:

```
None
```

```
kubectl scale deployment nginx-deployment --replicas=0
```

```
kubectl delete deployment nginx-deployment
```



```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 29
administrador@k8smastere: ~/Escritorio/k8s$ kubectl get nodes
NAME      STATUS   ROLES      AGE      VERSION
k8smastere   Ready    control-plane   47h     v1.28.10
worker01     Ready    <none>     47h     v1.28.10
worker02     Ready    <none>     47h     v1.28.10
administrador@k8smastere:~/Escritorio/k8s$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2           2           23h
administrador@k8smastere:~/Escritorio/k8s$ kubectl scale deployment nginx-deployment --replicas=0
deployment.apps/nginx-deployment scaled
administrador@k8smastere:~/Escritorio/k8s$ kubectl get pods
No resources found in default namespace.
administrador@k8smastere:~/Escritorio/k8s$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   0/0     0           0           23h
administrador@k8smastere:~/Escritorio/k8s$ kubectl delete deployment nginx-deployment
deployment.apps "nginx-deployment" deleted
administrador@k8smastere:~/Escritorio/k8s$ kubectl get deployments
No resources found in default namespace.
```

Con estos comandos lo que conseguimos es hacer que el deployment no lance ningún pod y elimine aquellos que están creados, aunque se mantiene a la espera de nuestras órdenes para volver a lanzar pods. Por ello utilizamos después el “delete deployment” para eliminarlo por completo.

Una vez hemos eliminado el despliegue, debemos crear un Configmap en Kubernetes. Esto se hace para modificar la página por defecto que trae Nginx en el pod y utilizar la nuestra. Además, si en un futuro deseamos cambiar el código para actualizarlo o arreglar bugs, no es necesario eliminar el deployment, basta con actualizar el código del configmap y el solo se encarga de actualizar la información de los pods que está utilizando el dicho deployment.

```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal
administrador@k8smastere: ~/Escritorio/k8s$ nano configmap.yaml
```

```
GNU nano 6.2 configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: html-config
data:
  index.html: |
    <!DOCTYPE html>
    <html lang="es">
      <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Demostración TFG Kubernetes</title>
        <style>
          body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
            color: #333;
            display: flex;
```

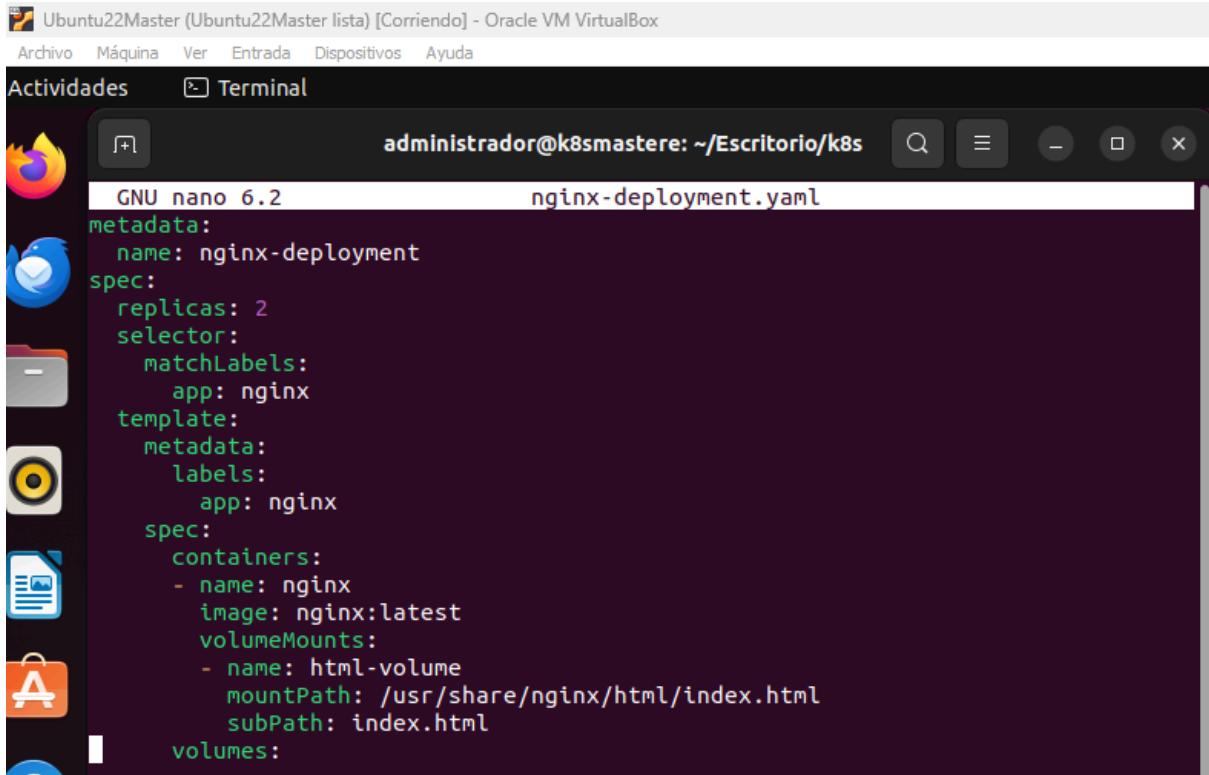
Una vez tenemos completo el archivo, procedemos a ejecutarlo:

None

```
kubectl apply -f configmap.yaml
```

```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal
administrador@k8smastere: ~/Escritorio/k8s$ kubectl apply -f configmap.yaml
```

El siguiente paso es diseñar nuestro nuevo deployment de Nginx, al cual le decimos que debe tener dos pods funcionando de manera continua en el apartado réplicas y que debe coger la información html de la ruta /usr/share/nginx/html/index.html

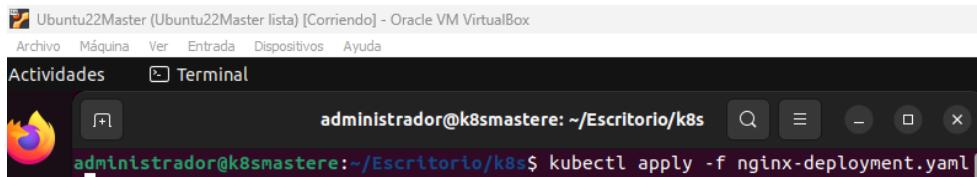


```
GNU nano 6.2          nginx-deployment.yaml
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          volumeMounts:
            - name: html-volume
              mountPath: /usr/share/nginx/html/index.html
      volumes:
```

Procedemos a aplicar el servicio:

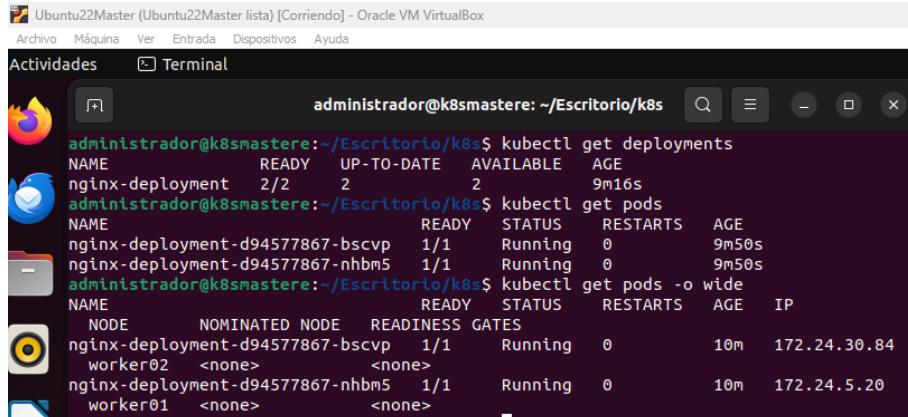
None

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



```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal
administrador@k8smastere: ~/Escritorio/k8s$ kubectl apply -f nginx-deployment.yaml
```

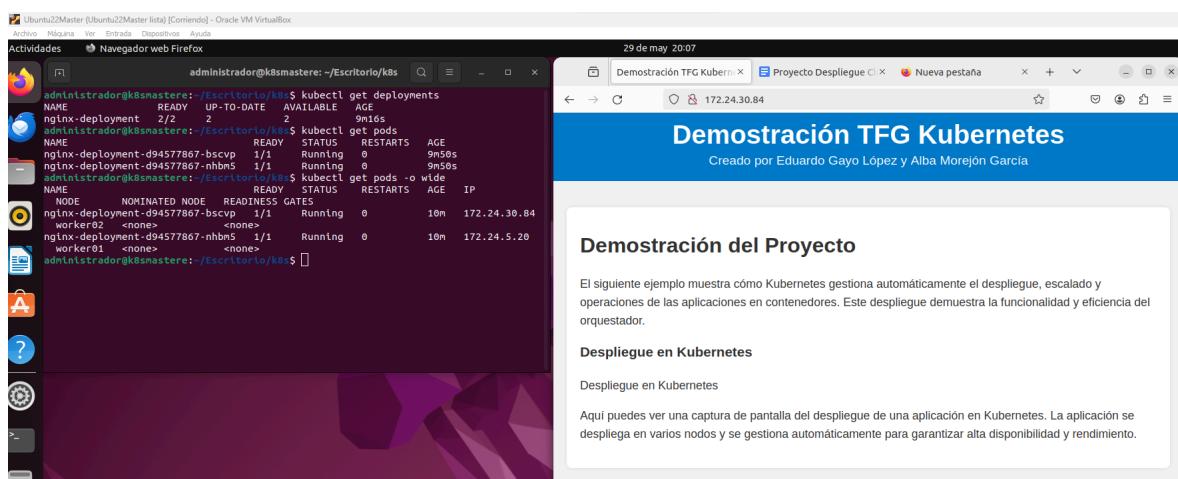
Ejecutamos los comandos `kubectl get deployments`, `kubectl get pods` y `kubectl get pods -o wide` para asegurarnos que tanto el despliegue como los pods funcionan correctamente.



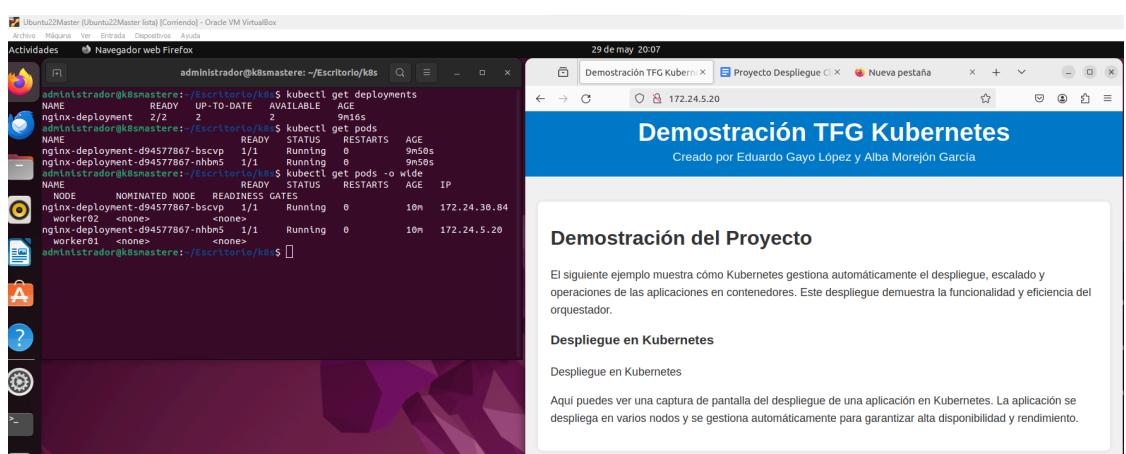
```
Ubuntu22Master (Ubuntu22Master lista) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal
administrador@k8smastere: ~/Escritorio/k8s$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2           2           9m16s
administrador@k8smastere: ~/Escritorio/k8s$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
nginx-deployment-d94577867-bscvp  1/1     Running   0          9m50s
nginx-deployment-d94577867-nhbms  1/1     Running   0          9m50s
administrador@k8smastere: ~/Escritorio/k8s$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP
      NODE    NOMINATED-NODE  READINESS GATES
nginx-deployment-d94577867-bscvp  1/1     Running   0          10m   172.24.30.84
  worker02 <none>        <none>
nginx-deployment-d94577867-nhbms  1/1     Running   0          10m   172.24.5.20
  worker01 <none>        <none>
```

Al ver que los pods se ejecutan correctamente, hacemos la prueba de entrar a las IPs de los pods para ver si el resultado mostrado es el deseado.

Ejemplo: <http://172.24.30.84>



Ejemplo: <http://172.24.5.20>



Ahora que ya hemos comprobado que funciona, es el momento de comprobar la alta disponibilidad. Para ello vamos a tumbar el nodo Worker01.

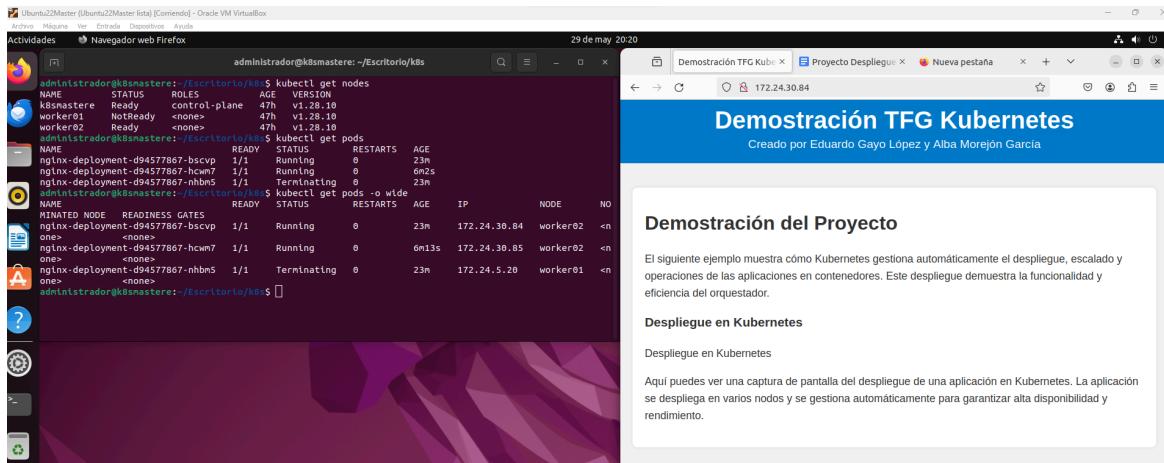
Como el despliegue especificaba que debía haber dos pods ejecutándose de manera continua, Kubernetes decide lanzar este segundo pod en el nodo Worker02, ya que no encuentra respuesta por parte del Worker01. Si observamos la captura de pantalla, vemos que al ejecutar el comando kubectl get nodes nos dice que el nodo Worker01 se encuentra en estado “NotReady” ya que, aunque sigue dentro del clúster, el nodo K8master no recibe ninguna respuesta suya.

Si nos fijamos en el resultado de kubectl get pods, observamos que hay 3 pods en este momento. Los dos primeros son los creados por el deployment que nos muestran nuestra página web, mientras que el tercero observamos que se encuentra en estado “terminating”, esto es porque es el pod que se encontraba desplegado en el Worker01 y, al tumbar la máquina, se ha quedado “muerto”.

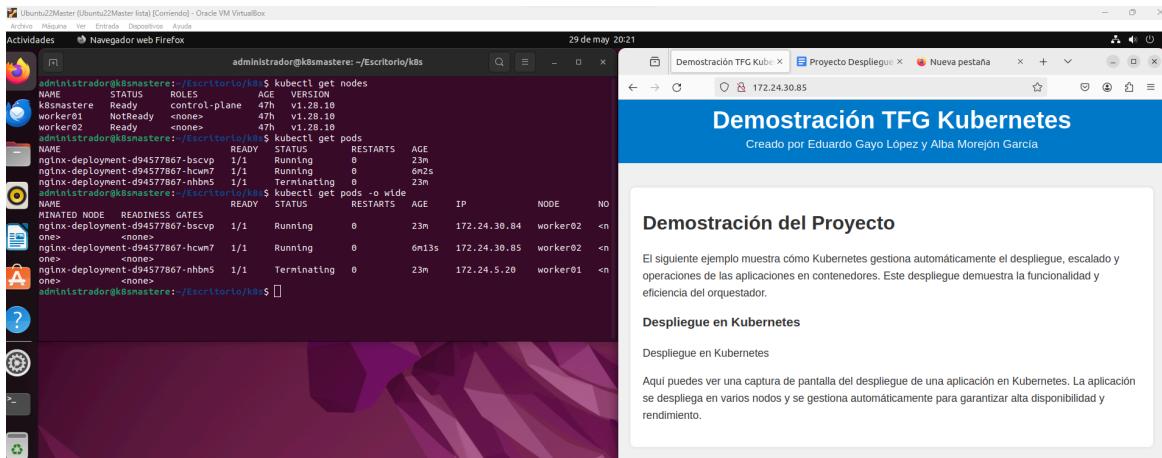
Si observamos el tercer comando, kubectl get pods -o wide, vemos que los dos primeros pods se están ejecutando en el Worker02 ya que es el único que se encuentra activado en este momento, mientras el tercero, del cual habíamos dicho que estaba en estado “terminating” se encuentra en el nodo Worker01

Si accedemos a las IPs de los nodos, observamos que nuestra página sigue en activo en cada uno de ellos. Hay que acceder a través del protocolo HTTP ya que el deployment se ha configurado de esa manera utilizando el puerto 80 y sin certificados.

Ejemplo: <http://172.24.30.84>



<http://172.24.30.85>



Con esto queda demostrada la alta disponibilidad del clúster de kubernetes que hemos desplegado, donde vemos que, aunque falle una de las máquinas, se despliegan los pods requeridos en la otra sin ningún problema, manteniendo así el servicio operativo a pesar de la incidencia.

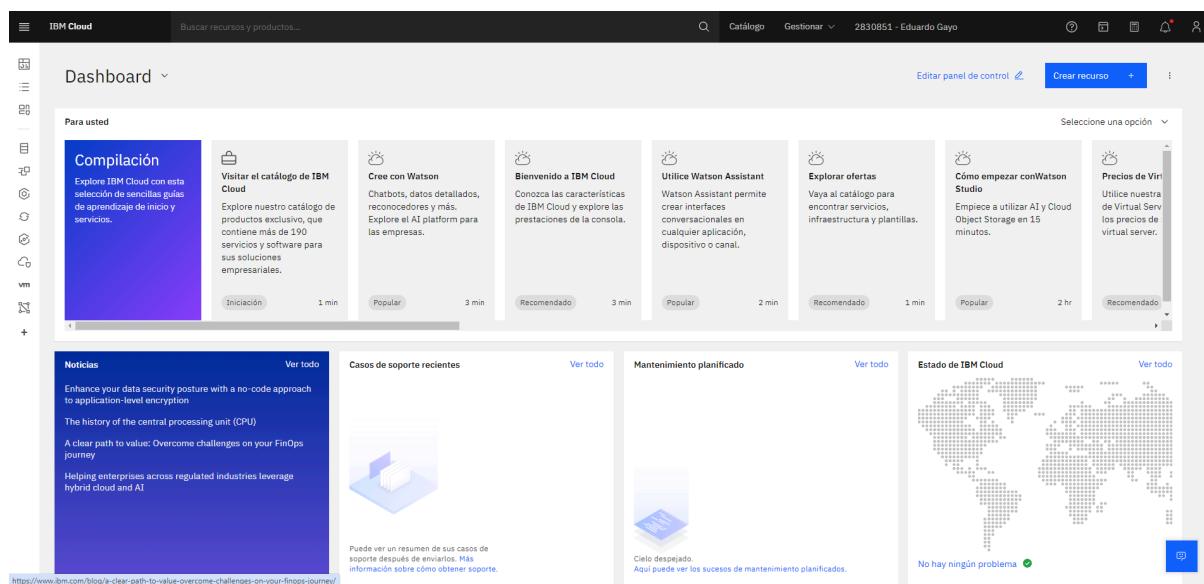
6. Despliegue del clúster de Kubernetes en IBMCloud

Una vez hemos desplegado nuestro clúster en local y hemos hecho las pruebas necesarias, vamos a ir paso más allá y desplegaremos un nuevo clúster en una plataforma en la nube. Esto supone una comodidad para las empresas ya que no tienen que ocuparse del mantenimiento de las máquinas físicas, ni de la configuración inicial, ya que es el proveedor de servicios se encarga del mantenimiento y de facilitar servicios para el despliegue. Sin embargo, la contratación de estos servicios supone un coste mensual que, dependiendo de las configuraciones que necesitemos, puede aumentar hasta los miles de euros al mes.

Existen una gran cantidad de proveedores de servicios que ofrecen máquinas virtuales en la nube o servicios de clúster de Kubernetes en la nube. Amazon Web Services, Google Cloud o IBM Cloud son solo algunos ejemplos. Nosotros vamos a escoger IBM Cloud como nuestro proveedor para realizar pruebas. El principal motivo de escoger esta opción por encima de otras es que IBM Cloud ofrece con el registro en su página 200 euros para que puedas probar sus servicios y ver si se adecuan a tus necesidades. Esto nos permitirá desplegar un pequeño clúster con dos nodos para realizar pruebas, pero algunas de las configuraciones más interesantes no podremos verlas, ya que requieren un coste superior a nuestro saldo, por lo que solo arañamos la superficie de todo lo que puede ofrecer este proveedor de servicios.

6.1 Configuración inicial

Lo primero que debemos hacer es darnos de alta en el servicio de IBM Cloud y llenar todos los campos necesarios. Una vez el proceso de registro ha sido completado, podemos acceder a nuestro Dashboard de IBM Cloud, donde tenemos una enorme cantidad de servicios en la nube para elegir. Vamos a pulsar en “Crear recurso” situado en la parte superior izquierda de la pantalla y ahí vamos a buscar “IBM Kubernetes Service” para crear nuestro clúster.



The screenshot shows the IBM Cloud dashboard with a dark header bar. On the left, there's a sidebar with icons for services like VM, RDS, Watson, and others. The main area has a "Para usted" section with cards for "Compilación", "Visitar el catálogo de IBM Cloud", "Cree con Watson", "Bienvenido a IBM Cloud", "Utilice Watson Assistant", "Explorar ofertas", "Cómo empezar con Watson Studio", and "Precios de Virtual Server". Below this, there are sections for "Noticias", "Casos de soporte recientes", "Mantenimiento planificado", and "Estado de IBM Cloud". A world map in the "Estado de IBM Cloud" section shows green dots indicating no problems.

Una vez hemos accedido al servicio de Kubernetes nos despliega un amplio menú de opciones que podemos escoger para personalizar nuestro clúster y que se adapte a nuestras necesidades. La primera opción que nos da es escoger la red en la que deseamos trabajar. Podemos crear nuestra red privada virtual o ejecutar el clúster sobre VLAN nativas de IBM. Vamos a escoger la segunda opción por un tema económico, pero es interesante ver las opciones que te permite personalizar en la red privada.

Catálogo /

Kubernetes Service

Autor: IBM • Documentos • Documentos de API

Crear Acerca de

Entregue sus aplicaciones más rápidamente en las nubes con Red Hat OpenShift



Infraestructura

Elija en qué red y entorno de cálculo desea que se ejecute el clúster. [Obtenga más información sobre las diferencias.](#)

VPC Cree una red virtual totalmente personalizable y definida por software con un aislamiento superior mediante IBM Cloud VPC.

Clásica Ejecute el clúster con subredes nativas y redes VLAN en nuestra infraestructura clásica.

ICMPv4 o SSH y crear hasta 3 subredes, lo que se adapta a las necesidades de cada usuario. Podemos darle una ubicación geográfica, poner etiquetas, permitir o denegar peticiones.

Crear Virtual Private Cloud

Ubicación ①
Seleccione la ubicación donde desea crear el nube privada virtual.

Geografía
 Europe

Región
 Madrid

Detalles

Nombre

Utilice solo caracteres alfanuméricos en minúsculas y guiones (sin espacios).

Grupo de recursos ①

[Ver todos los grupos de recursos](#)

Etiquetas (opcional) ①

Etiquetas (opcional) ⓘ
Ejemplos: env:dev, version-1

Etiquetas de gestión de acceso (opcional) ⓘ
Ejemplos: access:dev, proj:version-1

Lista de control de accesos predeterminada de Grupo de seguridad predeterminado ⓘ
VPC Permitir SSH Permitir ping
Reglas de la ACL predeterminada (Permitir todo)

Acceso clásico ⓘ
 Habilitar acceso a recursos clásicos Crear un prefijo predeterminado para cada zona

Crear subred en cada zona

Subred 1

Nombre	Zona	Grupo de recursos	Editar
sn-20240601-01	Madrid 1	Default	Editar
Prefijo de dirección	Número de direcciones	Rango de IP	
10.251.0.0/18	256	10.251.0.0/24	

Subred 2

Nombre	Zona	Grupo de recursos	Editar
sn-20240601-02	Madrid 2	Default	Editar
Prefijo de dirección	Número de direcciones	Rango de IP	
10.251.64.0/18	256	10.251.64.0/24	

Subred 3

Nombre	Zona	Grupo de recursos	Editar
sn-20240601-03	Madrid 3	Default	Editar
Prefijo de dirección	Número de direcciones	Rango de IP	
10.251.128.0/18	256	10.251.128.0/24	

Añadir subred +

Obtener llamada de API de ejemplo </>

Nube privada virtual

Choose an existing VPC or create a new one. [Learn more](#)

Nube privada virtual

cluster-ibm-cloud Madrid [Crear VPC](#)

Ubicación

Choose your worker zones and configure your subnets. [Learn more](#)

Zonas y subredes de trabajador

Madrid 1	<input checked="" type="checkbox"/>
Subred	sn-20240601-01
Editar	

Madrid 2	<input type="checkbox"/>

Madrid 3	<input type="checkbox"/>

[View all subnets](#)

Continuamos personalizando nuestro cluster y a continuación vemos como nos permite escoger la versión de Kubernetes que queramos desplegar. Por defecto IBM recomienda utilizar la 1.29, pero te ofrece la versión 1.27, 1.28 y 1.30. Nosotros hemos optado por desplegar la 1.28 ya que es la que hemos utilizado en el despliegue local y es con la que estamos más familiarizados.

Justo debajo encontramos la personalización de los nodos. IBM únicamente nos permite conocer las características de nuestros workers, desconocemos las capacidades del master, pero hemos de suponer que son potentes. Con respecto a los trabajadores, las opciones son muy amplias. Nosotros hemos optado por la configuración más barata para ahorrar costes y poder ejecutar dos nodos trabajadores, los cuales tienen cada uno dos núcleos en su CPU, 4 GB de memoria, 100GB de disco duro y un sistema operativo Ubuntu 20. Sin embargo, IBM nos permite lanzar nodos de 1TB de RAM, varios TB de almacenamiento, añadir almacenamiento secundario, un aumento de la velocidad de red y hasta 12 núcleos en el procesador.

Versión de Kubernetes

Seleccione la versión de plataforma Kubernetes para el clúster. Para obtener más información sobre las versiones, incluidos enlaces a las notas de release de la comunidad de plataforma de contenedor, [consulte la documentación](#).

1.28.10

Agrupación de trabajadores

Configure una agrupación de trabajadores con el tipo y el número de nodos de trabajador que deseé para ejecutar su primera carga de trabajo. En cualquier momento, puede añadir más agrupaciones de trabajadores con distintos tipos, o cambiar el tamaño de las agrupaciones de trabajadores según las necesidades de recursos de sus cargas de trabajo.

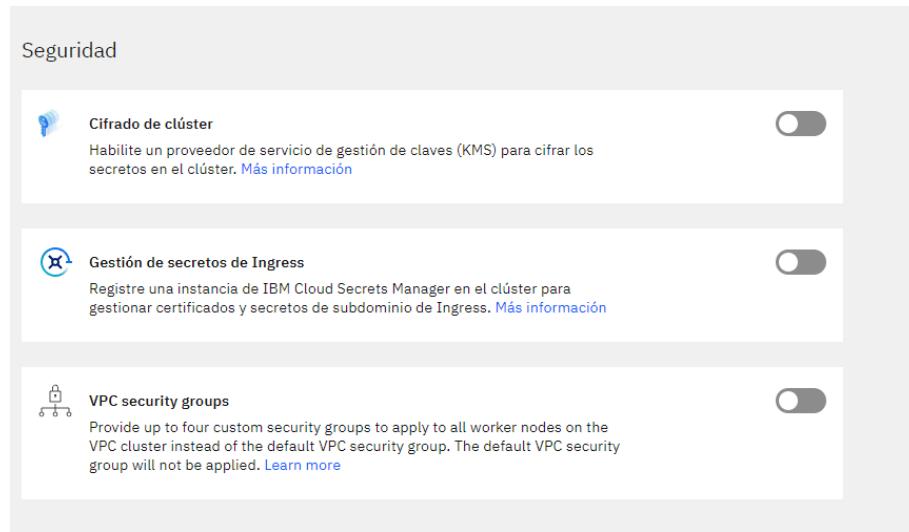
Nombre de la agrupación de trabajadores

predeterminado

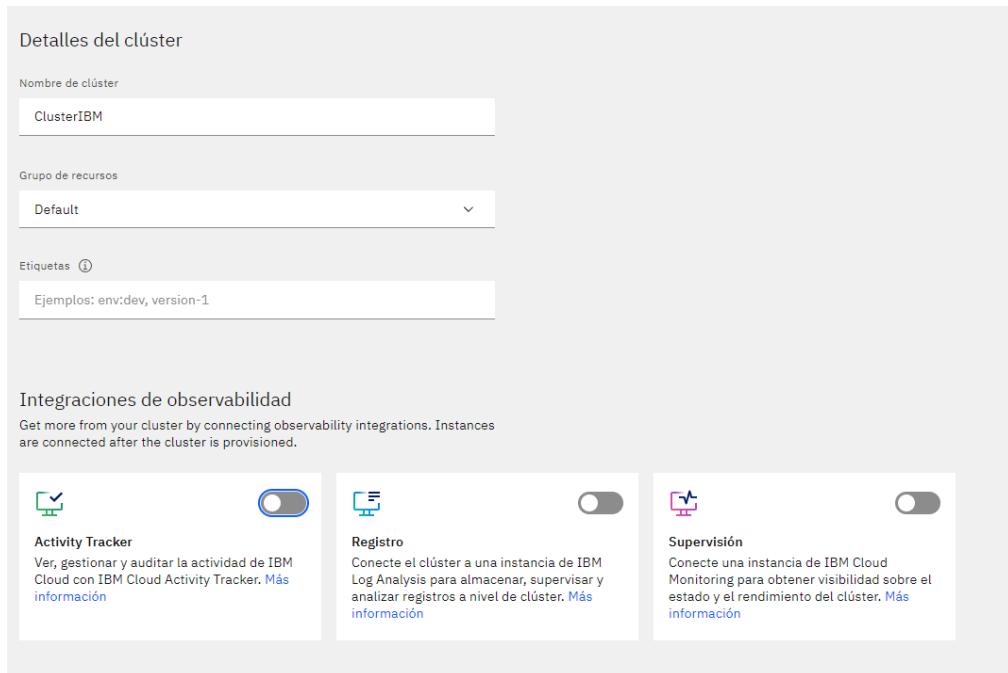
cx2.2x4 Virtual - compartido x86-64 Ubuntu 20			Nodos de trabajador por zona
2 vCPU	4GB Memoria	4Gbps Red	2
100GB Almacenamiento primario	Ninguno Almacenamiento secundario	0,12 €/hora Coste	x 1 zona = 2 worker nodes total

Cambiar tipo ↲

El siguiente apartado nos da ciertas opciones de seguridad, como cifrar nuestro clúster, gestionar certificados TLS o crear grupos de seguridad. Sin duda se trata de opciones muy interesantes pero que lamentablemente debemos descartar por su coste monetario.



Por último tenemos opciones para auditar y supervisar nuestro clúster con distintas herramientas que nos ofrece IBM Cloud, pero no las vamos a necesitar.



Una vez finalizada nuestra configuración, IBM Cloud nos calcula el precio a la hora y mensual de los servicios que hemos contratado. Vemos que nuestro total mensual será de 175€ al mes por desplegar este servicio.

The screenshot shows a dark-themed interface for configuring a Kubernetes cluster. At the top, there's a navigation bar with 'Resumen' and a dropdown set to 'España'. Below this, the configuration details for a 'Clúster de Kubernetes' are listed:

- Nodos de trabajador**: 2, price **0,23 €/hora**.
 - u3c.2x4 - 2 vCPU 4GB RAM
 - Virtual - compartido
 - x86-64
 - Ubuntu 20

At the bottom, the estimated monthly cost is displayed as **165,33 €/me**. A note below states: *Puede que se apliquen cargos adicionales para redes y ancho de banda. El total mensual real variará con los precios por niveles.*

Two buttons are present at the bottom: a large blue 'Crear' button and a smaller white 'Añadir a estimación' button.

6.2 Despliegue y pruebas con clúster de IBM Cloud

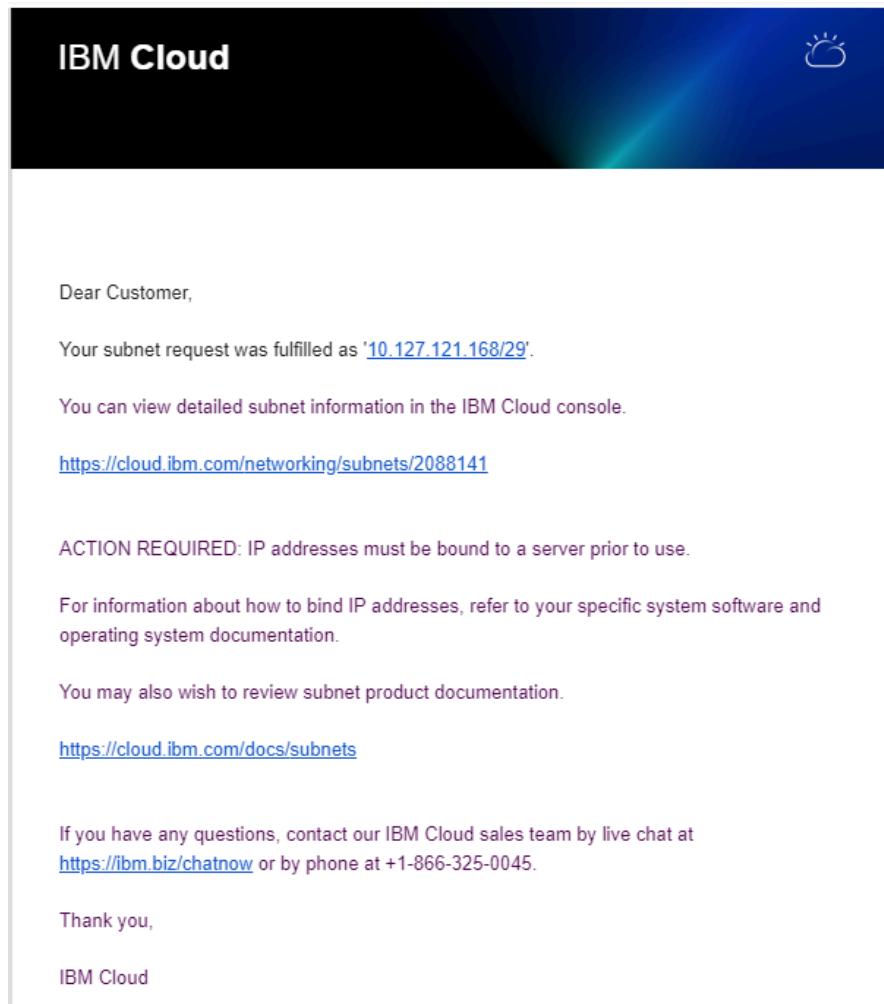
Una vez hemos completado la personalización de nuestro clúster nos aparecerá esta pantalla de monitoreo del mismo, donde vemos cómo se van desplegando las máquinas con los requisitos antes mencionados. Este proceso puede durar alrededor de 20 minutos, dependiendo de las distintas configuraciones.

The screenshot shows the 'Visión general' (General View) of a Kubernetes cluster named 'clusterIBMccloud'. The top navigation bar includes 'Clústeres / clusterIBMccloud', 'Preparando el maestro, trabajadores...', 'Añadir etiquetas', 'Ayuda', 'Panel de control de Kubernetes', and 'Acciones...'. The main area displays four status boxes: 'Worker node status' (2 de 2, Pendiente), 'Estado del complemento' (1 de 1, Pendiente), 'Estado del maestro' (Desconocido), and 'Estado de Ingress' (Desconocido). Below these are sections for 'Detalles' (Details) and 'Worker node health' (2 total worker nodes, 100% pending). A sidebar on the left lists 'Visión general', 'Nodos de trabajador', 'Agrupaciones de trabajadores', 'Redes', 'Ingress', and 'DevOps'.

Una vez terminado el proceso de instalación, vemos como nuestros nodos y nuestro maestro se encuentran en buen estado.

The screenshot shows the 'Visión general' (General View) of the same Kubernetes cluster after installation. The top navigation bar is identical. The main area now shows all components in a 'Buen estado' (Good) status: 'Worker node status' (2 de 2, Buen estado), 'Estado del complemento' (0 de 0, Buen estado), 'Estado del maestro' (Normal), and 'Estado de Ingress' (Buen estado). The 'Worker node health' section shows 100% normal nodes. The 'Redes' section includes a URL for the ingress service and a 'Gestionar' (Manage) button. The sidebar on the left remains the same.

También recibiremos una notificación al correo electrónico indicándonos el rango de nuestra subred que nos ha ofrecido IBM.



IBM Cloud nos permite controlar nuestro clúster a través del dashboard de Kubernetes, como ya hicimos anteriormente con el clúster local, o a través de su propio shell. Utilizamos el siguiente comando:

None

```
ibmcloud ks cluster config -c <id-del-clúster>
```

Esto nos permite descargar la configuración adecuada en nuestros nodos y utilizar el comando kubectl para ejecutar pods y despliegues en nuestro clúster.



```
eduardogayolopez@cloudshell:~$ ibmcloud ks cluster config -c cpdkuehm0942otmg8240
OK
The configuration for cpdkuehm0942otmg8240 was downloaded successfully.

Added context for cpdkuehm0942otmg8240 to the current kubeconfig file.
You can now execute 'kubectl' commands against your cluster. For example, run 'kubectl get nodes'.
eduardogayolopez@cloudshell:~$ 
```

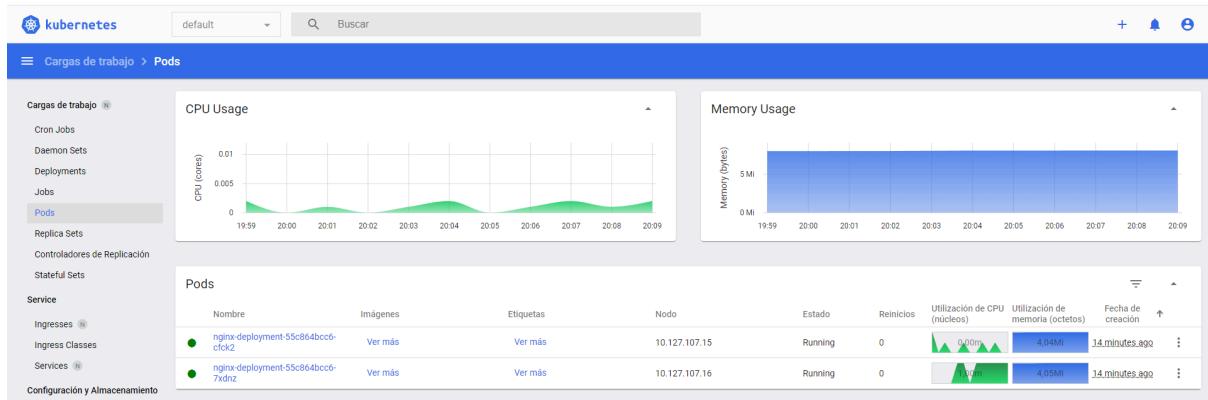


```
eduardogayolopez@cloudshell:~$ kubectl get nodes
NAME      STATUS    ROLES   AGE     VERSION
10.251.0.4 Ready     <none>  6m13s   v1.28.9+IKS
10.251.0.5 Ready     <none>  6m17s   v1.28.9+IKS
eduardogayolopez@cloudshell:~$ kubectl get deployments
No resources found in default namespace.
eduardogayolopez@cloudshell:~$ kubectl get pods
No resources found in default namespace.
eduardogayolopez@cloudshell:~$ 
```

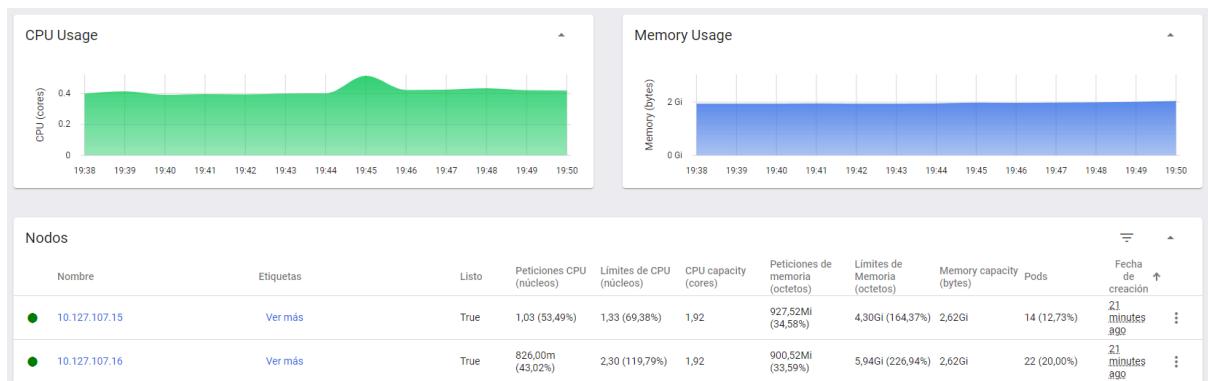
Vamos a utilizar primero el dashboard de Kubernetes para crear un despliegue sencillo de nginx mediante la opción de formulario. Esto simplifica al máximo el despliegue de aplicaciones en Kubernetes, ya que no es necesario escribir el archivo .yaml como habitualmente, sino que Kubernetes lo hace por nosotros.

A screenshot of the Kubernetes dashboard's "Create" page for a Deployment. The left sidebar lists various Kubernetes resources: Cargas de trabajo (Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets), Service (Ingresses, Services), and Configuración y Almacenamiento (Config Maps, Persistent Volume Claims, Secrets, Storage Classes). The main form is titled "Crear" and is set to "Crear desde formulario". It has three tabs: "Crear desde entrada", "Crear desde un fichero", and "Crear desde un formulario" (selected). The "Nombre de la aplicación" field contains "nginx-deployment". The "Imagen del contenedor" field contains "nginx". The "Número de pods" field is set to "2". Under the "Service" section, the "Service" dropdown is set to "External". The "Puerto" field is "80" and the "Puerto objetivo" field is "80", with "Protocolo" set to "TCP". The "Espacio de nombre" field is "default". At the bottom are buttons for "Desplegar" (Deploy), "Preview", "Cancelar", and "Show advanced options".

Como vemos, basta únicamente con darle un nombre a la aplicación (en este caso nginx-deployment), decirle la imagen que va a utilizar (si no especificamos la versión, utilizará la última versión que haya en el repositorio de docker), especificamos el número de pods, le decimos que es un servicio externo porque va a tener salida al exterior mediante IP pública y que va a escuchar en el puerto 80. Finalmente pinchamos en desplegar y ya tenemos nuestro despliegue funcionando.



Vemos también nuestros nodos desplegados.



Podemos ver los pods creados en el propio dashboard de Kubernetes.

```
eduardogayolopez@cloudshell:~$ kubectl get svc
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP  172.21.8.1   <none>        443/TCP   42m
my-report-service   NodePort   172.21.77.182  <none>        8081:30434/TCP   71s
nginx     LoadBalancer  172.21.100.3  159.8.65.99  80:31867/TCP   26m
eduardogayolopez@cloudshell:~$ 
```

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

Y también lo vemos en la consola de comandos. Si accedemos a la IP externa de nginx a través del puerto que nos dice el nodeport, vemos la página por defecto de nginx.

Ahora la idea es lanzar nuestra página personalizada para que tenga visualización en el exterior. Para ello vamos a crear un config map, un despliegue de nginx y un balanceador de carga para que las peticiones se repartan por los nodos.

Lo primero será escribir el .yaml de nuestro deployment de Nginx, especificando que el contenedor del puerto va a ser el 80, que va a tener 2 réplicas y que va a utilizar la ruta /usr/share/nginx/html/index.html, ya que vamos a crear un config map para que el contenedor utilice nuestro html personalizado.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels: app: nginx
  template:
    metadata:
      labels:
        ibmcloud: albayedu
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - containerPort: 80
          volumeMounts:
            - name: html-config
              subPath: index.html
              mountPath: /usr/share/nginx/html/index.html
      volumes:
        - name: html-config
          configMap:
            name: nginx-index-html
```

Después diseñamos nuestro config map donde le damos las especificaciones del html para que el navegador pueda interpretarlo.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-index-html
data:
  index.html: |
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Demostración TFG Kubernetes</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f4f4f4;
      color: #333;
      display: flex;
      flex-direction: column;
      min-height: 100vh;
    }
    header {
      background-color: #007acc;
      color: white;
      padding: 10px 0;
      text-align: center;
    }
    header h1 {
      margin: 0;
      font-size: 2.5em;
    }
    header p {
      margin: 5px 0 10px;
      font-size: 1.2em;
    }
    .container {
      flex: 1;
      padding: 20px;
    }
    .section-title {
      font-size: 2em;
      margin-top: 20px;
    }
    .section-content {
      margin-top: 10px;
      font-size: 1.1em;
      line-height: 1.6;
    }
  </style>
</head>
<body>
  <header>
    <h1>¡Bienvenidos!</h1>
    <p>Este sitio web es una demostración de la implementación de Kubernetes para la gestión de aplicaciones. Puedes navegar por las diferentes secciones para ver cómo se manejan los recursos y se distribuyen las peticiones entre los nodos.</p>
  </header>
  <div class="container">
    <h2>Sección 1: Introducción</h2>
    <p>En esta sección se presentan los fundamentos y conceptos básicos de Kubernetes, así como su arquitectura y funcionalidades principales.</p>
    <h2>Sección 2: Instalación y Configuración</h2>
    <p>Aprende sobre la instalación de Kubernetes en tu entorno local o en la nube, así como la configuración de los componentes clave para una implementación exitosa.</p>
    <h2>Sección 3: Gestión de Recursos</h2>
    <p>Descubre cómo gestionar los recursos disponibles en tu cluster, tanto en términos de capacidad como de calidad de servicio, para optimizar el rendimiento de tus aplicaciones.</p>
    <h2>Sección 4: Despliegues y Actualizaciones</h2>
    <p>Aprende sobre la creación y administración de despliegues en Kubernetes, así como las estrategias para realizar actualizaciones seguras y controladas de tus aplicaciones.</p>
    <h2>Sección 5: Balanceo de Carga y Accesos Seguros</h2>
    <p>Explora las técnicas para implementar balanceadores de carga y servicios de seguridad en tu cluster, garantizando la disponibilidad y la integridad de tus datos.</p>
  </div>
</body>

```

El siguiente paso es lanzar el Load Balancer, para que el despliegue de Nginx pueda redirigir las peticiones a un pod u otro dependiendo de las necesidades que tenga en ese momento.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 80
```

Por último, lanzamos el servicio “nodeport” lo que va a abrir un puerto y mantenerlo a la espera de peticiones. La especificación de “port” va a ordenar el puerto que se encuentra abierto dentro del clúster. El “targetport” es el puerto en que la aplicación está corriendo dentro de los pods. El tráfico que llegue al puerto 8081 será redirigido al puerto 80. El nodeport será el puerto de cada nodo del clúster que estará disponible, y es lo que permite acceder al servicio desde fuera del clúster. Lo dejamos comentado para que sea Kubernetes el que decida el puerto a abrir, pero podemos especificarle uno en concreto si lo necesitamos.

```
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport-service
  labels:
    app: nginx
spec:
  selector: app: nginx
  type: NodePort
  ports:
    - port: 8081
      targetPort: 80
      # nodePort: 30001
```

Con esta configuración, ya tenemos nuestra página modificada disponible desde fuera del clúster a través de la IP pública 159.8.64.194:31012

IBM Cloud Shell

Sesión 1 +

```
eduardogayolopez@cloudshell:~$ kubectl get service
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes     ClusterIP  172.21.0.1  <none>        443/TCP   31m
nginx-deployment-service  LoadBalancer  172.21.10.10  <none>        8081:31012/TCP  6m23s
nginx-deployment  LoadBalancer  172.21.76.26  159.8.64.194  80:31555/TCP  20m

eduardogayolopez@cloudshell:~$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
nginx-deployment-5cdf5d5bc-5808t  1/1    Running   0          8m20s
nginx-deployment-5cdf5d5bc-xsn2h  1/1    Running   0          8m20s

eduardogayolopez@cloudshell:~$ [ ]
```

Demostración TFG Kubernetes

Creado por Eduardo Gayo López y Alba Morejón García

Demostración del Proyecto

El siguiente ejemplo muestra cómo Kubernetes gestiona automáticamente el despliegue, escalado y operaciones de las aplicaciones en contenedores. Este despliegue demuestra la funcionalidad y eficiencia del orquestador.

Despliegue en Kubernetes

Aquí puedes ver una captura de pantalla del despliegue de una aplicación en Kubernetes. La aplicación se despliega en varios nodos y se gestiona automáticamente para garantizar alta disponibilidad y rendimiento.

Podemos observar los logs de los pods en tiempo real para ver cómo llegan las peticiones GET con el siguiente comando:

None

```
kubectl logs -f <nombre_pod>
```

Recordamos que nosotros creamos un deployment con dos réplicas/pods. Probamos a eliminar uno de los pods y vemos que se genera otro automáticamente, siguiendo los requisitos configurados anteriormente.

```
eduardogayolopez@cloudshell:~$ kubectl get pods -o wide
NAME           READY   STATUS    RESTARTS   AGE     IP          NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-5cdfbd5bc5-8d8mt  1/1    Running   0          20m    172.30.81.18  10.127.107.16  <none>        <none>
nginx-deployment-5cdfbd5bc5-8lrlz1  1/1    Running   0          24s    172.30.108.74  10.127.107.15  <none>        <none>
eduardogayolopez@cloudshell:~$ kubectl delete pod nginx-deployment-5cdfbd5bc5-8lrlz1
pod "nginx-deployment-5cdfbd5bc5-8lrlz1" deleted

^ceduardogayolopez@cloudshell:~$ kubectl get pods -o widezl
NAME           READY   STATUS    RESTARTS   AGE     IP          NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-5cdfbd5bc5-8d8mt  1/1    Running   0          20m    172.30.81.18  10.127.107.16  <none>        <none>
nginx-deployment-5cdfbd5bc5-8lrlz1  1/1    Terminating   0          67s    172.30.108.74  10.127.107.15  <none>        <none>
nginx-deployment-5cdfbd5bc5-rjdm7  1/1    Running   0          26s    172.30.108.75  10.127.107.15  <none>        <none>
eduardogayolopez@cloudshell:~$
```

7. Conclusiones

Kubernetes es un software muy completo que nos permite gestionar nuestros contenedores a través de pods y tener control de nuestros nodos y los eventos que ocurren en ellos de una forma bastante sencilla. Es una herramienta ideal para despliegues de software que requieren alta disponibilidad, ya que los pods se van replicando de forma automática cuando alguno falla, lo que supone una tolerancia a errores muy importante. También es perfecta para aplicaciones que pueden tener picos puntuales de actividad, ya que incluye una función de autoescalado horizontal que activa nodos según la carga de trabajo que tenga en el momento.

Sin embargo, no es una herramienta sencilla de utilizar. El simple hecho de realizar un despliegue sencillo ya requiere conocimientos medios sobre las redes y los sistemas operativos, por lo que no está pensada para el usuario común. Además, Kubernetes ofrece tal cantidad de opciones, personalización y recursos que requiere una gran cantidad de tiempo y esfuerzo para llegar a dominarlo.

En lo referente a los servicios en la nube, son un recurso excepcional para la computación a día de hoy, poniendo al alcance del público medio recursos bajo demanda que permiten iniciar negocios online sin tener que preocuparse del hardware que conlleva detrás.

Es un mercado en alza que nos ofrece varios proveedores como AWS, IBM Cloud, Google Cloud, Microsoft Azure, etc. lo que supone competitividad entre las mismas y precios más asequibles para el usuario final, así como mejora constante de los servicios ofrecidos. Además, este tipo de recursos cuentan con toda clase de guías, vídeos y ayudas para que cualquiera pueda empezar a utilizar sus servicios, no sin ello permitir una personalización en las configuraciones para las que ya son necesarios conocimientos informáticos avanzados, por lo que su público es muy amplio.

En definitiva, tanto Kubernetes como los servicios en la nube son herramientas muy potentes y utilizadas por las empresas en el día de hoy. No cabe duda que serán una opción muy escogida por parte de los usuarios tanto en la actualidad como en el futuro. Estas herramientas ofrecen un abanico de posibilidades amplísimo y, a pesar del estudio en profundidad que hemos realizado sobre ellas, sólo hemos arañado la superficie de todo lo que pueden llegar a ofrecer.

8. Webgrafía

Rojas, S. (2023). *Instalando Kubernetes en Ubuntu 22.04*. Medium. Recuperado de <https://medium.com/@sirpyerre/instalando-kubernetes-en-ubuntu-22-04-8e29c39c7671>

IBM. (n.d.). *¿Qué es la contenerización?*. IBM. Recuperado de <https://www.ibm.com/es-es/topics/containerization>

Buchanan, I. (n.d.). *Contenedores vs. máquinas virtuales*. Atlassian. Recuperado de <https://www.atlassian.com/es/microservices/cloud-computing/containers-vs-vms>

Kubernetes. (2024). *¿Qué es Kubernetes?* Recuperado de <https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/>

Gaspar, R. (2021). *Despliegue de un cluster Kubernetes altamente disponible en Google Cloud Platform* (Trabajo de fin de grado, Universitat Politècnica de València). Recuperado de <https://riunet.upv.es/bitstream/handle/10251/188567/Gaspar%20-%20Despliegue%20de%20un%20cluster%20Kubernetes%20altamente%20disponible%20en%20Google%20Cloud%20Platform.pdf?sequence=1>

HowtoForge. (2022, mayo 31). *Cómo instalar y configurar Kubernetes y Docker en Ubuntu 18.04 LTS*. Recuperado de <https://howtoforge.es/como-instalar-y-configurar-kubernetes-y-docker-en-ubuntu-18-04-lts/>

López, L. (2020, diciembre 21). *Kubernetes: La guía de instalación definitiva*. Blog de Pleiades Tecnología. Recuperado de <https://blog.plds.es/?p=1152>

Coronado, A. (2019, agosto 1). *Despliegue de aplicaciones sobre Kubernetes*. Recuperado de <https://www.albertcoronado.com/2019/08/01/despliegue-de-aplicaciones-sobre-kubernetes/>

IBM. (2024). *Calico*. Recuperado de <https://www.ibm.com/docs/es/cloud-private/3.2.x?topic=ins-calico>

Tigera, Inc. (2024). *About Calico*. Recuperado de <https://docs.tigera.io/calico/latest/about/>

Jovani, A. (2023). *Despliegue de un cluster Kubernetes altamente disponible en IBM Cloud* (Trabajo de fin de grado, Universitat Politècnica de València). Recuperado de <https://riunet.upv.es/bitstream/handle/10251/198107/Jovani%20-%20Despliegue%20de%20un%20cluster%20Kubernetes%20altamente%20disponible%20en%20IBM%20Cloud.pdf?sequence=1&isAllowed=y>

Red Hat. (s.f.). ¿Qué son los servicios en la nube? Red Hat. Recuperado de <https://www.redhat.com/es/topics/cloud-computing/what-are-cloud-services>

IMARC Group. (2023). *Mercado de orquestación de contenedores: Tendencias globales de la industria, participación, tamaño, crecimiento, oportunidad y pronóstico 2024-2029*. IMARC Group. Recuperado de <https://www.imarcgroup.com/container-orchestration-market>

Stackscale. (2023, 1 de febrero). *Modelos de servicio cloud: IaaS, PaaS y SaaS*. Stackscale. Recuperado de <https://www.stackscale.com/es/blog/modelos-de-servicio-cloud/>

IBM. (2023). *IBM Cloud Kubernetes Service*. IBM. Recuperado de <https://www.ibm.com/es-es/products/kubernetes-service>

Portaltic/EP. (2019, 4 de noviembre). Así adapta IBM la nube a las necesidades de la empresa: más allá de Kubernetes. El Español. Recuperado de https://www.elespanol.com/invertia/empresas/tecnologia/20191104/adapta-ibm-nube-necesidades-empresa-all-a-kubernetes/441957007_0.html

Asum Cloud. (n.d.). *¿Cuál es el mejor orquestador de contenedores? Descúbrelo ahora.* Recuperado de <https://ausum.cloud/kubernetes-swarm-mesos-mejor-orquestador-contenedores/#swarm>