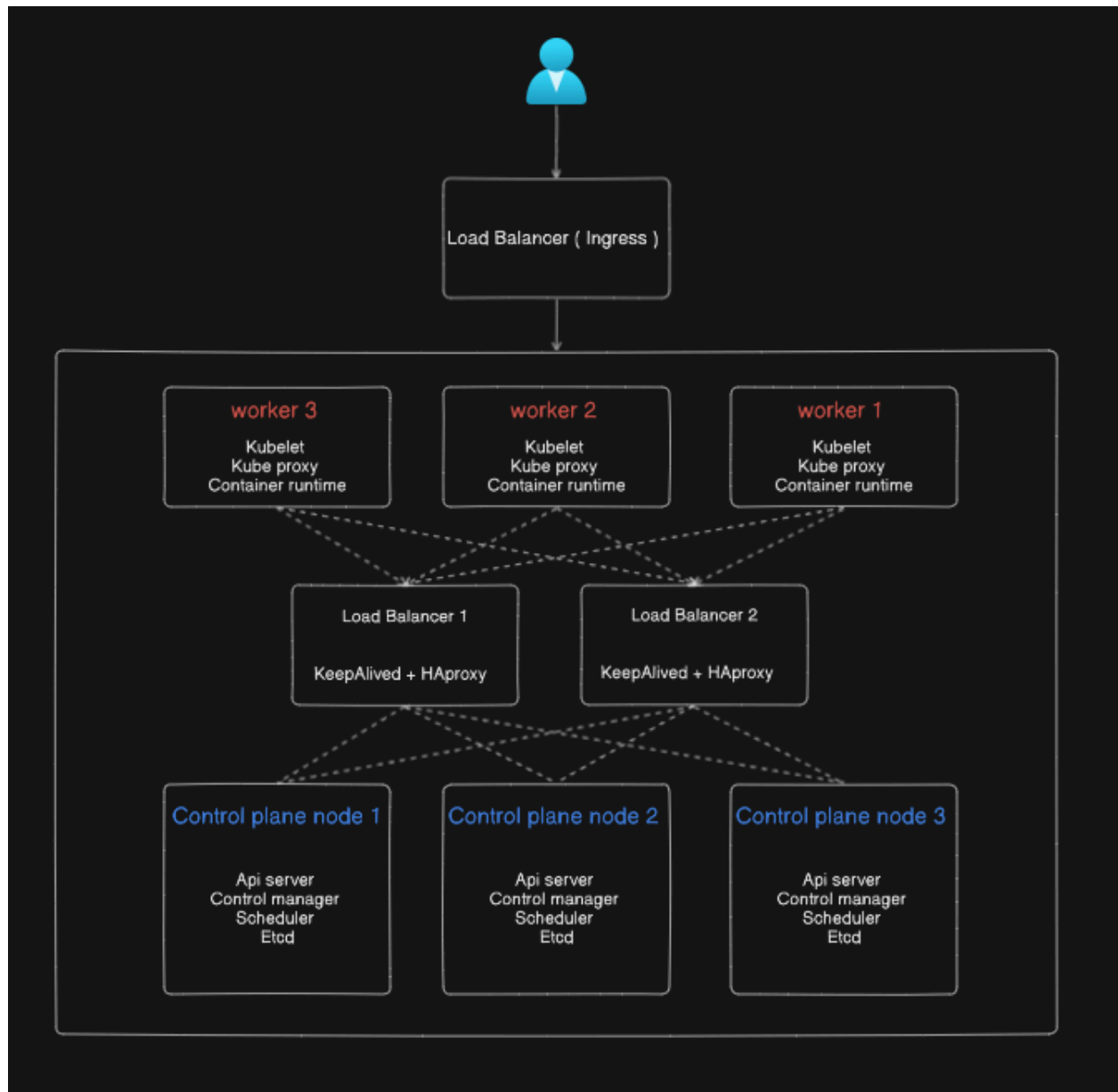


# Highly Available Kubernetes Cluster



💡 C'est quoi Kubernetes ?

Kubernetes (ou K8s pour les intimes), c'est comme un **chef d'orchestre** qui gère des applications sur plein de machines.

Tu veux lancer une appli ? Il la déploie, la surveille, redémarre si elle tombe, et s'occupe de la faire tourner là où il y a de la place.

---

## C'est quoi un cluster Kubernetes ?

Un **cluster**, c'est un **groupe de machines** (physiques ou virtuelles) qui bossent ensemble pour héberger des applications.

- Y'a des **chefs (masters)** qui pilotent
  - Et des **ouvriers (workers)** qui font tourner les applications
- 

## Pourquoi utiliser un cluster multi-nœuds *hautement disponible* ?

Pour éviter que tout plante si **une machine tombe en panne**.

Avec plusieurs nœuds, Kubernetes peut **continuer à fonctionner même si une partie du cluster a un souci**.

 **Disponibilité continue, résilience, et scalabilité.**

---

## Les composants du master node (plan de contrôle)

Ce sont les cerveaux du cluster :

- **API Server** : La porte d'entrée, c'est lui qui reçoit toutes les commandes.
  - **Scheduler** : Il décide où lancer les applications.
  - **Controller Manager** : Il surveille que tout fonctionne bien.
  - **etcd** : C'est la base de données du cluster (état global du système).
- 

## Les composants des worker nodes (nœuds de travail)

Ce sont les muscles :

- **Kubelet** : Il suit les ordres du master.
- **Kube-proxy** : Il gère le réseau.

- **Runtime de conteneur** (comme Docker) : Il fait tourner les applis.

## 🧩 HAProxy et Keepalived, c'est quoi ?

- **HAProxy** : répartit les requêtes entre les serveurs de contrôle ( `kube-apiserver` ).
- **Keepalived** : fournit une **IP virtuelle** → si une machine tombe, une autre prend le relais automatiquement.

## ? Pourquoi 3 nœuds master ? Pas 1, ni 2 ?

C'est quoi un **quorum** ?

Un **quorum**, c'est le **nombre minimum de nœuds** (dans un groupe) qui doivent être **en ligne et d'accord** pour que le système puisse **prendre des décisions** (écrire, élire un leader, etc.)

🎯 En Kubernetes, le quorum est utilisé par etcd, la base de données des masters.

Formule du quorum :

$$\text{quorum} = (\text{nombre total de nœuds} / 2) + 1$$

- Le quorum est toujours une **majorité**.
- Sans quorum, etcd est bloqué → donc le **cluster Kubernetes est figé !**

Différence entre **2 masters** et **3 masters**

🧠 Cas	Nombre total de masters	Quorum requis	Tolérance aux pannes
❌ 2 masters	2	2	0 (aucune panne possible)
✅ 3 masters	3	2	1 panne possible

Exemple avec **2 nœuds masters**

- `master1` et `master2`

- **Quorum = 2**
  - Si 1 tombe, il ne reste qu'un seul nœud → quorum **non atteint** ❌
- ➡ **Cluster bloqué** même s'il te reste un master

Exemple avec **3 nœuds masters**

- `master1` , `master2` , `master3`
  - **Quorum = 2**
  - Si 1 tombe (peu importe lequel), il reste 2 nœuds → quorum **ok** ✅
- ➡ **Cluster continue à fonctionner normalement**

## 🌐 C'est quoi l'Ingress ?

L'**Ingress** est comme un portier d'hôtel 🛎️. Il gère les routes HTTP (ex: `/admin` , `/app` ) et les redirige vers les bonnes applis.

🧠 **Exemple :**

`www.monsite.com/admin` → Pod A  
`www.monsite.com/shop` → Pod B

## 👤 Comment un utilisateur accède à l'application ?

1. Il tape une URL (ex : `myapp.com` )
2. Le **load balancer externe** redirige vers l'Ingress
3. L'Ingress envoie la requête vers le bon pod (ex: application Joget)
4. Le pod répond → l'utilisateur voit son interface

## ⚙️ Aider un DevOps à configurer les VM (pour Joget + DB)

Voici une proposition simple et réaliste :

Rôle VM	CPU	RAM	Stockage	Détails
Master Node (x3)	2	2 Go	40 Go	etcd + control plane

Worker Node (x2+)	4	8 Go	60 Go	Joget + apps
Base de données (DB)	4	8 Go	60 Go	MySQL/Postgres
Load Balancer (x2)	1	1 Go	20 Go	HAProxy + Keepalived
NFS Server (option)	1	2 Go	100 Go	Stockage partagé

## Les 3 Load Balancers dans ton cluster Kubernetes HA :

### 1. Load Balancer Externe

- **Rôle** : Point d'entrée pour les utilisateurs finaux (navigateur, mobile, API, etc.)
- **IP publique/virtuelle** : ex. `192.168.1.100` ou `myapp.company.com`
- **Redirige vers** : les 2 load balancers internes (HAProxy)
- **Souvent géré par** : Keepalived (pour failover) + floating IP

### 2. Load Balancer Interne 1 (HAProxy 1) et Interne 2 (HAProxy 2)

- **Rôle** : répartissent le trafic vers les `kube-apiserver` des 3 nœuds de contrôle
- **Ils sont en redondance** grâce à Keepalived
- **Ils assurent que le plan de contrôle reste disponible même si un LB interne tombe**

## Comment fonctionne un cluster Kubernetes avec plusieurs masters ?

Dans une architecture **multi-masters**, plusieurs serveurs (nœuds de contrôle) hébergent chacun une copie des composants critiques de Kubernetes ( `kube-apiserver` , `etcd` , `scheduler` , etc.).

👉 Cela permet au cluster de continuer à fonctionner même si un ou deux masters tombent en panne. C'est le principe de la **haute disponibilité (HA)**.

---




## Pourquoi faut-il un Load Balancer interne ?

Le Load Balancer interne agit comme un **point d'entrée unique** vers les `kube-apiserver` répartis sur les masters.

Tous les composants (kubelet, kubectl, etc.) communiquent avec **cette seule adresse** (ex: `192.168.193.135:6443`), sans se soucier de quel master est actif.

---

### ✅ Ses avantages :

-  Répartit le trafic entre les masters
  -  Si un master tombe, les autres prennent le relais automatiquement
  -  Simplifie la configuration : une seule IP pour accéder au cluster
- 

## ❌ Et si on n'en met pas ?

Sans Load Balancer :

- Tu dois taper manuellement sur les IPs des masters
- Pas de bascule automatique en cas de panne
- Le cluster devient fragile et perd le bénéfice de la haute dispo