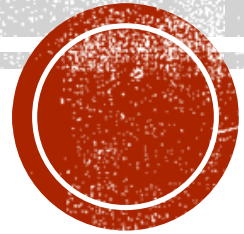
PRÁCTICA 4: KUBERNETES



Daniel Cascado Caballero

Mª José Morón Fernández

ÍNDICE

kubernetes

- **Introducción**
- Conceptos
- Arquitectura
- Modelo de red
- Práctica

MASCOTAS VS GANADO

Mascotas

- *non–cloud native* era.
- Máquina: ente individual
- Despliegue manual
- Fallo: re-despliegue con sus dependencias.
- Se conoce el SW de cada máquina

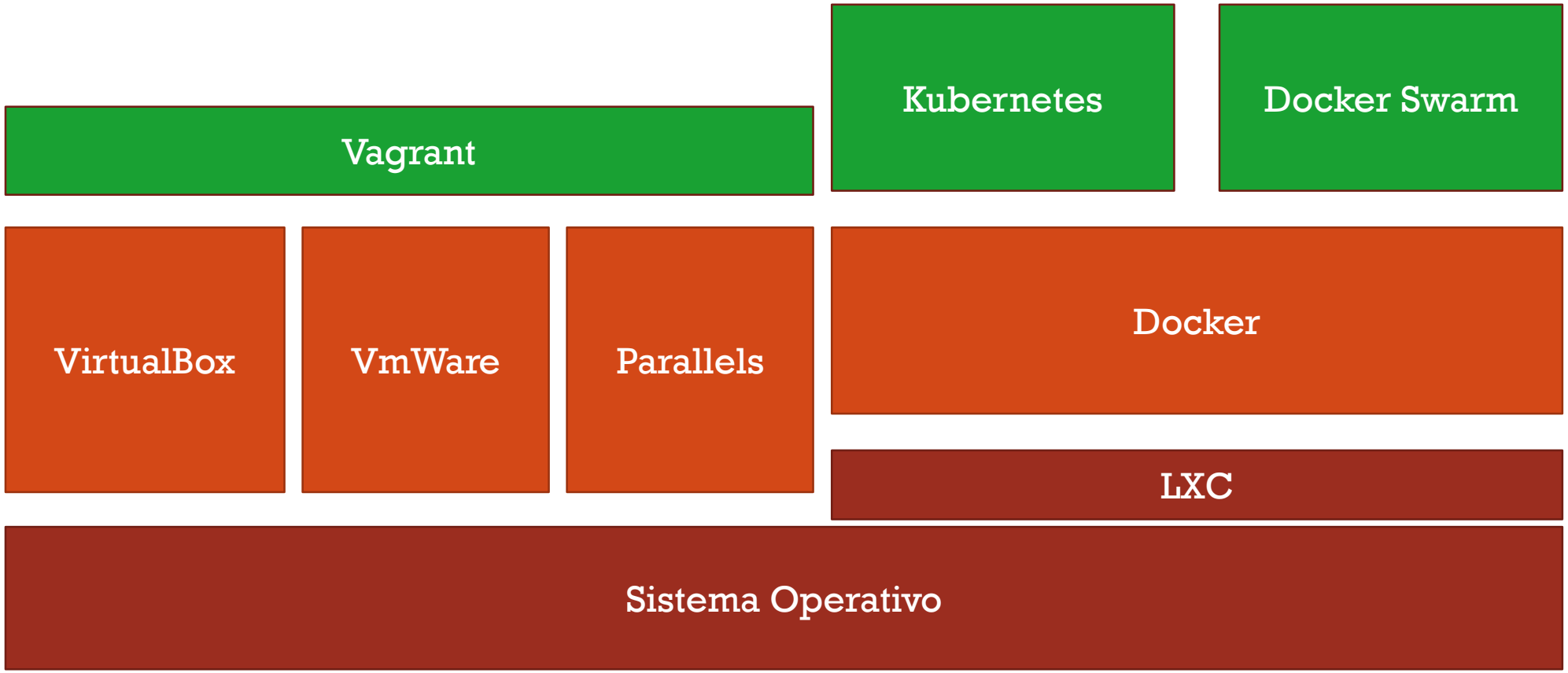
Ganado

- Máquina: ente anónimo.
- Despliegue automático
- Fallo : Reemplazo de forma autónoma y automática
- Permite escalar cómodamente el HW $\Rightarrow\uparrow$ Elasticidad

EL PROBLEMA A RESOLVER

- **Problema**: Se necesita un sistema de gestión de contenedores que proporciones escalado y tolerancia a fallos
- **Solución**: Kubernetes: Orquestación de contenedores:
  - Planifica la ejecución de contenedores en máquinas físicas o virtuales.
  - Monitorización y solución de fallos => garantiza el escalado.
- Desplegable en cluster físico o virtual.
- Restricciones de empaquetamiento:
  - Entorno de despliegue
  - Configuración del cluster.

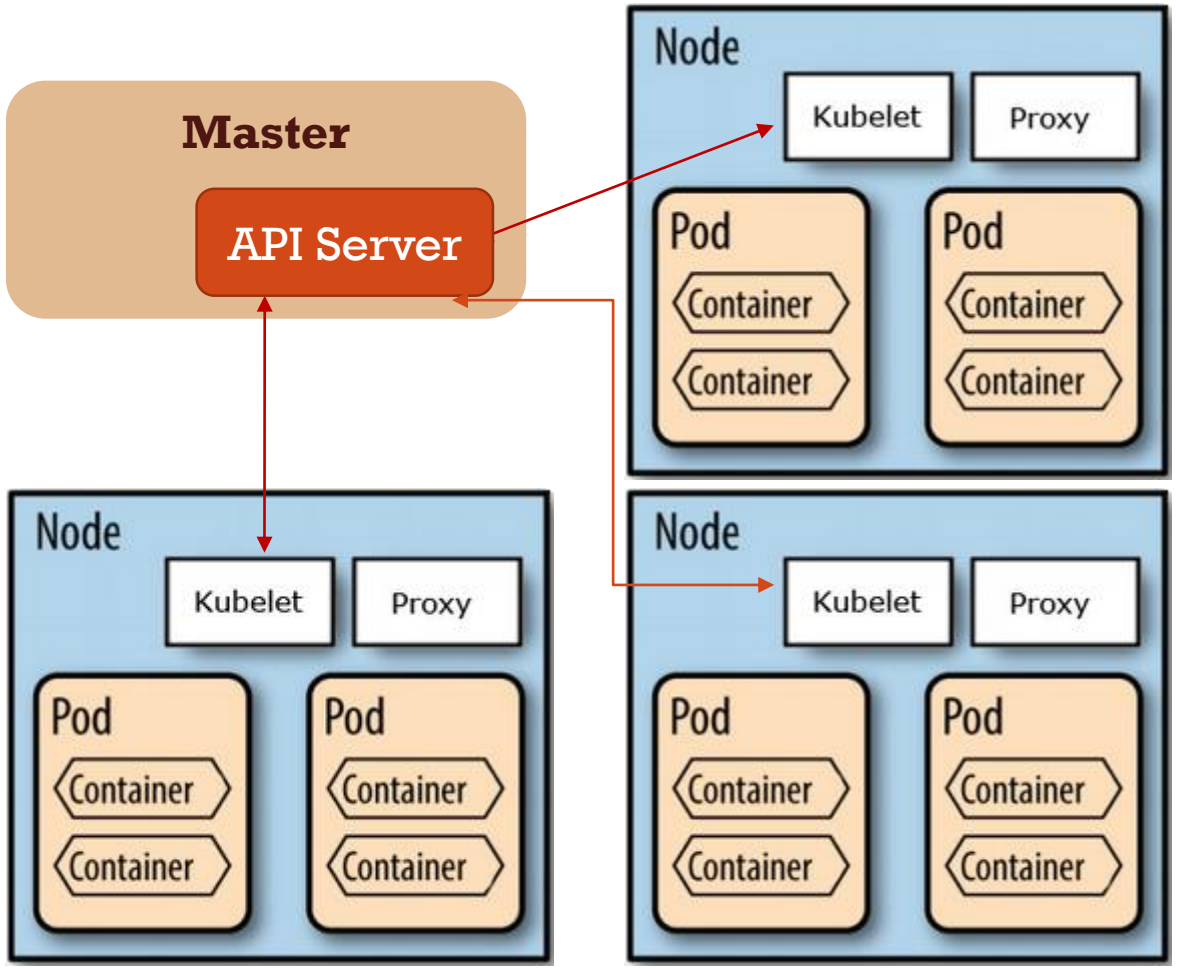
EL CAMINO A LA ORQUESTACIÓN...



ÍNDICE

CONCEPTOS

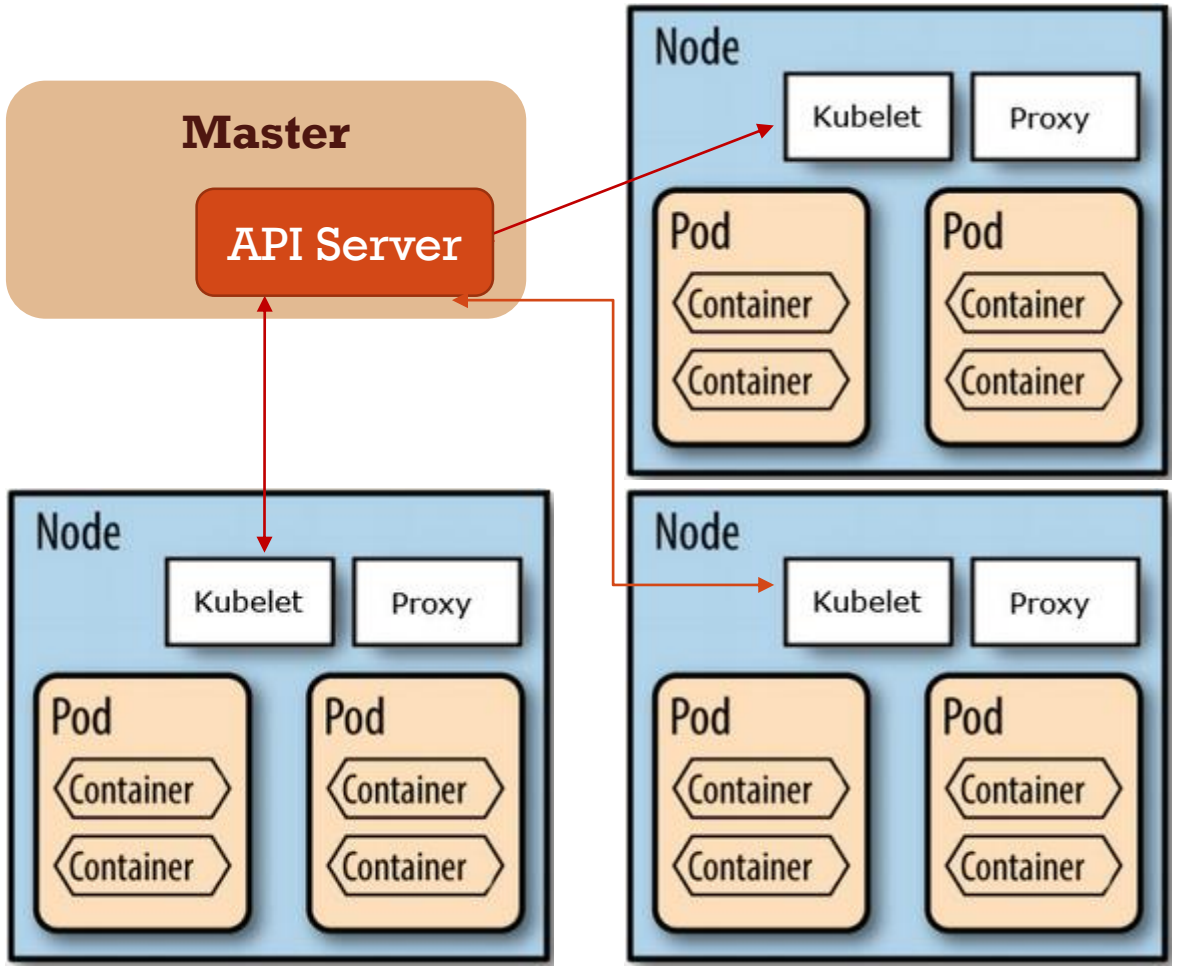
- **Cluster**: grupo de nodos, almacenamiento y recursos de red
  - Sistemas compuestos de múltiples nodos
  - Pueden contener clústeres virtuales
- **Nodo**= host físico o virtual = MINION
  - Ejecutar pods.
  - Gestionados por un master Kubernetes.
- **Master**: plano de control de Kubernetes. Residen en un solo nodo.
  - Componentes
    - API server, *scheduler* , controlador de gestión.
  - Funciones
    - Funcionamiento global
    - planificación a nivel de cluster de los pods
    - gestión de eventos.



CONCEPTOS

Pod: Unidad de trabajo en Kubernetes.

- Uno más contenedores (UUID): EFIMEROS
  - Almacenamiento compartido entre contenedores
  - Contenedores
    - Comparten IP y espacio de puertos ⇒ comms x localhost o interfaz comm procesos estándar.
- Controladores y conjuntos de replica:
  - Gestionan un grupo de pods identificados por un label selector y aseguran que un cierto número están siempre en ejecución.
  - **Controladores de réplica:** comprueban los miembros por igualdad de nombre
  - **Conjuntos de réplica:** Pueden usar selección basada en conjuntos.



CONCEPTOS

- **Etiquetas (*Labels*):** son parejas de clave-valor
  - Agrupar conjuntos de objetos (pods)
  - 1 objeto N etiquetas. 1 etiqueta N objetos (relación n a n)
  - Sintaxis estricta
- **Anotaciones (*Annotation*):**
  - Permiten asociar metadatos arbitrarios con objetos Kubernetes.
- **Selectores de etiqueta (*Label selectors*):** Utilizados para seleccionar objetos a partir de sus etiquetas.
  - *kubectl get pods -l environment=production,tier=frontend*
  - *kubectl get pods -l 'environment in (production),tier in (frontend)'*

CONCEPTOS

- **Servicios:** Grupo de pods, identificados por una etiqueta.
  - Exponen funcionalidad a usuarios o a otros servicios.
  - IP virtual para el servicio.
  - Operan a nivel TCP/UDP (nivel 3).
  - Descubrimiento por DNS o variables de entorno.
  - Balanceo configurable opcionalmente.
- **Volumen**:
  - Persistencia de los datos de un pod.

CONCEPTOS

- **Secret**:
  - Objetos pequeños con información sensible: credenciales y tokens.
  - Accesibles por el API server de Kubernetes
  - Dentro de un pod se almacenan en memoria (por seguridad)
    - Se almacenan como texto plano en etcd,
  - Pueden montarse como ficheros en pods (volúmenes secretos).
    - Puede ser montado en múltiples pods.
  - Predefinidos por Kubernetes para sus componentes. El usuario puede crear los suyos.

CONCEPTOS

Name:

- 1 objeto: nombre + UUID => referencia para la API
  - 253 caracteres y utilizar caracteres alfanuméricos, guiones (-) y punto(.)
- Nombre reutilizable. UUID generado por Kubernetes.

Namespace:

- Espacio de nombres de un cluster virtual.
- 1 cluster físico:
  - Varios clústeres virtuales segregados por espacio de nombres.
- Cluster virtual:
  - Sólo se pueden comunicar solamente a través de interfaces públicas.
- Se puede elegir el nodo donde se ejecutan sus pods.
  - Comparten almacenamiento persistente del pod.

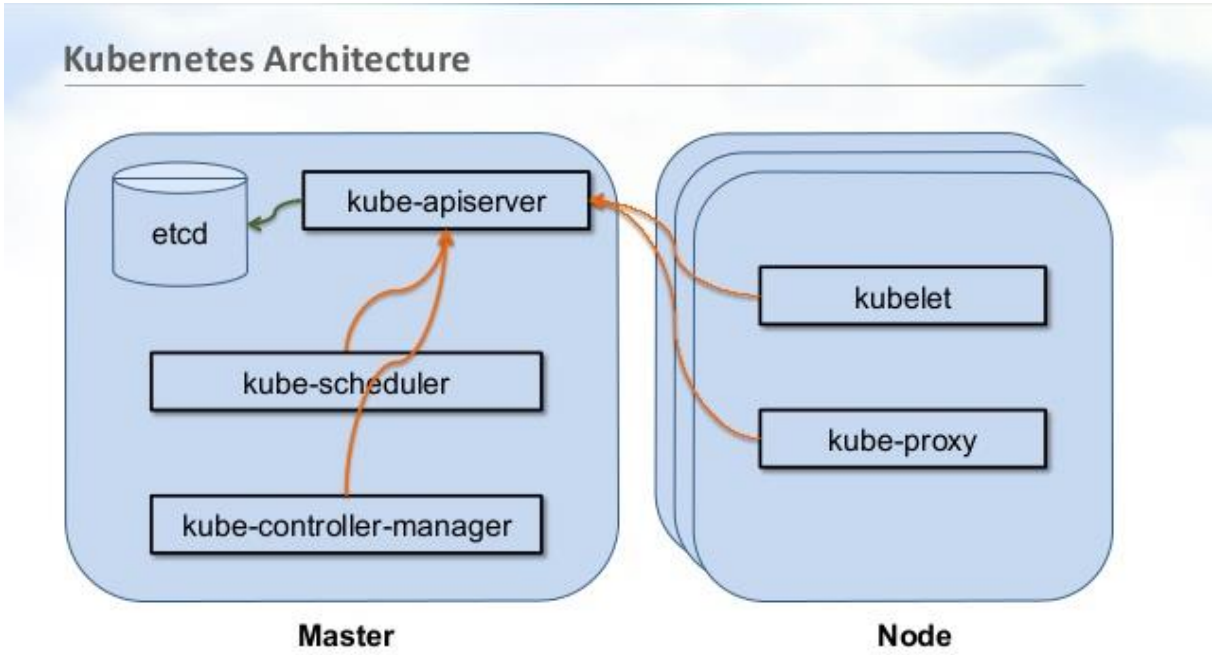
ÍNDICE

COMPONENTES

Master components

Normalmente se ejecutan en un nodo, pero en un cluster muy grande o de alta disponibilidad, pueden estar distribuidos en múltiples nodos.

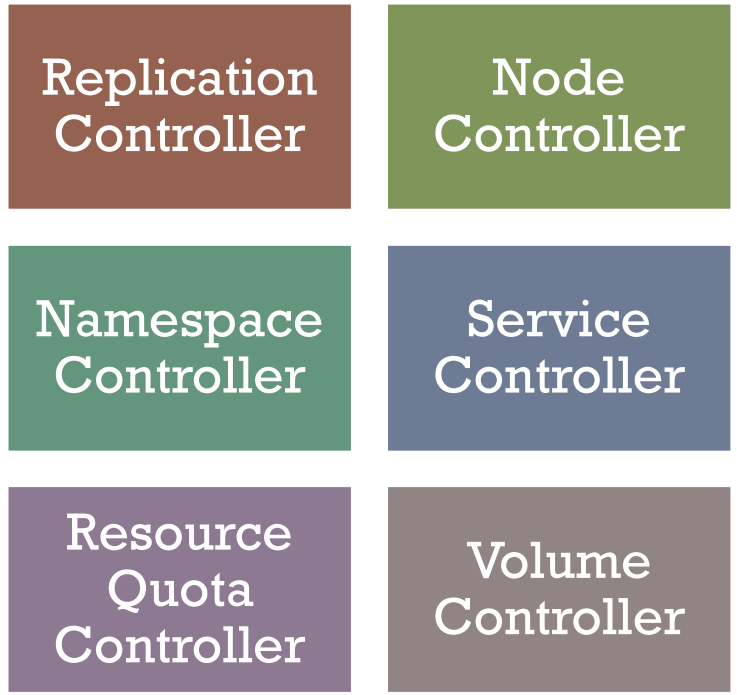
- **API server ()**:
  - Expone la API REST de Kubernetes.
  - Stateless: Puede escalar horizontalmente con facilidad
    - Almacena datos en el cluster etcd.
  - Es el plano de control de Kubernetes.
- **Etcd**:
  - Almacenamiento de datos distribuidos altamente fiable.
  - Almacena el estado del cluster completo.



COMPONENTES

Master components

- **Controller manager:**
  - **Objetivo**: Dirigir el cluster hacia el estado deseado ⇒ Todos estos controladores supervisan el estado del cluster a través de la API
  - Controlador de réplicas
  - Controlador de pod
  - Controlador de servicios
  - Controlador de volumen…

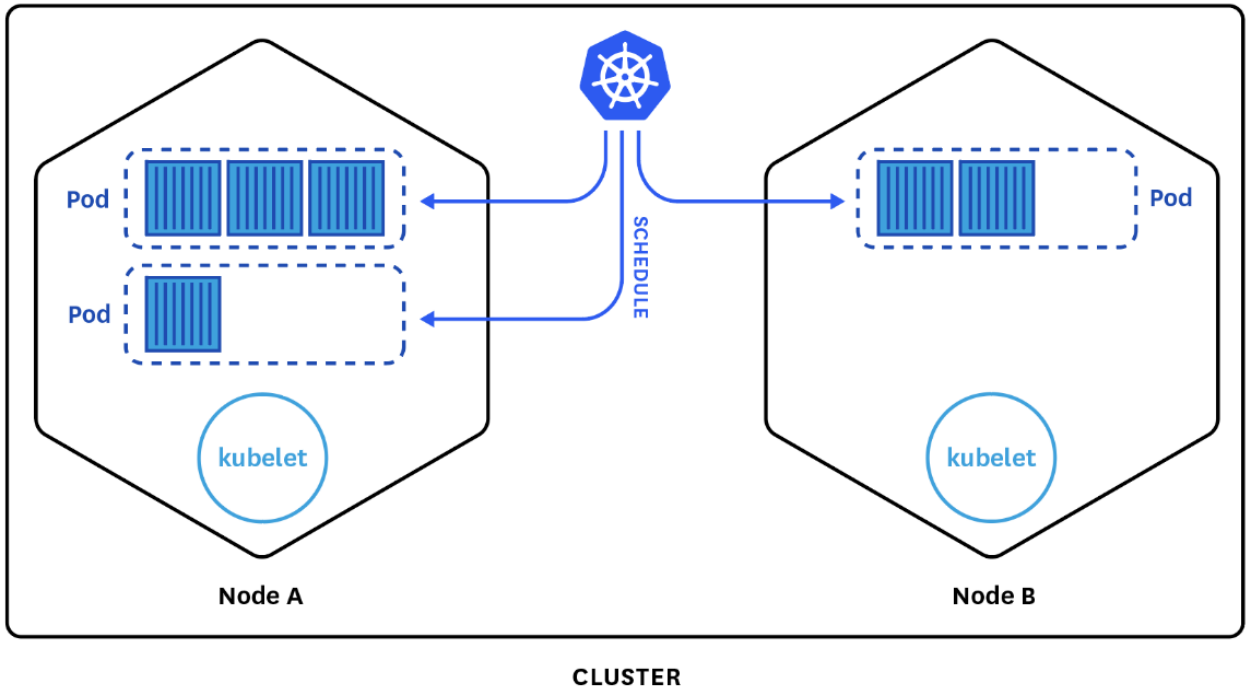


Normalmente se ejecutan en un nodo, pero en un cluster muy grande o de alta disponibilidad, pueden estar distribuidos en múltiples nodos.

COMPONENTES

Master components

- **Scheduler (*kube-scheduler*)**: Responsable de planificar los pods en los nodos ⇒ considerar la interacción de múltiples factores:
  - Requisitos de recursos
  - Requisitos de servicios.
  - Políticas de restricciones HW/SW.
  - Localidad de datos.



COMPONENTES

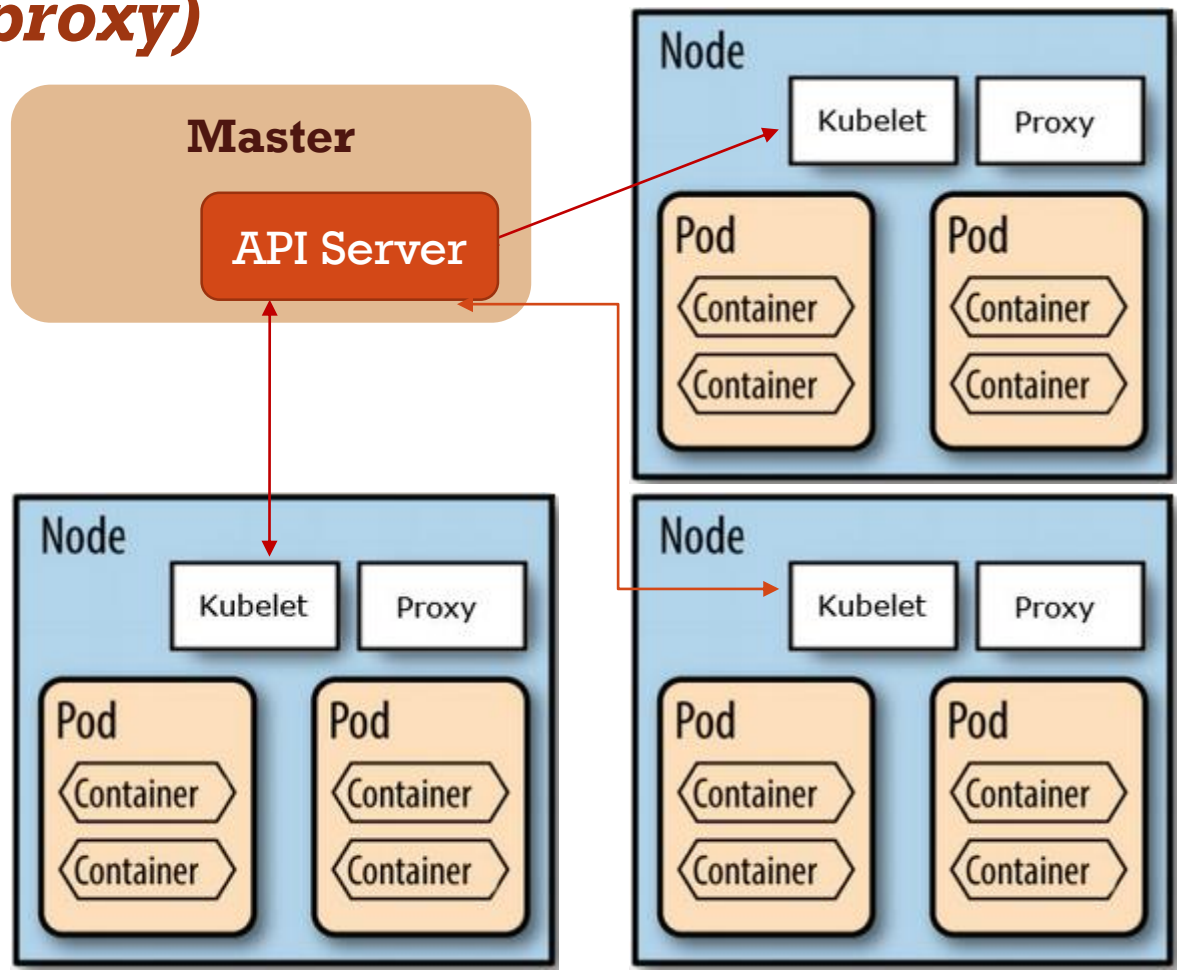
Master components

- **DNS**:
  - Desde Kubernetes 1.3, un servicio DNS es parte del cluster Kubernetes estándar.
  - Es planificado como un pod típico.
  - Cada servicio recibe un nombre DNS.
  - Los pods también pueden recibir un nombre DNS.
  - Muy útil para el descubrimiento automático de servicios.

COMPONENTES

Node components – Proxy (kube proxy)

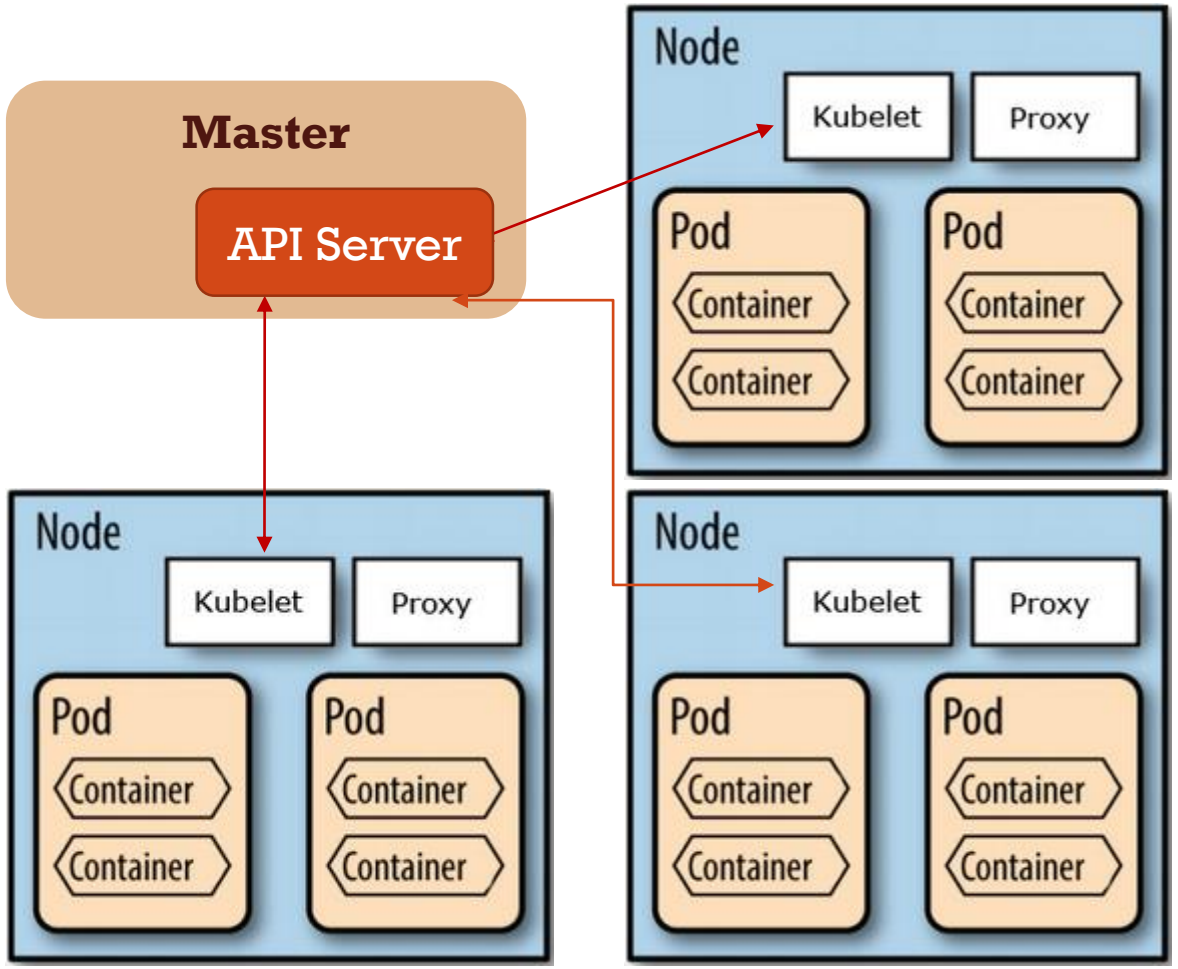
- Actúa como un guardián de red a bajo nivel en cada nodo.
- Expone los servicios Kubernetes localmente y puede hacer redireccionamiento TCP y UDP.
- Encuentra las IPs del cluster a través de variables de entorno o DNS.



COMPONENTES

Node components – Kubelet

- Supervisa comunicación con los master C.
- Gestiona los pods en ejecución:
  - Descargar de los pods secrets desde el API server
  - Montaje de volúmenes.
  - Ejecuta el contenedor del Pod (Docker o Rkt).
  - Notifica el estado del nodo y de cada pod.
- Realiza pruebas de vida del contenedor



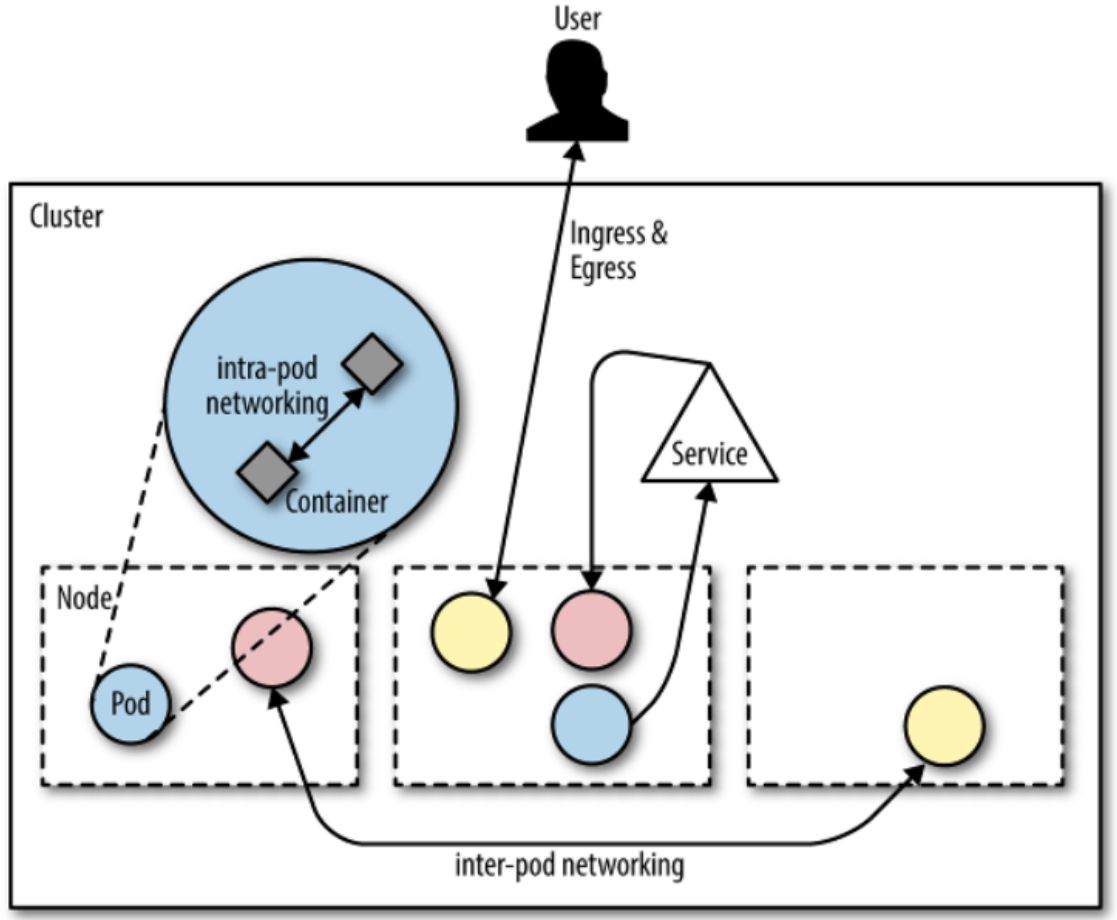
ÍNDICE

GENERALIDADES

- Kubernetes no especifica una solución de *networking* concreta
- **Una IP por pod**: (Equivalente a una MV…)
  - Los nodos pueden comunicarse con otros nodos sin NAT.
  - Los contenedores de un **pod** pueden comunicarse con otros contenedores sin NAT.
    - La IP que un contenedor ve es la misma IP que ve el resto de sus compañeros en el pod.

TIPOS DE TRÁFICO EN RED

- ***Intra-pod***: Todos los contenedores dentro de un pod comparten un espacio de nombres de red y se ven a través de la interfaz *localhost*
- ***Inter-pod***: Los pods pueden comunicarse con otros pods directamente o, preferiblemente, pueden utilizar servicios para comunicarse con otros pods.
- ***Ingress and egress***:
  - *Ingress: Tráfico* entrante desde usuarios externos o apps a pods
  - *Egress*: Llamadas a APIs externas realizadas desde los pods.



INTRA-POD NETWORKING

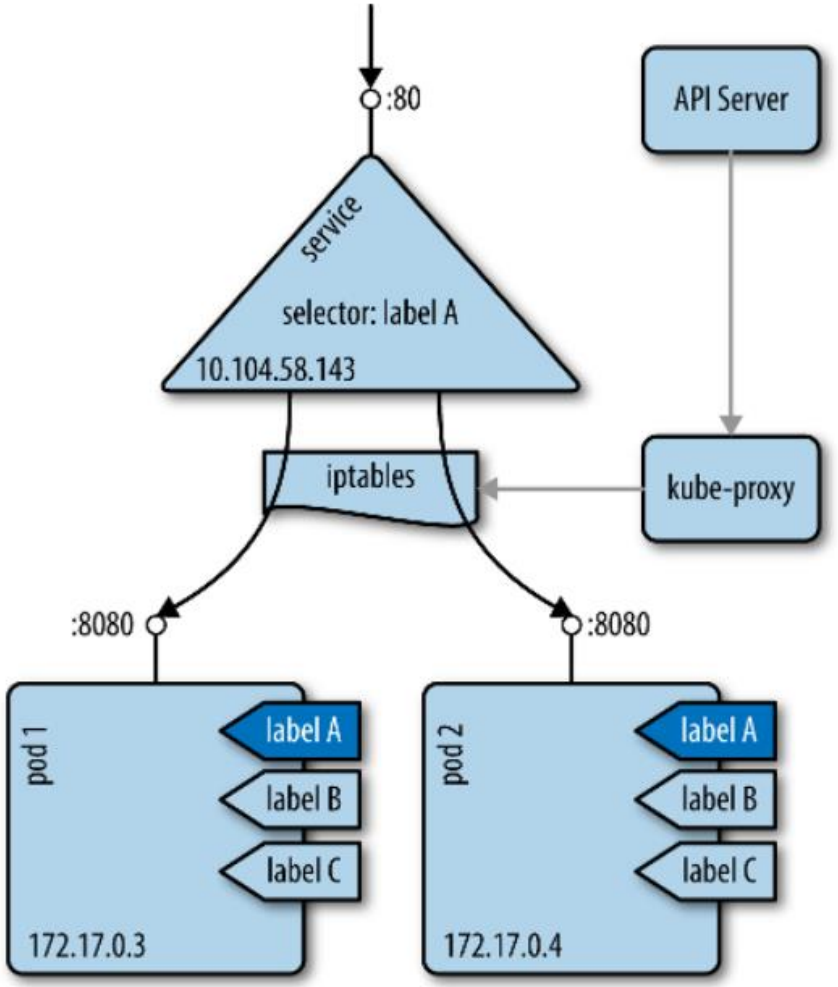
- Contenedor de infraestructura: 1er contenedor que kubelet lanza dentro de un pod
  - Toma la IP del pod y crea el espacio de nombres de la red.
  - Tiene el modo *bridge* activado
  - Si muere, el kubelet mata a todos los contenedores del pod y se reinicia
- El resto de los contenedores del pod:
  - se unen a la red del contenedor de infraestructura.
  - Se adhieren al espacio de nombre utilizando el modo *Joined containers*.
- Todos los contenedores dentro de un pod se pueden comunicar entre sí utilizando *localhost*.
- Espacio de puertos compartido entre contenedores: cuidado con los conflictos

INTER-POD NETWORKING

- Cada pod tiene una dirección IP real enrutable ⇒ pueden:
  - Comunicarse sin proxies o traducciones (como NAT) y sin necesitar gestionar reserva de puertos.
  - Utilizar los puertos conocidos y evitar usar mecanismos de alto nivel de descubrimiento de servicios.
- Se distinguen dos tipos de comunicación inter-pod (*East-West traffic*):
  - Comunicación directa:
    - El pod llamante necesita averiguar la dirección IP del destinatario
      - Riesgo: pods aparecen y desaparecen
  - Los pods usan servicios para comunicarse con otros pods.

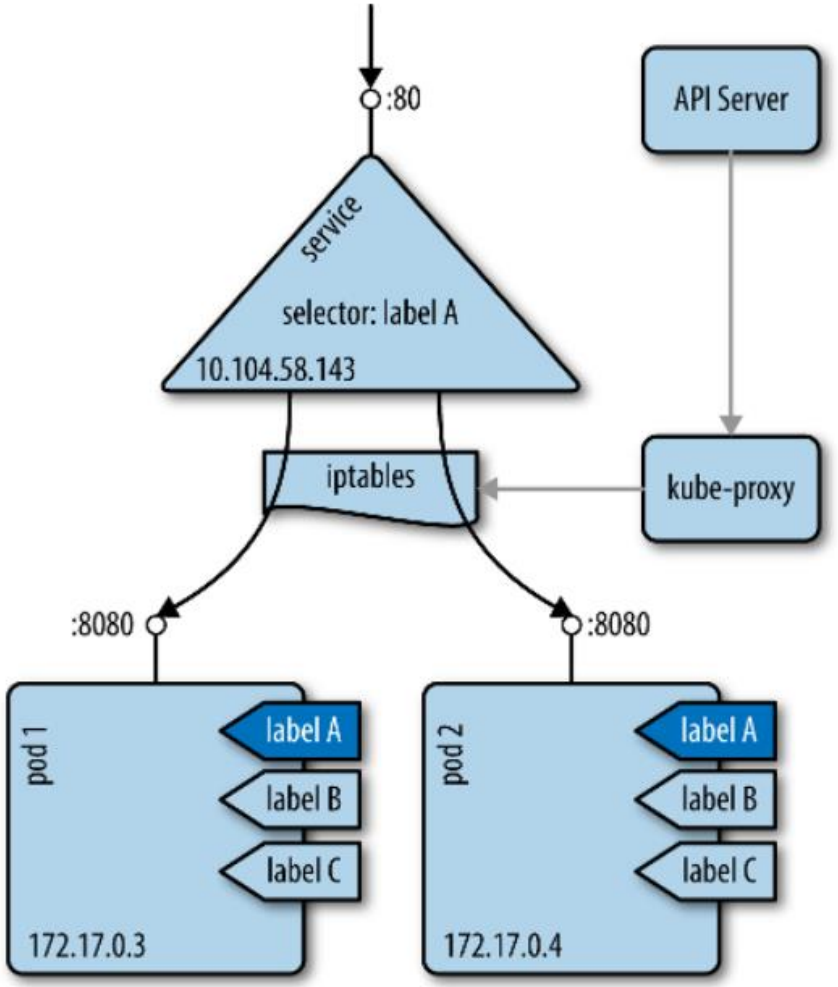
INTER-POD NETWORKING – VIP (VIRTUAL IP)

- **Objetivo**: Proporcionar un acceso estable para direccionar el tráfico a uno o más pods, cuyas direcciones IP van cambiando.
- Un servicio:
  - Proporciona una dirección IP virtual (VIP) estable que puede ser descubierta a través de DNS.
  - Permite a los clientes descubrir a los contenedores en ejecución en los pods y conectarse a ellos.
- La VIP no es una dirección IP real conectada a una interfaz de red



INTER-POD NETWORKING – VIP (VIRTUAL IP)

- Se especifica el conjunto de pods a los que se quiere que un servicio dirija el tráfico a través de un *label selector*:
  - Por ejemplo, para *spec.selector.app=someapp,* Kubernetes debería crear un servicio que enrute el tráfico a todos los pods con una etiqueta *app=someapp*.
- ***kube-proxy***: mantiene el mapping entre el VIP y el pod actualizado.
  - Consulta a la API server para "aprender" de los nuevos servicios en el cluster
  - Actualiza las reglas de iptables de los nodos, para proporcionar la información de enrutamiento necesaria.



INGRESS AND EGRESS (NORTH-SOUTH TRAFFIC)

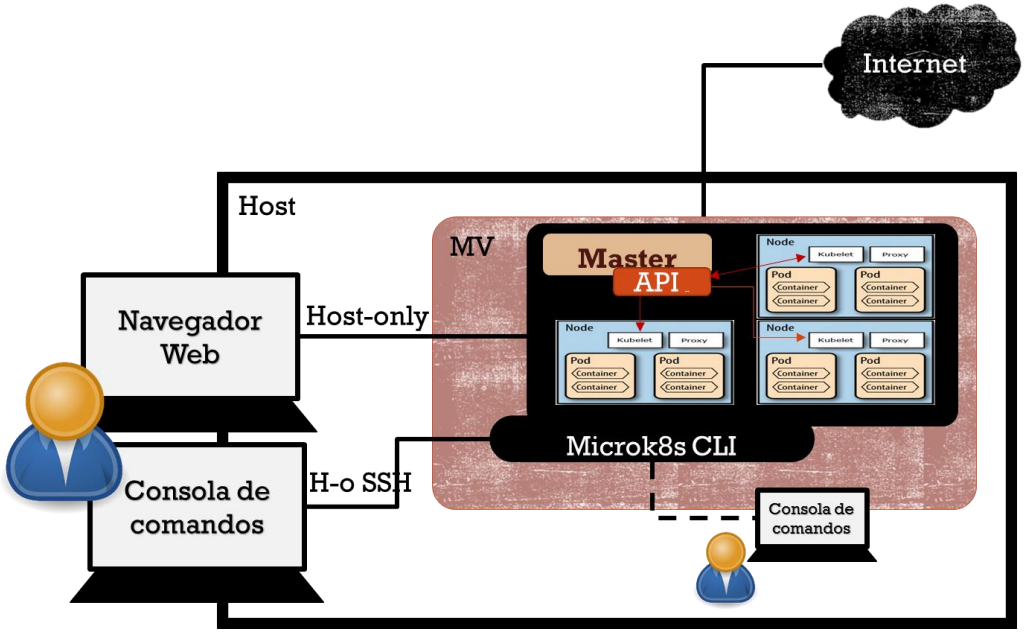
Es el tráfico que fluye entre el interior y exterior de un cluster Kubernetes,

- Alta disponibilidad
- Balanceo de carga de alto rendimiento para los servicios.
- ***Ingress***: Enruta tráfico procedente del exterior del cluster a un servicio del cluster.
  - Característica disponible a partir de Kubernetes 1.2
- ***Egress***: Se centra en cómo una app en un pod llama a un API externa al cluster.
  1. Controlar qué pods está autorizados a tener un *path* de comunicación hacia servicios exteriores
  2. Imponer políticas.

ÍNDICE

DESARROLLO

- Kubernetes en un solo nodo: MicroK8s
- Tutorial básico de Kubernetes
  - Instalación y comprobación de estado
    - Máquina virtual preconfigurada.
    - Interfaz host-only para consola y otros menesteres
  - Manejo básico de componentes del master
    - Dns
    - Dashboard
- Puesta en marcha de un servicio básico (**HITO**)
- Prueba de carga comparativa **(+)**



MÁQUINA VIRTUAL PRECONFIGURADA