ELFHEIN

KubeMagic

Un producto original, adaptado a tus necesidades y sin límites.

Autor: Javier Ramírez Moral

Fecha de Publicación: 12/06/2023

Documentación Técnica & Manual de Uso Primera Parte

Contenido

Resumen de este documento.

En este documento se pretende mostrar todo el apartado técnico con sus correspondientes explicaciones para complementarlo y poder usarlo posteriormente a modo de manual de uso referido únicamente a la Primera parte del proyecto.

Primera Parte: Introducción.

¿Qué es Kubernetes?

Kubernetes es una plataforma de código abierto que se utiliza para administración y orquestación de aplicaciones de aplicaciones que están almacenados en contenedores como por ejemplo Docker. Todo ello de manera eficiente y puede estar en distintos entornos ya sea en físico, virtual o en la nube. Además de todo esto facilita el escalado de las aplicaciones en entornos de producción en función de la demanda y garantizando su disponibilidad. También proporciona herramientas para poder monitorearlas y para la automatización de tareas de mantenimiento y actualización de las aplicaciones.

Fue desarrollada por Google y actualmente es mantenida por la Cloud Native Computing Foundation (CNCF).

1.Logo de Kubernetes

Características de Kubernetes

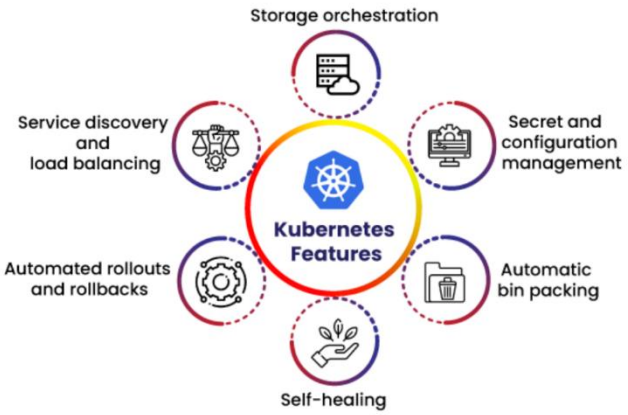
Kubernetes tiene varias características que lo hacen una plataforma popular para la orquestación de contenedores. Algunas de las características más importantes de Kubernetes son:

- Escalabilidad: Kubernetes es altamente escalable y puede manejar cargas de trabajo de gran tamaño. Puede agregar o eliminar nodos en un clúster en tiempo real para adaptarse a la demanda de recursos.
- Portabilidad: Kubernetes es compatible con una amplia variedad de plataformas de infraestructura, incluyendo nubes públicas, privadas y locales. Esto permite a los usuarios mover aplicaciones de un entorno a otro con facilidad.
- Automatización: Kubernetes automatiza muchas tareas relacionadas con la implementación y la gestión de aplicaciones, incluyendo el escalado automático, la gestión de recursos y la distribución de tráfico.
- Resiliencia: Kubernetes es altamente resistente a fallos. Puede detectar y recuperarse automáticamente de fallos de hardware o software en los nodos o contenedores.

- Configuración declarativa: Kubernetes utiliza archivos de configuración YAML para definir el estado deseado de una aplicación. Esto facilita la configuración y el mantenimiento de aplicaciones complejas.
- Despliegue sin interrupciones: Kubernetes admite el despliegue sin interrupciones de nuevas versiones de aplicaciones, lo que garantiza una experiencia de usuario fluida.
- Monitoreo y registro: Kubernetes proporciona herramientas integradas para el monitoreo y el registro de aplicaciones, lo que permite a los usuarios supervisar el estado y el rendimiento de las aplicaciones en tiempo real.

Hoy en día las aplicaciones se implementan mediante contenedores basados en la virtualización del sistema operativo en vez de en el hardware. Estos contenedores están aislados del resto y del servidor en donde los mismos se encuentran, pudiendo limitar los recursos que estos usan.

Todo esto hace que los contenedores sean fácilmente transportables entre distintas nubes y sistemas operativos gracias a que son fáciles de crear en comparación con las máquinas virtuales y además no están conectados a la infraestructura del anfitrión.



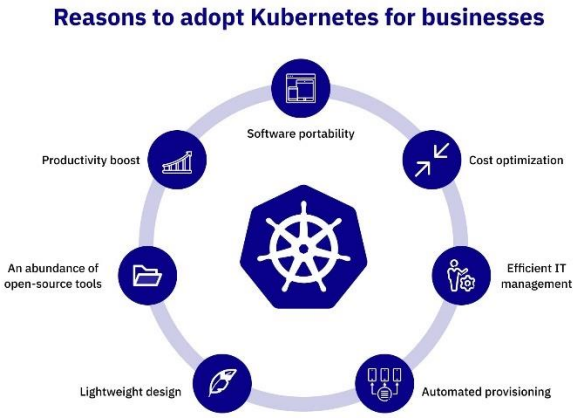
2.Características de Kubernetes

Utilidades de Kubernetes

Podemos usar Kubernetes en casos como:

- Despliegue de aplicaciones: Kubernetes permite desplegar aplicaciones en contenedores de manera automatizada y gestionar su escalabilidad y disponibilidad. Ya sean de análisis de datos, de aplicaciones móviles o sitios web como va a ser nuestro caso.
- Microservicios: Kubernetes es una plataforma popular para implementar arquitecturas de microservicios, donde cada servicio se ejecuta en un contenedor y se orquesta mediante Kubernetes.
- Entornos de prueba y desarrollo: Kubernetes es una plataforma ideal para crear entornos de prueba y desarrollo de manera rápida y eficiente, lo que ayuda a reducir el tiempo de comercialización de las aplicaciones.
- Abstracción de la infraestructura: Al usar Kubernetes, este se va a encargar de la computación, las redes y los espacios de almacenamiento de las cargas de trabajo.

De tal forma que nosotros como desarrollares, podemos centrarnos en la construcción de las aplicaciones y desentendernos de las capas subyacentes.



3.Utilidades de Kubernetes

¿Por qué entonces he decidido usar Kubernetes para este proyecto?

Una vez ya hemos visto qué es Kubernetes y cuáles son sus características se habrá podido intuir gran parte de mis razones para usar esta plataforma.

En este proyecto el cliente nos pedía básicamente una puesta en marcha de sus dos páginas webs rápida y que además luego fuera fácil de complementar con ajustes y herramientas adicionales para asegurar su correcto y constante funcionamiento.

Teniendo en cuenta que era fundamental tener un entorno escalable, adaptativo y dinámico para poder soportar el tráfico de ambas aplicaciones, opté por desplegar ambos escenarios usando Kubernetes. Son múltiples las ventajas que nos ofrece, un clúster robusto, monitoreo del estado de nuestras aplicaciones, actualizaciones sin tener que detener nada, en caso de que un contenedor falle este es reemplazado por otro automáticamente para que las aplicaciones sigan funcionando sin interrupciones, lo que hace que la experiencia del usuario sea fluida que al final es nuestro objetivo. Además de que como ambas páginas van a ser muy visitadas Kubernetes nos va a permitir aumentar o disminuir el número de recursos de las aplicaciones en función de la demanda.

Con Kubernetes también nos va a permitir maximizar la disponibilidad de tus aplicaciones, optimiza los recursos. Así como implementar todo en la nube, lo cual nos abre un gran abanico de opciones de configuración y administración además de contar con herramientas de monitoreo más potente y que pueden convivir entre ellas para hacer mucho más potente la parte de mantenimiento y seguridad de nuestro proyecto.

Componentes de Kubernetes

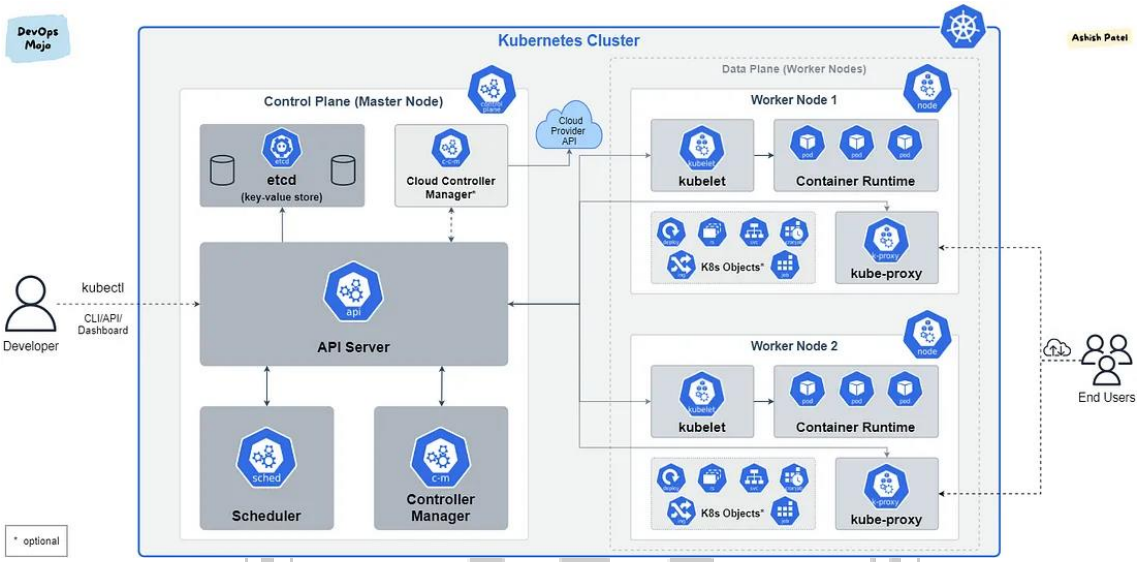
Ahora vamos a pasar a explicar los componentes que entran en juego en Kubernetes y como se relacionan entre sí. Vamos a entender cómo funciona Kubernetes.

Clúster:

Un clúster de Kubernetes es un conjunto de uno o más nodos que ejecutan el software de Kubernetes y trabajan juntos como una sola entidad para ejecutar y administrar

aplicaciones en contenedores. Cada nodo en el clúster es una máquina virtual o física que ejecuta un sistema operativo y el software de Kubernetes, y está conectado a una red que permite la comunicación entre los nodos.

Cuando se ejecutan aplicaciones en contenedores en un clúster de Kubernetes, los contenedores se ejecutan en pods. Los pods son la unidad básica de implementación en Kubernetes y pueden contener uno o más contenedores. Los nodos de trabajo ejecutan los pods, y el control plane de Kubernetes es responsable de orquestar la ejecución de los pods en los nodos de trabajo.



4.Clúster de Kubernetes

El clúster de Kubernetes utiliza un conjunto de componentes principales para coordinar y administrar la ejecución de aplicaciones en contenedores. Podemos diferenciar dos partes por las que se compone un clúster y sus diferentes partes en cada uno:

Control Plane: Es el componente central de un clúster de Kubernetes y se encarga de orquestar y administrar las operaciones del clúster. Todas las decisiones se hacen en él. El control plane incluye los siguientes componentes:

- kube-apiserver: es el servidor API que expone la API de Kubernetes a los usuarios y servicios. Cuando nosotros vayamos a interactuar con nuestro clúster por medio de la línea de comandos (con la herramienta kubectl que veremos más adelante), en realidad nos estamos comunicando con el servidor API. Este es el principal punto de administración de todo el clúster. Un dato interesante para saber es que este es el único componente dentro del clúster que se va a comunicar con el etcd (donde se almacena el estado de todo nuestro clúster), en donde se harán las modificaciones pertinentes según nosotros lo declaremos desde la línea de comandos. Además, también es el responsable del mecanismo de autentificación y autorización, todos los clientes de la API deben estar autenticados para poder interactuar con el servidor de la API. Todos los componentes del clúster se comunican con él. Coordina todos los procesos entre el plano de control y los Worker nodes.

- Cloud controller mánager: es un componente que actúa como intermediario entre el clúster de Kubernetes y el proveedor de la nube subyacente. Su función principal, es integrar y exponer las características específicas de la nube en la que se ejecuta el clúster de Kubernetes. Nos permite que el clúster de Kubernetes aprovisione recursos en la nube.

- kube-controller-mánager: es un conjunto de controladores que se encargan de realizar tareas como el control de replicas, la gestión de recursos y el manejo de errores. Básicamente observa el estado de nuestro clúster a través de la API y cuando se recibe la notificación de algún cambio se encarga de realizarlos siempre buscando que el estado actual de nuestras aplicaciones sea la misma que nosotros le especificamos (estado actual= estado deseado), por ejemplo, el controlador de replicación que vamos a ver más adelante.

- etcd: Es una base de datos de clave valor distribuida que almacena información sobre el estado del clúster de Kubernetes (qué nodos existen en el clúster, qué pods deberían estar ejecutándose, en qué nodos se están ejecutando y mucho más) y su configuración en todo momento de todos los objetos. Es como el cerebro de Kubernetes. etcd almacena todos los objetos bajo la clave del directorio /registry en formato clave-valor. Por ejemplo, la información sobre un pod llamado wordpress en el espacio de nombres predeterminado se puede encontrar en /registry/pods/default/WordPress

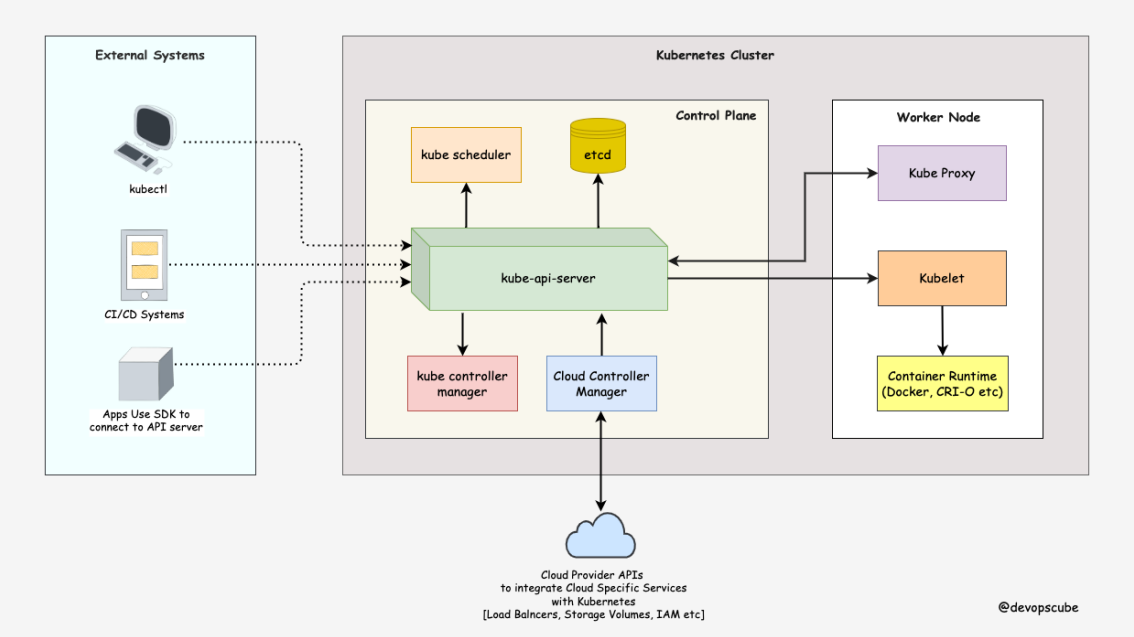
- Kube-scheduler: Es el componente que asigna pods a los nodos trabajadores en función de los recursos disponibles y las políticas de asignación. Cuando nosotros vamos a implementar un pod, definimos una serie de parámetros como lo pueden ser los recursos, volúmenes etc. La tarea principal es, por tanto, identificar la solicitud de creación y elegir el mejor nodo que cumpla con los requisitos para el pod.

Worker Nodes: Son las máquinas en las que se ejecutan los contenedores que forman las aplicaciones. Los nodos de trabajo incluyen los siguientes componentes:

- Kubelet: Es el agente que se ejecuta en cada nodo de trabajo y se comunica con el control plane para recibir instrucciones sobre cómo ejecutar los contenedores. Una vez que el pod tiene un nodo asignado, se activa el comportamiento normal de Kubelet y se crean el pod y sus contenedores. Básicamente se encarga de que todos los contenedores estén funcionando y en buen estado. Es como un agente supervisor del nodo.

- Kube-proxy: Es el componente responsable de enrutar el tráfico de red a los contenedores en los nodos de trabajo. Permite la comunicación con la red interna del clúster mediante reglas de red. Este se ejecuta en cada nodo. kube-proxy mantiene reglas de red en los nodos. Estas reglas de red permiten la comunicación de red con sus Pods desde sesiones de red dentro o fuera de su clúster. El proxy de Kube se comunica con el servidor API para obtener los detalles sobre el Servicio (ClusterIP) y las IP y los puertos respectivos de los pods (puntos finales).

- Container runtime: es el software que se encarga de ejecutar los contenedores en el nodo de trabajo. En nuestro caso será Docker.



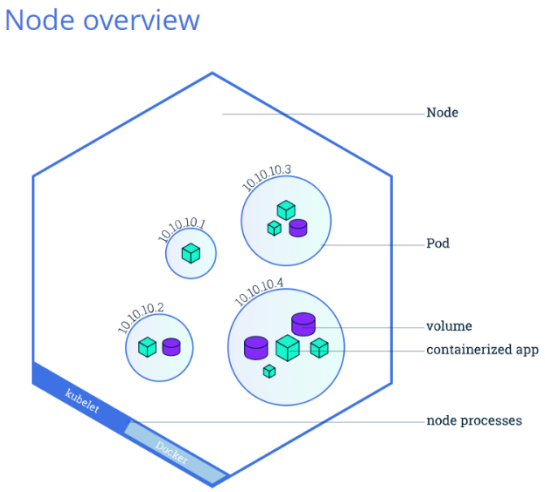
5.Vista general de la arquitectura de Kubernetes

Nodos:

Un nodo en Kubernetes es una máquina virtual o física que forma parte del clúster y se encarga de ejecutar los contenedores de las aplicaciones. Cada nodo tiene el agente llamado kubelet del que hemos hablado antes que se encarga de comunicarse con el plano de control de Kubernetes para recibir las órdenes de ejecución de los pods, y también de supervisar el estado de los contenedores para garantizar su correcta ejecución.

Cada nodo en Kubernetes puede tener uno o varios contenedores en ejecución, y se pueden añadir o quitar nodos de los clústeres según sea necesario. Los nodos también pueden tener diferentes recursos de hardware, CPU, memoria o almacenamiento, lo que permite escalar vertical o horizontalmente el clúster según sea necesario.

Los nodos pueden tener diferentes roles dentro del clúster, por ejemplo, un nodo puede ser designado como nodo maestro y tener los componentes del plano de control de Kubernetes en ejecución. Mientras que otros nodos pueden ser designados como nodos de trabajo y tener solo los contenedores de las aplicaciones. También se pueden añadir etiquetas a los nodos para facilitar la gestión y el escalado del clúster.



6.Nodo de Kubernetes

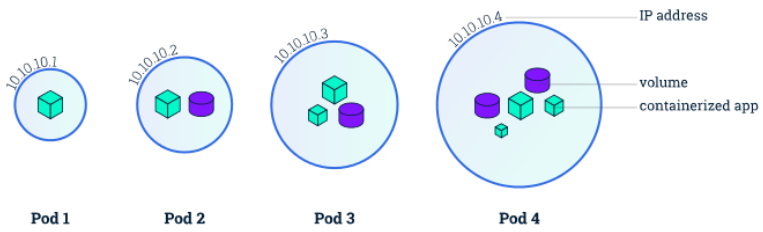
Pod:

Un pod la unidad básica de despliegue y ejecución de los contenedores mínima. Un pod consiste en uno o varios contenedores que comparten el mismo espacio de red y de almacenamiento de disco, y se ejecutan en el mismo nodo de Kubernetes.

Los pods son efímeros y pueden ser creados, eliminados y reemplazados de manera dinámica y automática en respuesta a diferentes eventos, como cambios en la demanda de la aplicación o fallos en el nodo o en el contenedor. Además, los pods son altamente escalables, lo que significa que se pueden crear múltiples instancias de un pod para manejar una carga de trabajo más grande.

Cada pod tiene su propia dirección IP dentro del clúster de Kubernetes, lo que permite que los contenedores dentro del pod se comisquen entre si a través de la red. Además, los pods pueden acceder a los recursos compartidos, como los volúmenes de almacenamiento.

Los pods son administrados por el controlador de replicación, que es responsable de garantizar que el número deseado de pods esté en ejecución en todo momento. También es posible definir diferentes políticas de reinicio, estrategias de actualización y otros aspectos de la configuración del pod a través de los objetos de Kubernetes.



7.Pod de Kubernetes

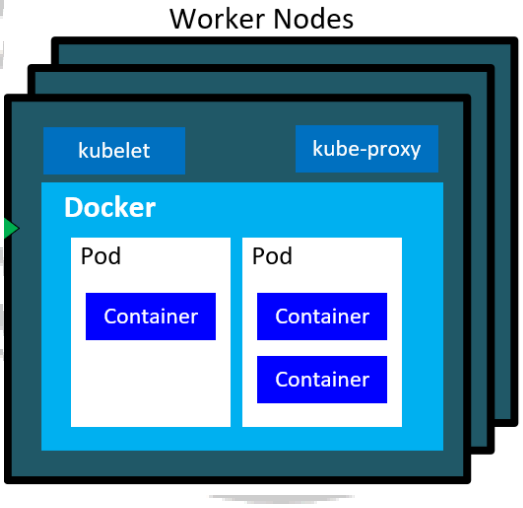
Contenedor:

Un contenedor es una unidad de software que se ejecuta dentro de un pod y que se encapsula una aplicación y todas sus dependencias y configuraciones en un entorno aislado y portátil.

Los contenedores son una forma de virtualización a nivel de sistema operativo que permite que múltiples aplicaciones se ejecuten en el mismo sistema operativo, pero de forma aislada y segura. Cada contenedor tiene su propio sistema de archivos, bibliotecas y configuraciones, lo que garantiza que las aplicaciones se ejecuten de manera coherente en diferentes entornos.

Kubernetes utiliza los contenedores como unidad básica de despliegue y de ejecución de las aplicaciones. Los contendores se pueden crear a partir de imágenes, que son archivos que contienen todo lo necesario para ejecutar aplicaciones, incluyendo el código, las bibliotecas, dependencias y configuraciones.

Además, Kubernetes proporciona una plataforma para administrar los contenedores a través de diferentes objetos, como los pods, los controladores de replicación y los servicios. Los objetos de Kubernetes permiten gestionar la escalabilidad, el equilibrio de carga, la resiliencia y otras características importantes de los contenedores y las aplicaciones que se ejecutan en ellos.



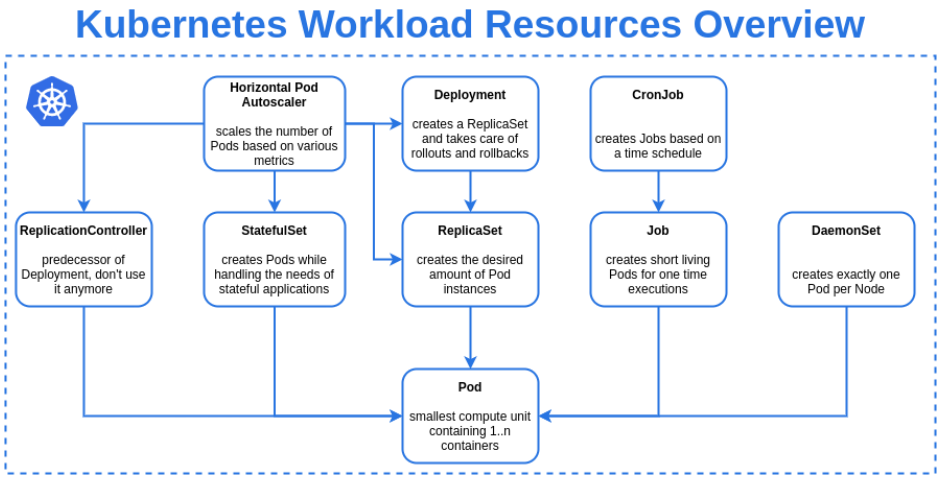
8.Arquitectura de Kubernetes

Workload:

Se refiere a cualquier tipo de tarea que se ejecuta en un clúster de Kubernetes. Puede ser una aplicación, un servicio, un microservicio, una tarea programada o cualquier tipo de trabajo que se ejecuta dentro de un clúster.

Kubernetes proporciona diferentes tipos de Workload, cada uno diseñado para satisfacer diferentes necesidades y requisitos de las aplicaciones. Los iremos viendo más adelante y en más detalles, pero por ejemplo tenemos los Deployment.

Cada tipo de Workload en Kubernetes se gestiona a través de diferentes objetos, que proporcionan diferentes funcionalidades y características para el despliegue y gestión de las aplicaciones. En general, los Workload en Kubernetes permiten desplegar y gestionar aplicaciones de forma fácil, eficiente y escalable.



9.Flujo de trabajo

Herramientas de Kubernetes

Ahora vamos a ver las tres herramientas de Kubernetes que vamos a usar para llevar a cabo la siguiente parte del proyecto.

Minikube es una herramienta que te permite crear y ejecutar clústeres de Kubernetes de un solo nodo en tu propia computadora. Minikube utiliza una máquina virtual (VM) para crear un entorno aislado en el que se ejecuta Kubernetes. Después de configurar el hipervisor, se crea un clúster de Kubernetes local. Este comando descarga y configura una imagen de máquina virtual de Kubernetes y la arranca en el hipervisor.

10.Logo Minikube

Kubectl es una herramienta de línea de comandos que te permite interactuar con un clúster de Kubernetes. Puedes utilizar kubectl para crear, modificar y eliminar objetos de Kubernetes, como pods, servicios y despliegues, y para obtener información sobre el estado del clúster.

Kubectl funciona mediante el envío de solicitudes HTTP a la API del servidor de Kubernetes. Estas solicitudes HTTP se envían en formato JSON o YAML y contienen información sobre los objetos de Kubernetes que deseas crear, modificar o eliminar, así como cualquier otra información relevante.

Cuando ejecutas un comando kubectl, la herramienta genera una solicitud HTTP correspondiente y la envía al servidor de Kubernetes. El servidor procesa la solicitud y realiza las acciones necesarias en el clúster de Kubernetes. Una vez que el servidor ha realizado la acción solicitada, devuelve una respuesta a kubectl que contiene información sobre el estado del clúster. Para ver todos los comandos disponibles que tenemos con esta herramienta, le recomiendo consultar esta página:

https://kubernetes.io/docs/reference/kubectl/cheatsheet/

11.Kubectl

Kubeconfig: Es un archivo de configuración que se utiliza para especificar cómo se deben autenticar y autorizar los usuarios, y cómo se deben acceder a los diferentes clústeres de Kubernetes.

El kubeconfig es utilizado por la herramienta de línea de comandos de Kubernetes, kubectl, para autenticar y autorizar los usuarios que realizan operaciones en los clústeres. El kubeconfig contiene información sobre el clúster, como la dirección del servidor API, el certificado del servidor y la clave privada, así como información sobre cómo autenticarse, como el nombre de usuario y la contraseña, o las credenciales de un token de acceso.

12.Kubeconfig

Docker:

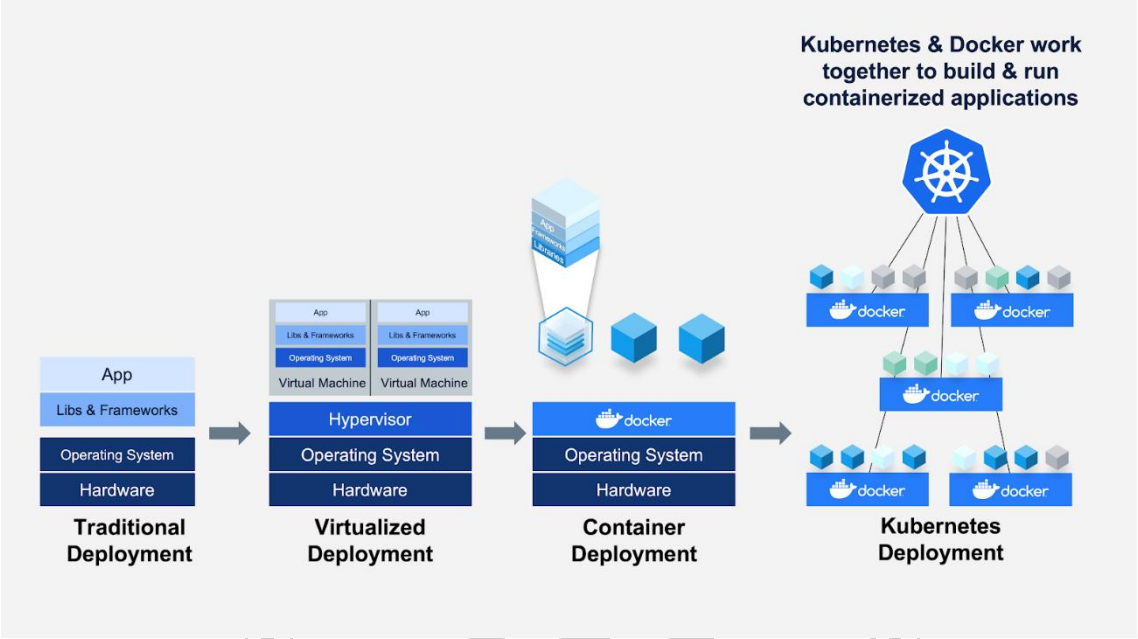
Me gustaría aclarar cómo podemos combinar ambas tecnologías. Por una parte, Docker es una tecnología que nos permite crear e implementar aplicaciones de manera rápida en tiempo de ejecuciones. Docker lo que hace es empaquetar este software de las aplicaciones en contenedores donde se incluye todo lo necesario como bibliotecas. Y Kubernetes va a administrar y orquestar estos contenedores con las aplicaciones dentro de ellos ejecutándose.

Ambas son tecnologías contenedoras y de código abierto. Su principal diferencia es en el papel que juega cada una a la hora de distribuir las aplicaciones en contenedores. Las desarrolladoras usan Docker para crear y administrar imágenes de contenedores. Y Kubernetes es usado administrar a gran escala y coordinar estos contenedores.

Básicamente Docker se usa para crear, compartir y ejecutar aplicaciones en contenedores por medio de un conjunto de herramientas como puedes ser Docker build para crear las imágenes, Docker compose para ejecutar las aplicaciones o Docker Hub para encontrar y compartir estas imágenes como puede ser GitHub a modo de repositorio.

Por otro lado, Kubernetes funciona administrando un clúster de instancias de computación. Lo que hace es programar estos contenedores que se ejecutan en nuestro clúster dependiendo de los recursos de computación que tengamos disponibles.

Docker nos agiliza el ciclo de vida del desarrollo y cargas de trabajo altamente portátiles y Kubernetes nos permite definir las aplicaciones complejas en los contenedores y ejecutar a una gran escala.



13.Comparativa del despliegue de aplicaciones