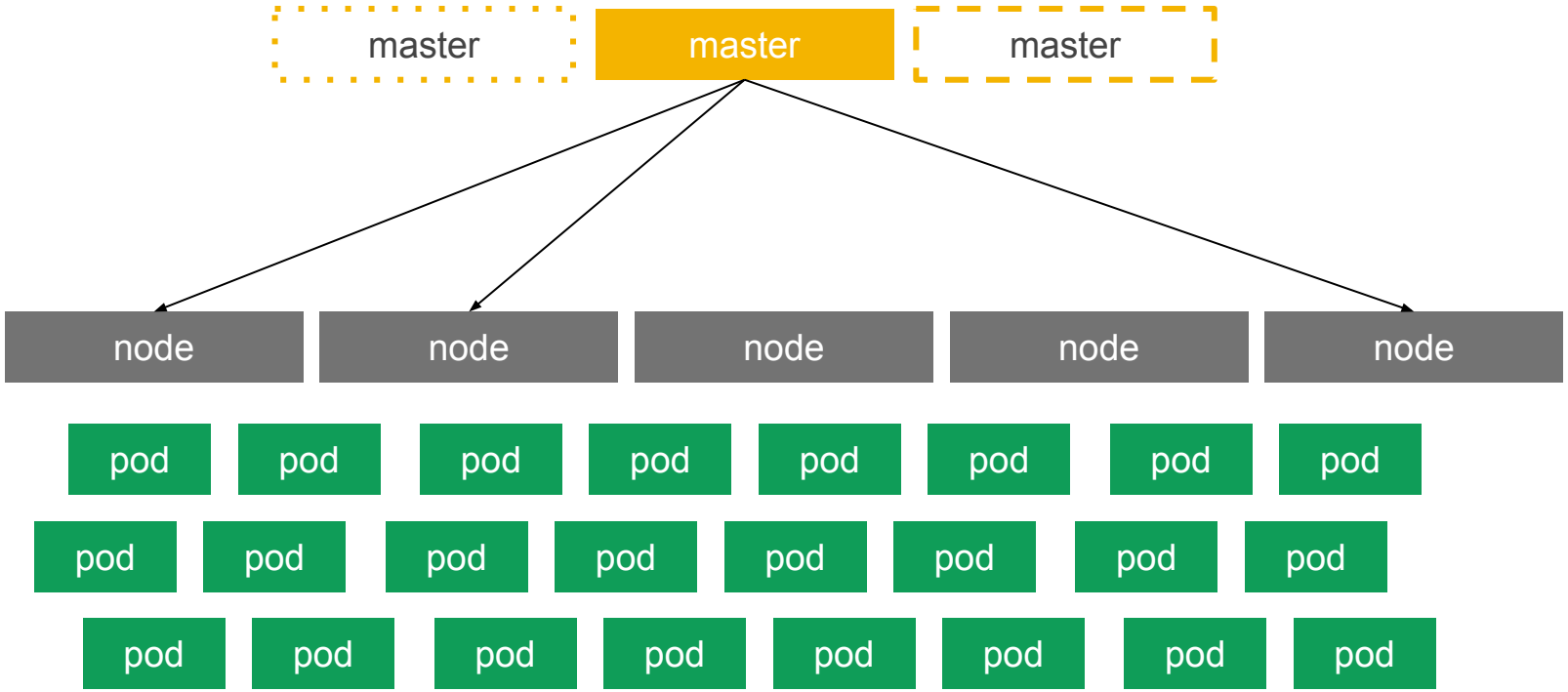
Arquitectura de Kubernetes

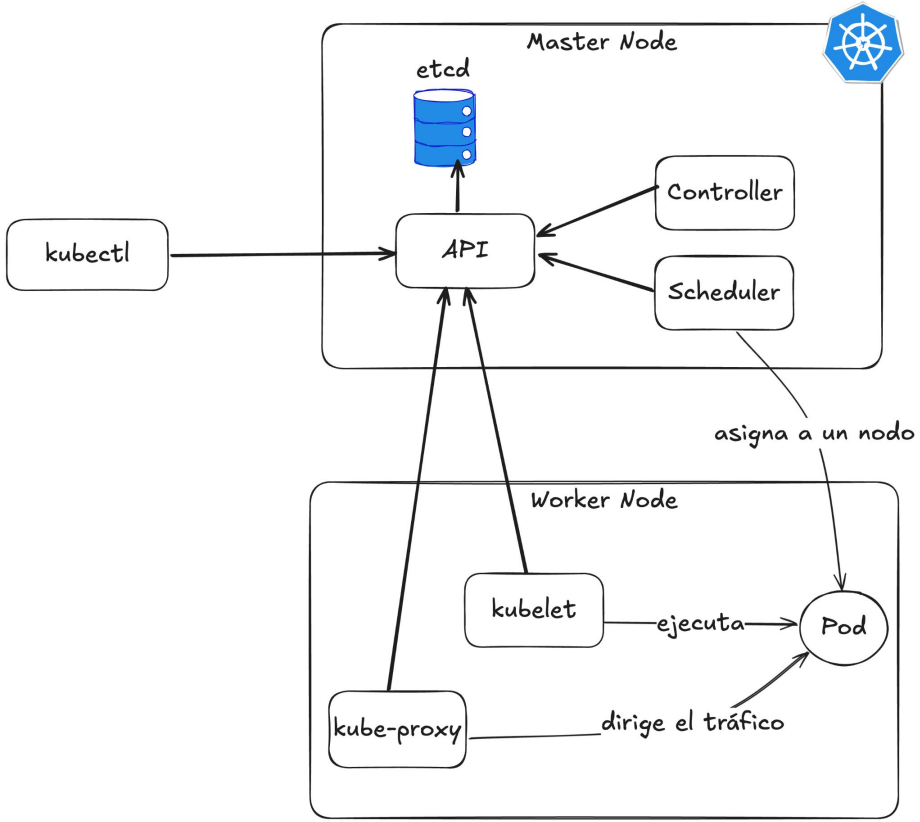
KEEPCODING
Tech School

Arquitectura de Kubernetes

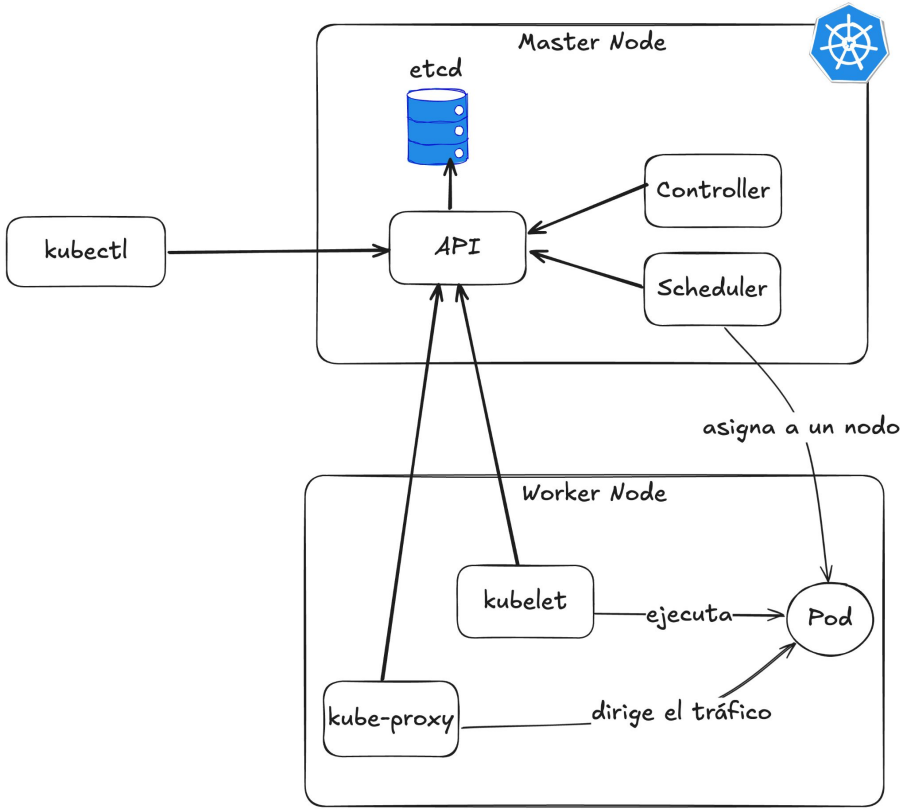


Arquitectura de Kubernetes

- 2 tipos de nodos: **master (control plane)** y **worker**.
- master: gestiona los workers y dispone del API.
- workers: se encargan de ejecutar nuestros contenedores.

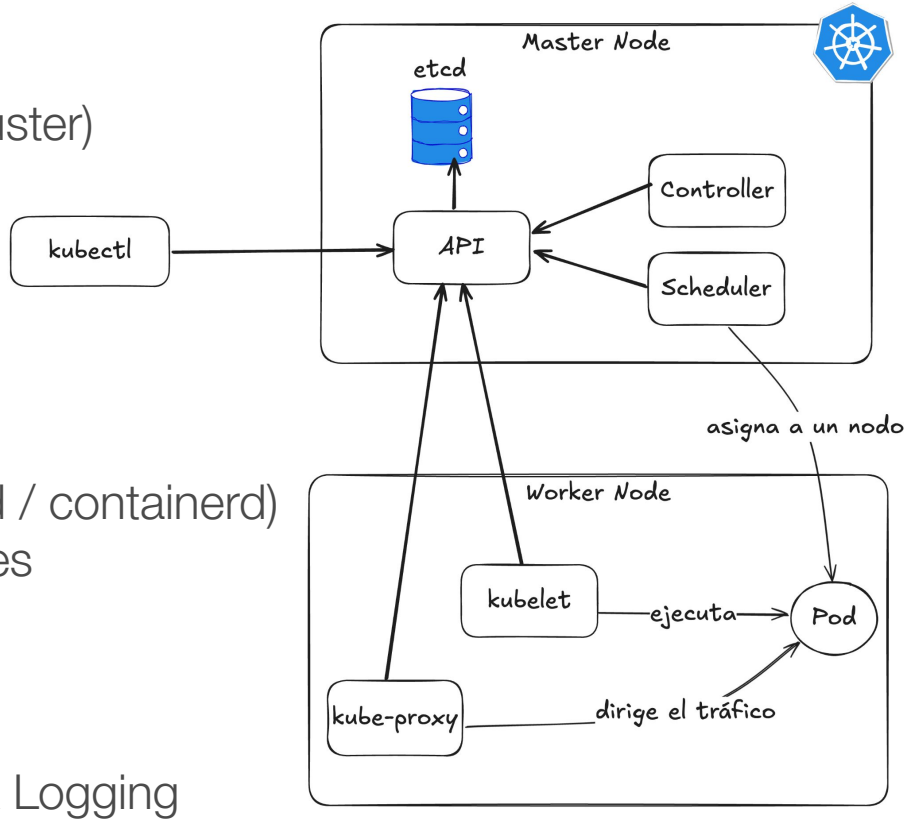


Arquitectura de Kubernetes



Componentes de Kubernetes

- Master (control plane):
  - etcd (podría correr fuera del cluster)
  - kube-apiserver
  - kube-controller-manager
  - kube-scheduler
  - cloud-controller-manager (*)
- Nodos:
  - container-runtime (Docker / rkd / containerd)
  - kubelet → gestión contenedores
  - kube-proxy → tráfico
- Addons:
  - DNS, Dashboard, Monitoring & Logging



etcd

- Es una base de datos consistente y en alta disponibilidad de tipo key/value (KV).
- Es donde Kubernetes almacena todos los datos.
- Si queremos hacer un backup de Kubernetes, etcd es lo que debemos respaldar.
- Tiene que estar correctamente securizado.
- Actualmente Kubernetes soporta etcd 3

https://github.com/etcd-io/etcd

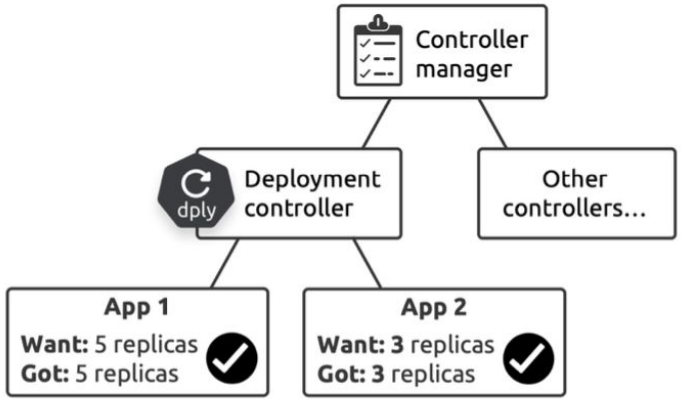
kube-apiserver

- Todas las acciones que realizaremos en k8s serán por API.
- Este componente expone el API hacia afuera.
- Diseñado para escalar horizontalmente y en alta disponibilidad.
- Los "audit logs" del API server guardan el registro de todas las acciones y eventos.
- Extensible por medio de CRDs (Custom Resource Definition)

kube-controller-manager

- Son varios componentes en uno.
  - Deployment controller
  - ReplicaSet controller
  - Endpoints controller
  - Service account and tokens controller

- Los controladores son procesos que observan el estado del sistema, lo comparan con el estado deseado y toman decisiones.



Nigel Poulton, *The Kubernetes Book (2024)*

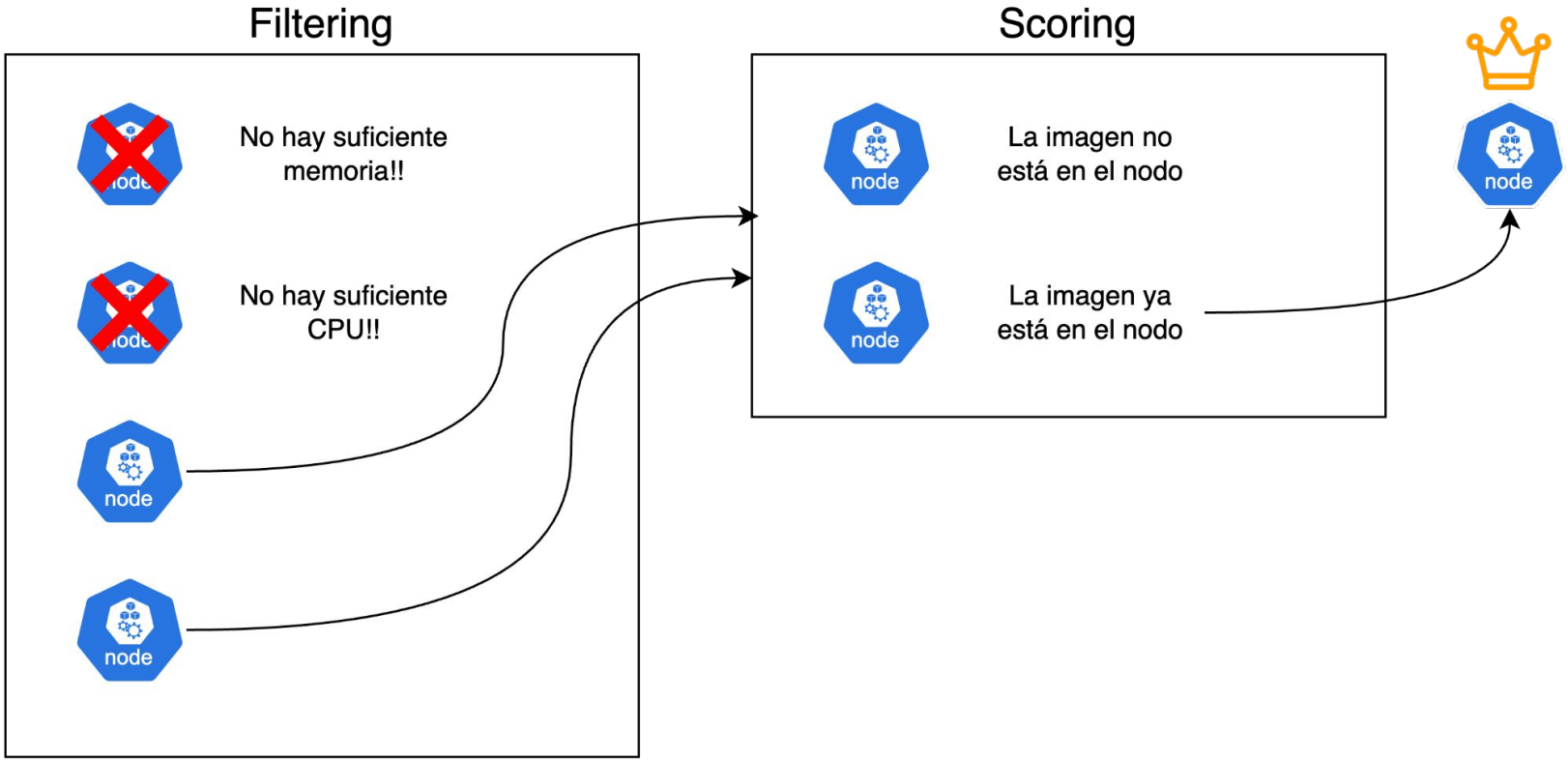
kube-scheduler

- Está “atento” a nuevos componentes que se despliegan en el cluster.
- Se encarga de decidir en qué nodo deben de ejecutarse en base a recursos disponibles en el sistema y opciones de configuración de cada componente a desplegar.
- Sigue el siguiente proceso
  - Observar si hay nuevas tareas registradas en la API server.
  - Identifica nodos capaces de ejecutarlas.
  - Asigna las tareas a los nodos.

kube-scheduler

- Proceso de selección de nodos
  - Identificación de nodos capaces:
    - Verifica taints, reglas de afinidad, CPU y memoria disponibles.
    - Filtra nodos incapaces de ejecutar las tareas.
  - Clasificación de nodos:
    - Factores como:
      - Imagen requerida ya está disponible.
      - CPU y memoria libres.
      - Número de tareas actuales.
    - Cada factor otorga puntos y los nodos con más puntos son seleccionados.
  - Manejo de tareas pendientes:
    - Si no hay un nodo adecuado, las tareas quedan pendientes.
    - Si hay escalado automático, se añade un nuevo nodo y se asigna la tarea.

kube-scheduler



cloud-controller-manager

- Es el componente específico que se encarga de interaccionar con la nube en la que se “apoya” el cluster.
- Habrá un addon específico para cada tipo de cloud.
- Funcionalidades:
  - Node Controller (máquinas virtuales)
  - Route Controller (networking)
  - Service Controller (LB)
  - Volume Controller (discos)

kubelet

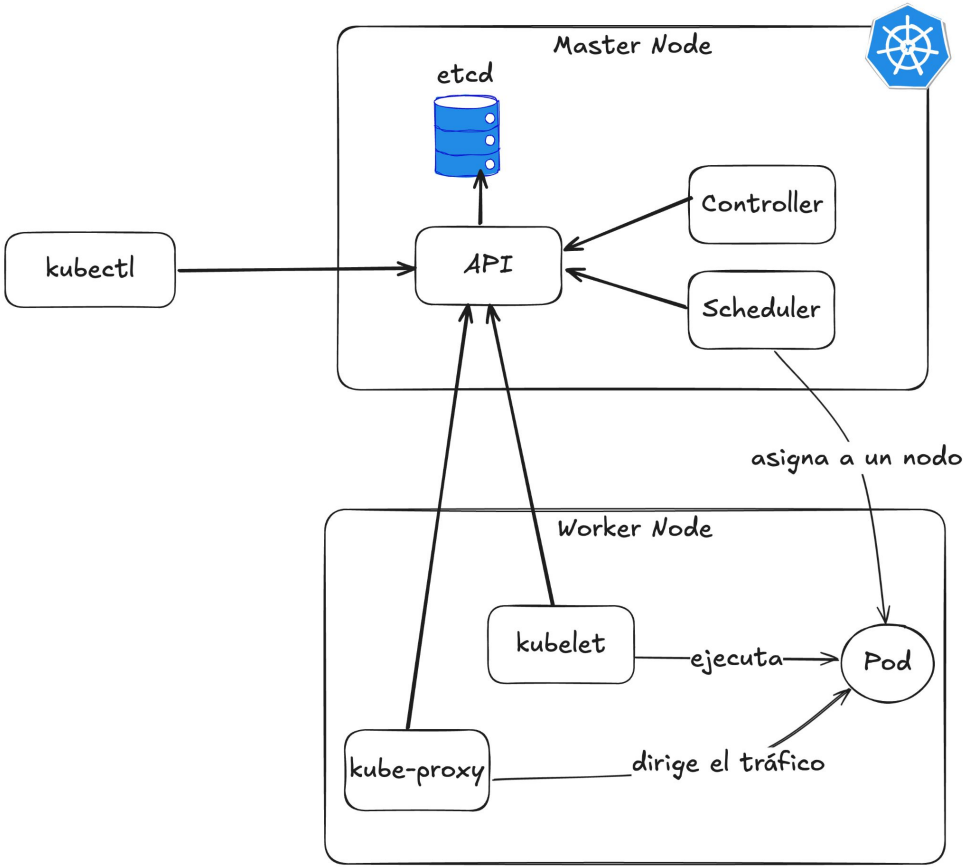
- Es el agente que se ejecuta en cada nodo worker.
- Monitoriza el estado de los PODs y comunica cualquier cambio de estado al API server.
- Observa si hay PODs asignados al nodo que tengan que ejecutarse.
- Si hay nuevos PODs, le pide al runtime que los ejecute.
- Se encarga de que los contenedores de los PODs se estén ejecutando (responsable por ejemplo de liveness y readiness probes).

kube-proxy

- Se ejecuta en cada nodo del clúster y observa los servicios nuevos a través del API Server.
- Componente que sirve para crear una red para comunicar unos pods con otros a través de servicios.
- Programa reglas de iptables en el kernel del nodo para redirigir el tráfico hacia uno de los endpoints del servicio.
- Reescribe las reglas de iptables si los endpoints del servicio cambian (por ejemplo, si los Pods cambian de estado).

kubectl

- Es el CLI, el cliente desde el que trabajaremos para interaccionar con Kubernetes.
  - kubectl get pods
  - kubectl get nodes
  - kubectl describe node <name>



Qué os parece?

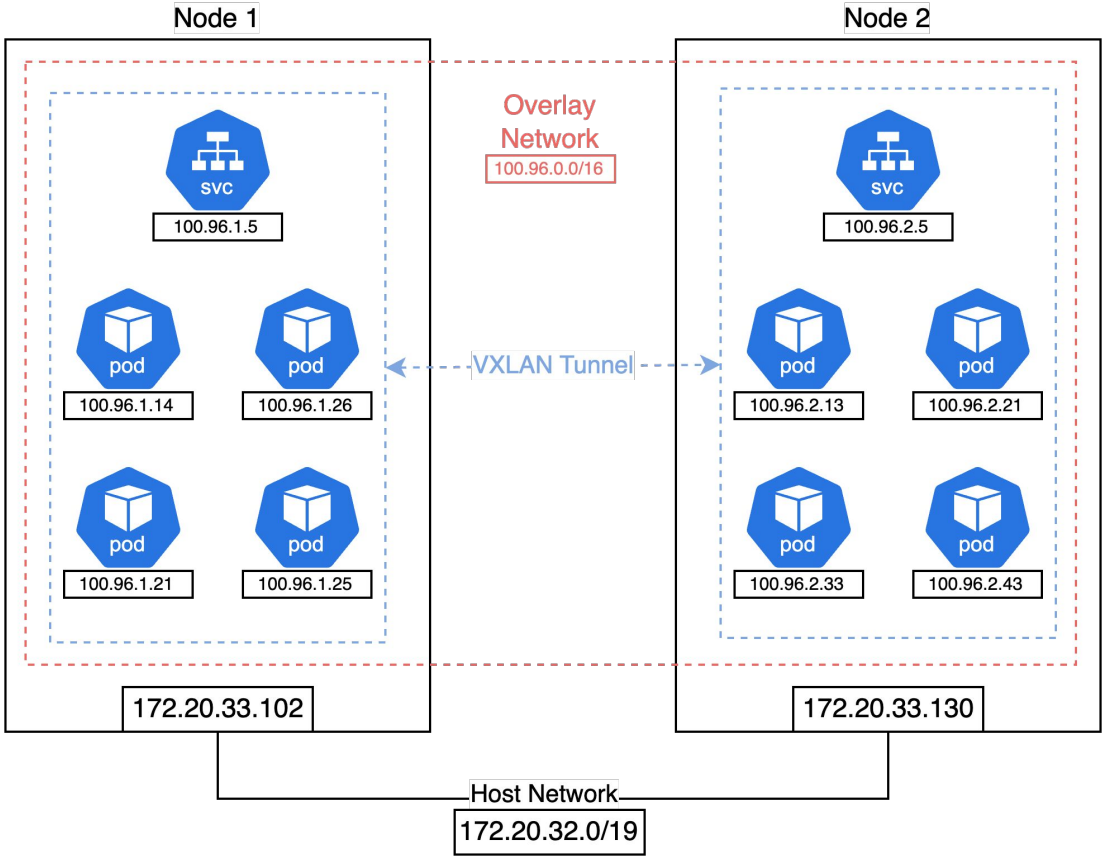
Networking en Kubernetes

- Algo complicada
- Principios básicos, Kubernetes asume / requiere lo siguiente:
  - **Todos los PODs han de poder comunicarse entre ellos sin NAT.**
  - **Todos los nodos han de poder comunicarse con todos los PODs sin NAT**
  - **La IP con la que un POD se ve a sí mismo es la misma IP con la que el resto le ve.**

Networking en Kubernetes

- Implementar lo anterior es un reto teniendo en cuenta cómo funcionaba la red en Docker.
- Para resolver esto necesitamos una "**overlay network**", red superpuesta.
- Esto lo resuelven los plugins de red (CNI) que podemos usar en Kubernetes: Flannel, Calico, OpenvSwitch, etc. Estos plugins utilizarán tecnologías como vxlan y virtualización de red para implementar las comunicaciones dentro del cluster.
- No hay una solución de networking estándar en Kubernetes (al usuario normal no le afecta mucho la tecnología usada)

Networking en Kubernetes



Networking en Kubernetes

- Rangos de red (CIDRs):
  - Rango real de los nodos de Kubernetes
  - Cluster CIDR: rango completo de red overlay
  - Rango para PODs en cada nodo (una subred por nodo)
  - Rango para Servicios (subred)
- Importante
  - **Entender que hay una red dentro de los nodos del cluster que es completamente virtual, y los propios nodos están a su vez en la red real** (parecido a las redes en docker pero en este caso abarcando múltiples hosts).

Enlaces de interés

- Arquitectura:
  - https://kubernetes.io/es/docs/concepts/architecture/
  - https://aprenderdevops.com/arquitectura-de-kubernetes/
  - https://www.redhat.com/es/topics/containers/kubernetes-architecture
- Networking
  - https://kubernetes.io/docs/concepts/cluster-administration/networking/
  - https://kubernetes.io/docs/concepts/services-networking/#the-kubernetes-network-model
  - CNI: https://www.techtarget.com/searchitoperations/tip/Explore-network-plugins-for-Kubernetes-CNI-explained

- Qué concepto creéis que es más difícil de entender?

En la próxima clase…

- Veremos cómo instalar Kubernetes y las diferentes opciones que hay.
- Daremos los primeros pasos con **PODs** y **Services.**
- Nos familiarizaremos con la herramienta **kubectl** mediante una serie de ejercicios prácticos.

En la próxima clase…

- Qué necesitaremos?
  - Instalar Minikube -> https://minikube.sigs.k8s.io/docs/start/
  - Instalar Kubectl -> https://kubernetes.io/docs/tasks/tools/#kubectl

KEEPCODING

Tech School

Madrid | Barcelona | Bogotá