Introducción a Kubernetes

Antonio Espin Herranz

Introducción

- Definición: Kubernetes (**K8s**) es una plataforma de **código abierto** para la **orquestación** de **contenedores** que automatiza la implementación, escalado y gestión de aplicaciones en contenedores.

- Origen: Fue desarrollado por Google y donado a la Cloud Native Computing Foundation (**CNCF**) en 2015.

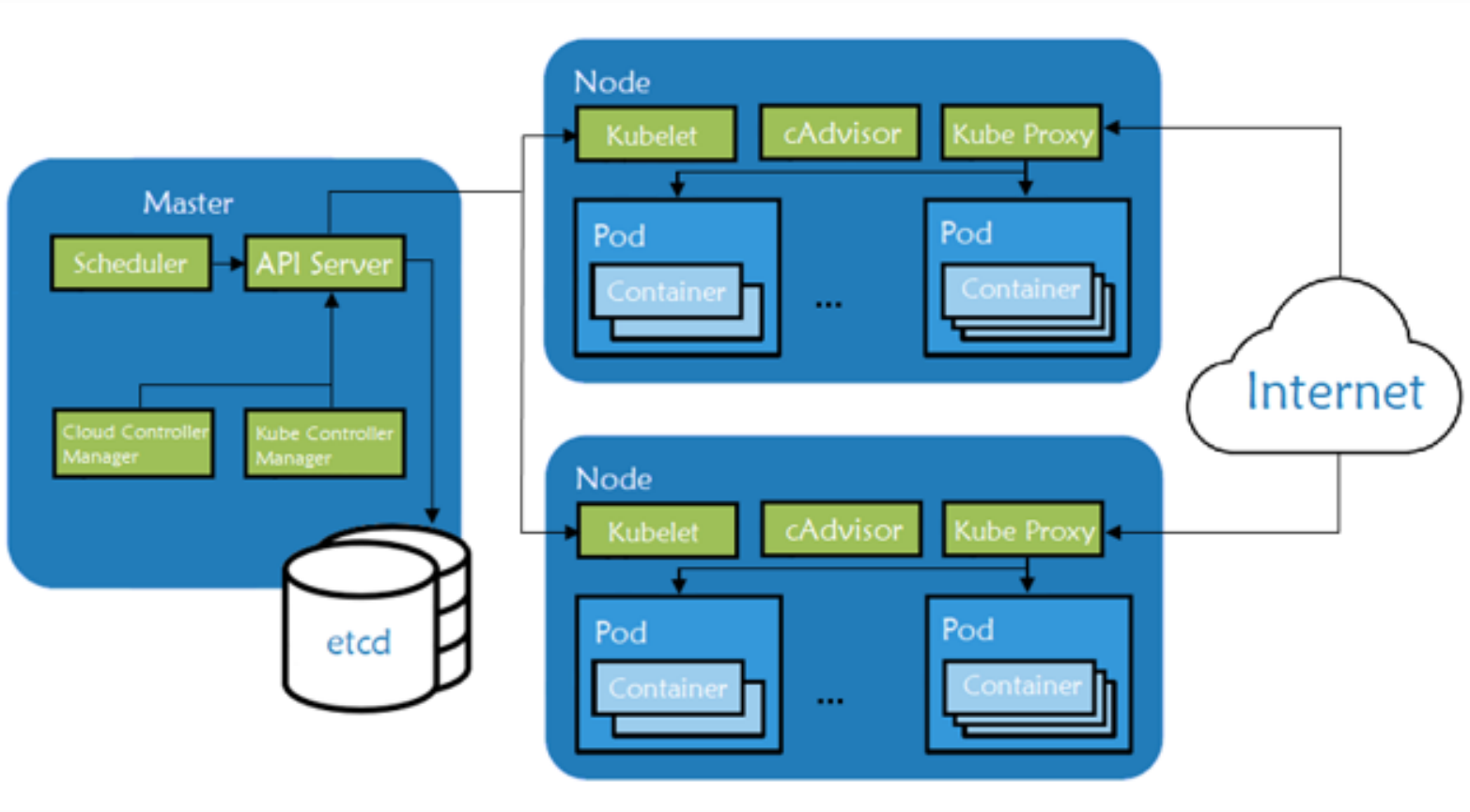
- Propósito: Simplifica la gestión de aplicaciones en contenedores, permitiendo que sean altamente escalables y resilientes.

Limitaciones de Docker

- Docker (docker engine) gestiona completamente la ejecución de un contenedor en un determinado nodo a partir de una imagen, pero no proporciona toda la funcionalidad que necesitamos para ejecutar aplicaciones en entornos en producción.

- Existen diferentes preguntas que nos podemos hacer acerca de esto :
  - ¿Qué hacemos con los cambios entre versiones?
  - ¿Cómo hacemos los cambios en producción?
  - ¿Cómo se balancea la carga entre múltiples contenedores iguales?
  - ¿Cómo se conectan contenedores que se ejecuten en diferentes demonios de docker?
  - ¿Se puede hacer una actualización de una aplicación sin interrupción?
  - ¿Se puede variar a demanda el número de réplicas de un determinado contenedor?
  - ¿Es posible mover la carga entre diferentes nodos?

Arquitectura

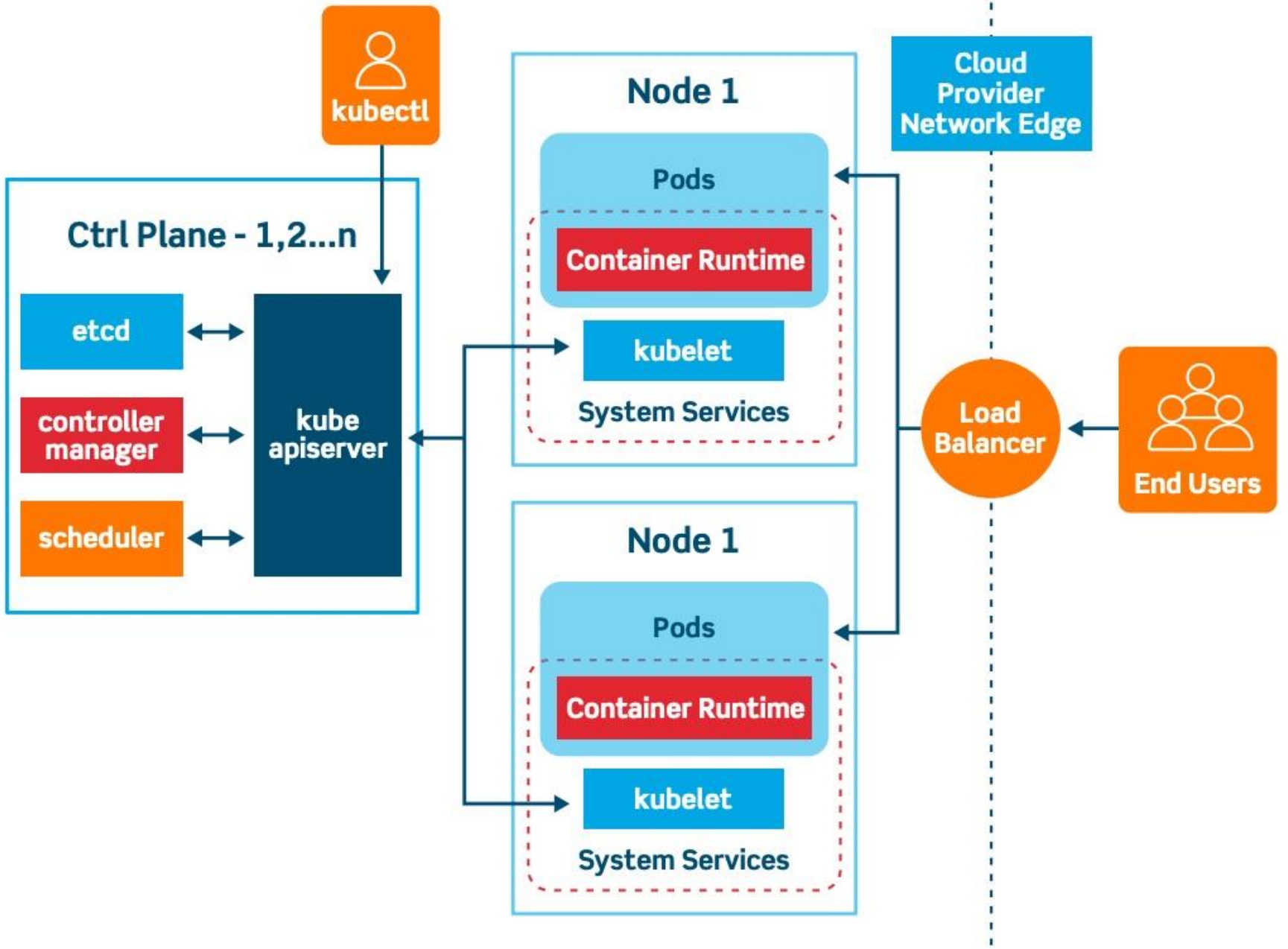


- Un cluster de kubernetes está formado por un nodo Master y de 2 a n nodos Worker.

Arquitectura

- **Nodo maestro (Master Node):** Es el cerebro de Kubernetes y coordina el clúster. Incluye:
- **Kube API Server**: Maneja las solicitudes externas y sirve como el punto de entrada.
- **Scheduler**: Decide en qué nodos deben ejecutarse los contenedores.
- **Controller Manager**: Asegura que el estado deseado del clúster se mantenga.
- **etcd**: Una base de datos distribuida para guardar toda la configuración del clúster.
- **Nodos de trabajo (Worker Nodes):** Son los responsables de ejecutar las aplicaciones en contenedores. Cada nodo tiene:
- **Kubelet**: Asegura que los contenedores estén ejecutándose correctamente.
- **Kube-proxy:** Gestiona la red para la comunicación entre contenedores y servicios.
- **Contenedor Runtime**: Es el software encargado de ejecutar los contenedores (como Docker o containerd).
- **Container Advisor**, es una herramienta de código abierto diseñada para analizar y exponer datos sobre el uso de recursos y el rendimiento de contenedores en ejecución. Funciona como un proceso en segundo plano (daemon) que recopila, procesa y exporta información sobre contenedores.

Arquitectura



Componentes clave

- **Cluster**: Es el conjunto de nodos que ejecutan aplicaciones en Kubernetes.
- **Nodo**: Una máquina (física o virtual) que ejecuta los contenedores.
- **Pod**: La unidad más pequeña en Kubernetes, que agrupa uno o más contenedores que comparten recursos.
- **Control Plane**: Administra el estado deseado del clúster (p.ej., asegura que haya un número correcto de réplicas).

Cluster

- Es el conjunto de todos los nodos (tanto el plano de control como los nodos de trabajo) que trabajan juntos para orquestar aplicaciones.

- **Networking en Kubernetes:**
  - Kubernetes utiliza un modelo de red plano donde cada Pod obtiene su propia dirección IP.
  - Las políticas de red permiten controlar la comunicación entre Pods y servicios.

- **Objetos de Kubernetes:**
  - Son entidades persistentes que describen el estado deseado del clúster.

  - **Pods**: Unidades básicas de implementación.
  - **Services**: Permiten exponer Pods a otras partes del clúster o al exterior.
  - **Deployments**: Gestionan la creación y el escalado de Pods.
  - **ConfigMaps** y **Secrets**: Almacenan datos de configuración para los Pods.

Control Plane

- El plano de control gestiona **el estado deseado del clúster**, distribuye las tareas y supervisa los nodos. Los componentes clave incluyen:

- **API Server**:
  - Es la puerta de entrada a Kubernetes, proporcionando una interfaz para interactuar con el clúster.
  - Administra solicitudes (por ejemplo, desde kubectl) y las valida antes de pasarlas al almacenamiento o a otros componentes.
- **etcd**:
  - Es una base de datos distribuida que almacena el estado de todo el clúster.
  - Es altamente consistente y garantiza la sincronización entre los nodos.

- **Scheduler**:
  - Decide en qué nodo ejecutar cada Pod en función de la disponibilidad de recursos y las restricciones definidas.
  - Optimiza la utilización de los recursos en el clúster.

- **Controller Manager**:
  - Coordina diferentes "controladores" (**controllers**) que se encargan de garantizar que el estado deseado (definido en los manifiestos) se cumpla.

  - Ejemplos:
  - ReplicaSet Controller: asegura el número deseado de réplicas de Pods.
  - Node Controller: supervisa los nodos y detecta fallos.

Flujo de trabajo

- Un **usuario o administrador define un estado deseado** (por ejemplo, una aplicación de **3 réplicas**) mediante manifiestos YAML enviados al API Server.
- El **Scheduler asigna Pods** a los nodos disponibles.
- Los **Kubelets en cada nodo aseguran que los Pods se ejecuten según lo especificado**.
- El **Controller Manager supervisa y corrige cualquier desviación** del estado deseado.
- **Kubernetes** se encarga de **escalar**, **actualizar** y recuperar la aplicación automáticamente.

Funcionalidades principales

- **Escalado automático**: Ajusta **dinámicamente** la **cantidad de Pods** según la demanda.
- **Autocuración**: **Reemplaza Pods fallidos** automáticamente.
- **Despliegues declarativos**: **Define** el **estado deseado** de la aplicación y Kubernetes lo mantiene.
- **Balanceo de carga**: **Distribuye el tráfico** entre los Pods disponibles.
- **Actualizaciones continuas**: Permite **realizar despliegues** y actualizaciones **sin interrumpir** la disponibilidad.

Casos de uso

- **Microservicios**: Gestión de aplicaciones desglosadas en componentes pequeños y autónomos.
- **Cargas de trabajo híbridas**: Ejecuta aplicaciones tanto en la nube como en entornos locales.
- **Procesamiento en lote**: Ideal para tareas que requieren muchos recursos momentáneamente.
- **Entornos de desarrollo y prueba**: Simplifica la configuración y creación de entornos aislados.

Ventajas

- **Portabilidad**: **Compatible** con múltiples proveedores de nube y entornos locales.
- **Escalabilidad**: Gestiona fácilmente el crecimiento de aplicaciones y recursos.
- **Resiliencia**: **Mantiene las aplicaciones funcionando a pesar de fallos** en los contenedores o nodos.
- **Eficiencia operativa**: Optimiza el uso de los recursos de infraestructura.

Los controladores en Kubernetes

- En **Kubernetes**, los **controladores** (controllers) son componentes clave que **se encargan de garantizar que el estado real del clúster coincida con el estado deseado definido por el usuario**.
  - Actúan como un sistema de "bucle de control", detectando discrepancias y realizando las acciones necesarias para corregirlas.

- Cómo funcionan los controladores:
  - **Definición del estado deseado**: El usuario describe el estado deseado en un manifiesto (archivo YAML/JSON), como el número de réplicas de un servicio o la configuración de un objeto.
  - **Supervisión constante**: El controlador supervisa el estado actual del clúster a través del API Server.
  - **Corrección**: Si detecta una discrepancia entre el estado real y el deseado, toma medidas para corregirla automáticamente.

Tipos de controladores I

- **ReplicaSet Controller:**
  - Garantiza que haya un número específico de Pods en ejecución en todo momento.
  - Es el responsable detrás de los objetos Deployment.

- **Deployment Controller:**
  - Gestiona la creación y el escalado de despliegues.
  - Permite actualizaciones continuas y reversión de versiones si algo sale mal.

- **Node Controller:**
  - Supervisa los nodos del clúster y actúa si un nodo deja de estar disponible (por ejemplo, redistribuyendo Pods).

- **Job Controller:**
  - Gestiona trabajos que deben completarse una vez (como tareas en lote).

Tipos de controladores II

- **DaemonSet Controller:**
  - Asegura que un Pod específico esté ejecutándose en todos (o un subconjunto) de los nodos del clúster, útil para tareas como monitoreo.

- **StatefulSet Controller:**
  - Administra aplicaciones con requisitos de estado persistente, como bases de datos.

- **Service Controller:**
  - Configura servicios para exponer aplicaciones dentro o fuera del clúster.

- **Horizontal Pod Autoscaler (HPA):**
  - Escala dinámicamente los Pods basándose en métricas como el uso de CPU o memoria.

- **CronJob Controller:**
  - Administra tareas programadas que se ejecutan en horarios específicos.

Herramientas en línea

- **KillerKoda**: https://killercoda.com/
  - Nos permite tener entornos interactivos de determinados productos.
  - Tenemos de múltiples entornos, pero hay uno específico de K8s.
  - Tendremos que validarnos con Google o GitHub, etc.
  - No son muy potentes, y de vez en cuando te echan y hay que volver a entrar.