2026 k8s 需知

從 5 年前（2021年）的 K8s 基礎出發，系統性地梳理了 **2026 年現代化 Kubernetes 架構**的演進。

身為系統建設負責人與架構師，以下是整理的核心重點：

1. 基礎架構的根本變革

- **脫離 Docker：** K8s 已移除 Dockershim，底層運行環境轉為 **containerd/CRI-O**。
- **輕量化首選： K3s** (K8s/2) 成為主流，適合邊緣運算與本地測試。
- **工具鏈升級：** 建議在 Mac 使用 **OrbStack** 快速啟動 K3s；使用 **Kind** 模擬多節點生產環境。
- **遷移利器：** 使用 **Kompose** 將舊有的 `docker-compose.yaml` 轉換為 K8s 物件。

2. 網路與流量調度 (最核心的演進)

- **Gateway API 取代 Ingress：** 這是 2026 年最重要的標準。透過角色分離（GatewayClass, Gateway, HTTPRoute）解決權限與配置混亂問題。
- **Service 角色重新定義：**
  - **ClusterIP：** 內部通訊基石，實現 Pod 間的備援與負載平衡。
  - **NodePort：** 您的實體環境中，對接手動 SLB 的關鍵入口。
  - **LoadBalancer：** 雲端自動化專用，在地端手動環境可被 NodePort 取代。
  - **ExternalName：** 用於將叢集外部服務（如舊資料庫）內部化。
- **現代交警 Traefik：** K3s 內建，支援自動發現與 Gateway API，負責精準指揮 Traffic。

3. 高階治理與維運

- **版本與權限控制：** 透過 **Label (v1/v2)** 區分版本，並利用 **HTTPRoute Weight** 實現金絲雀發佈或藍綠部署。
- **Service Mesh (服務網格)：** 如 Istio 或 Cilium，處理東西向流量的加密 (mTLS)、觀測與斷路保護。
- **YAML 物件現代化：** 建議在既有清單中加入 **StorageClass** (動態儲存)、**NetworkPolicy** (零信任安全) 與 **ServiceAccount** (權限管理)。

給架構師的實務建議路徑：

1. **本地實驗：** 在 **OrbStack** 啟動 K8s，將一個簡單服務從 `docker-compose` 透過 `kompose` 搬上去。
2. **網路重構：** 練習不寫 Ingress，直接用 **Gateway API + HTTPRoute** 透過 NodePort 對接您的實體 SLB。
3. **穩定性強化：** 試著部屬兩個版本的 Pod，並在 `HTTPRoute` 調整權重進行流量切分。

在 2026 年的本地開發環境中，**Minikube** 已不再是唯一或首選的工具。對於像您這樣追求效能、且在 Mac (Apple Silicon) 環境下工作的架構師來說，目前的替代方案主要分為兩類：**極簡整合型**與**專業容器型**。

以下是目前最推薦的替代方案：

---

1. OrbStack (Mac 用戶的頂級首選)

如果您在 Mac 上工作，**OrbStack** 是目前公認最強大的 Minikube 替代品。它不是一個獨立的 K8s 工具，而是一個極輕量化的 Docker Desktop 替代方案，內建了 Kubernetes 支持。

- **技術核心：** 內建基於 **K3s** 的輕量化叢集。
- **優勢：**
  - **極速啟動：** 啟動 K8s 僅需約 2-5 秒。
  - **低資源消耗：** 閒置時幾乎不佔 CPU，且 RAM 是動態分配的。
  - **原生網路：** 支援直接透過 `.orb.local` 網域存取叢集內的 Service，不需要手動 `port-forward`。
- **適用場景：** 日常開發、快速測試 API 與微服務。

---

2. Kind (Kubernetes in Docker)

這是目前 DevOps 生態系中的標準工具，也是我們之前討論過、最適合模擬生產環境的工具。

- **技術核心：** 將每個 K8s 節點跑在一個 Docker 容器內。
- **優勢：**

    - **模擬多節點：** 透過一個 YAML 配置就能啟動 1 個 Master + N 個 Worker 節點。
    - **CI/CD 友好：** 因為底層是 Docker，它可以在 GitLab Runner 或 GitHub Actions 中輕鬆執行。
    - **Gateway API 測試：** 非常適合用來測試最新的 Gateway API 或 Service Mesh 配置。
- **適用場景：** 測試集群架構行為（如 Pod 親和性、節點故障轉移）。

---

3. K3d (K3s in Docker)

這是專門為 **K3s** 設計的封裝工具，類似於 Kind，但底層跑的是更輕量的 K3s。

- **優勢：** 繼承了 K3s 的快速與低資源特性，同時擁有 Kind 的多節點管理能力。
- **適用場景：** 需要模擬多節點但電腦資源（RAM）較為吃緊時。

---

4. 方案對照表

工具	啟動速度	資源佔用	模擬多節點	推薦等級 (2026)
OrbStack	極快	極低	較難 (內建單節點)	□□□□□ (Mac 首選)
Kind	中等	中等	**極易**	□□□□ (架構測試首選)
K3d	快	低	容易	□□□□ (輕量首選)
Minikube	慢	高 (需 VM)	普通	□□ (已逐漸式微)

專業建議：

如果您現在要建置本地環境：

1. **安裝 OrbStack：** 作為您的 Docker 與基礎 K8s 環境，解決 90% 的開發需求。
2. **搭配 Kind：** 當您需要測試複雜的負載平衡、多節點調度或跨節點網路時，在 OrbStack 上跑 Kind 叢集。

修訂版本 #1
★由 treeman 建立於 9 ⍰|⍰⍰ 2026 15:57:36
🖉由 treeman 更新於 9 ⍰|⍰⍰ 2026 16:17:51