

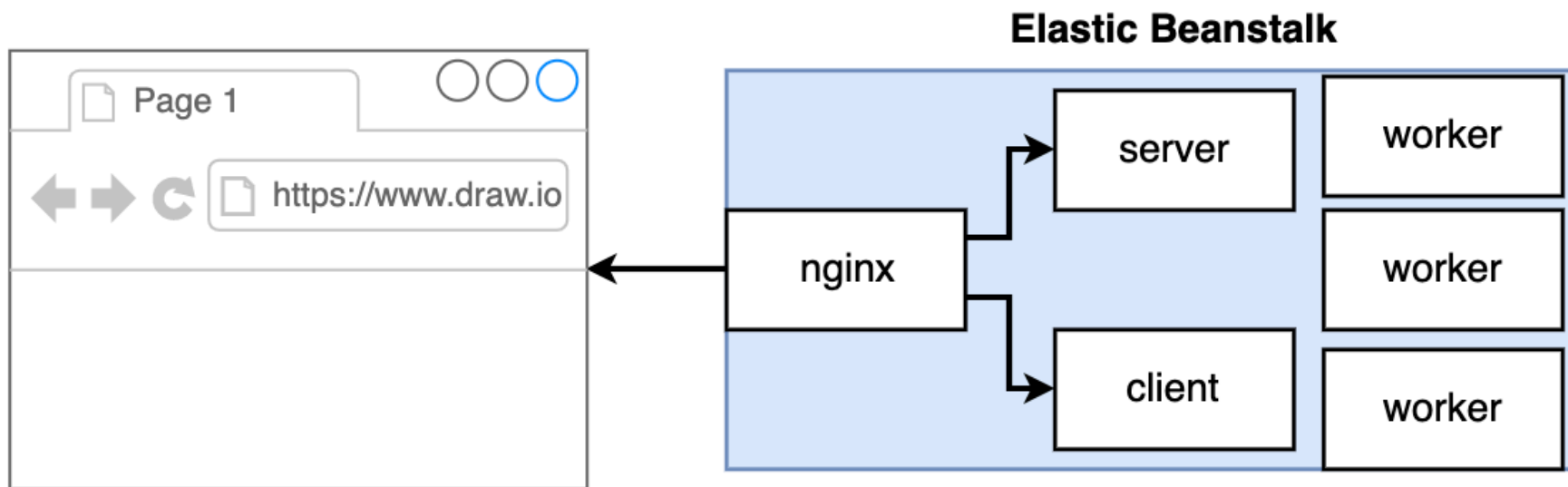
Docker & Kubernetes (3)

David Chiu

Kubernetes

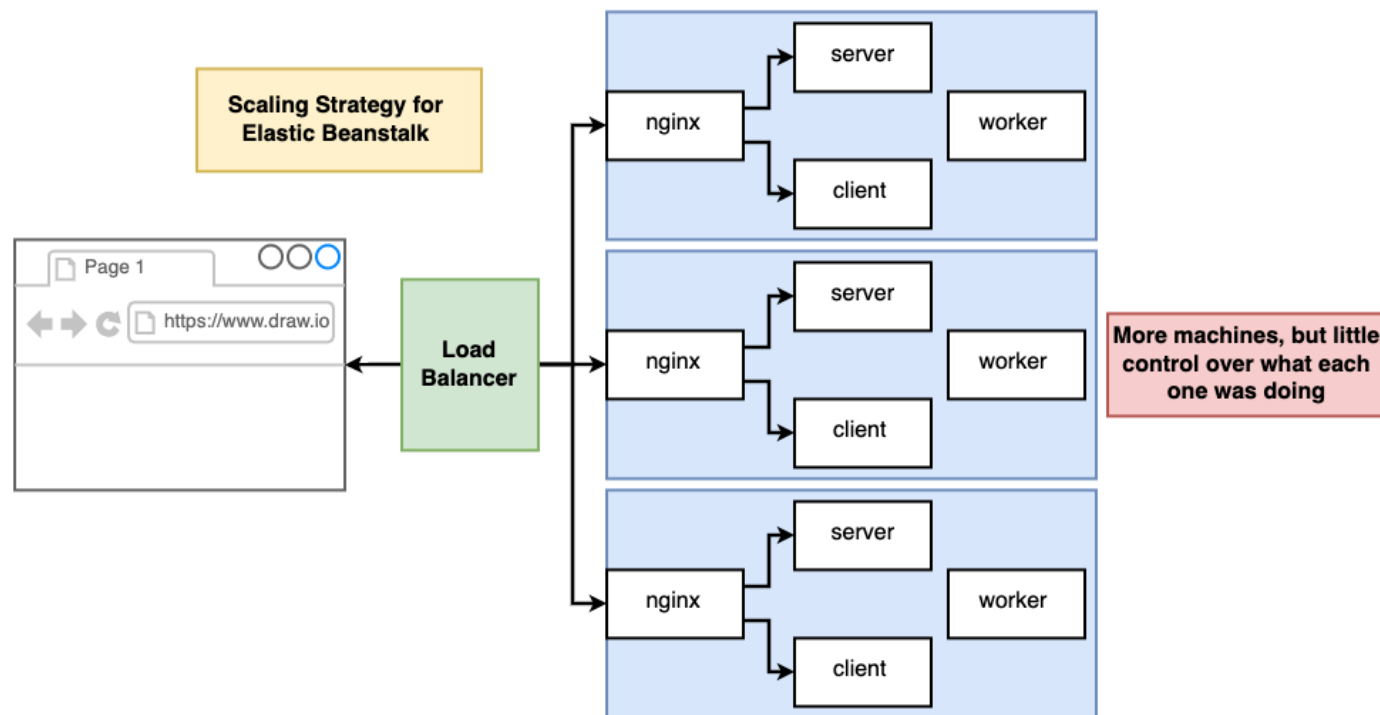
服務如何擴展？

■ 如何擴展既有服務？

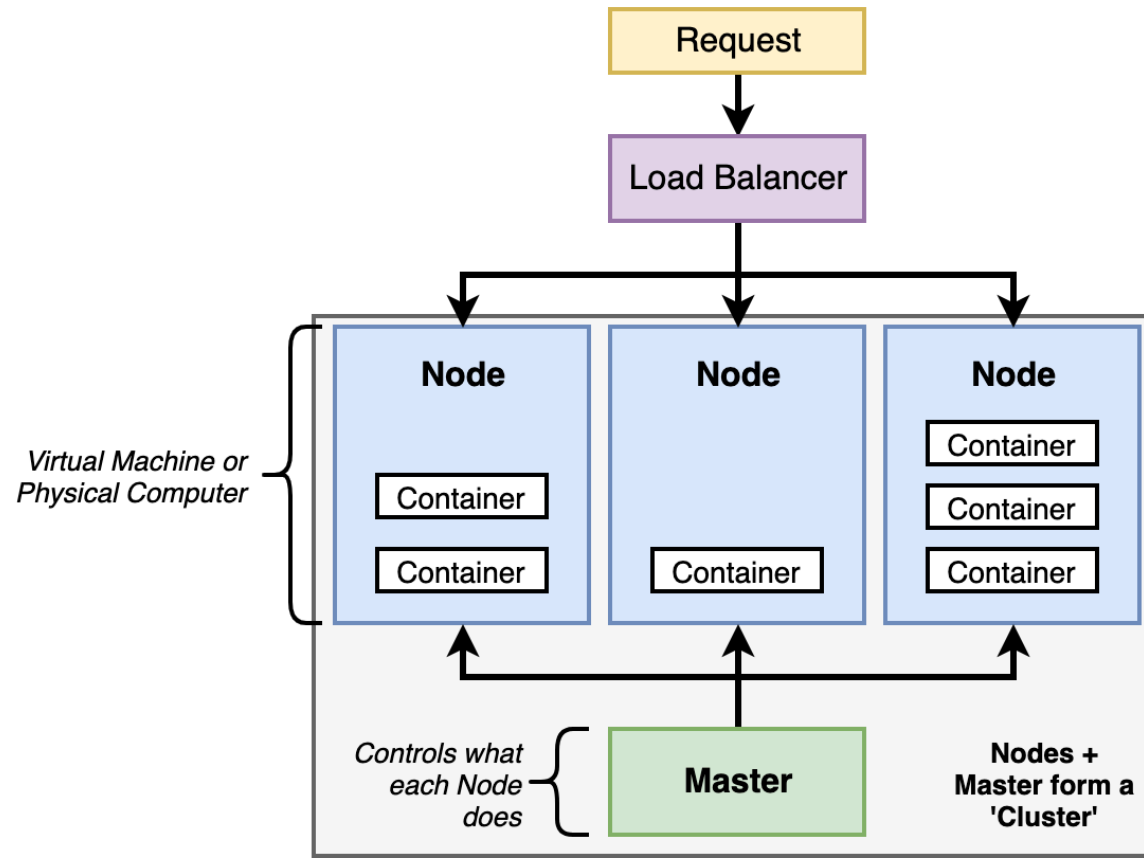


服務如何擴展？

■ 增加 VM，前端使用Load Balancer 管理流量



使用主從式架構管理機器



管理Container 延伸出的需求

- 當有需要管理多個機器(Node)上多個Container 時
- 當你需要使用多個Images 生成多個 Containers 時
- 希望可以透過一個Master 方便管理所有節點上的Containers

Kubernetes

- 「Kubernetes」一名取自於希臘原文，意即「掌舵手、船長」，延續Docker容器的海洋系列命名。而經常見到的「K8s」則是將“ubernete”八個字母縮寫為“8”，是現今資訊界常見的縮寫手法。
- Kubernetes是一個由Google開發並開源的系統，專門用以自動化部屬、彈性擴充及容器應用管理，而在最一開始便針對分散式叢集架構實現這些功能



Kubernetes 管理特性 (1)

■ 多租戶管理介面 (Multi-tenancy)

Kubernetes的多租戶管理以RBAC系統為基礎，管理者可以定義多個系統角色 (Cluster Role)，各自擁有不同的叢集權限，操作特定的命名空間 (namespace) 及資源 (Resources)

例如：管理者建立一命名空間「Dev」、系統角色「dev-user」、「dev-monitor」、系統帳戶「user」及「monitor」，綁定同名角色及帳戶，並生成相應定義檔 (kube-config)。

dev-user角色僅能在Dev命名空間部署、監控、管理容器應用，而非其他命名空間 (如default、kube-system等)；dev-monitor角色則僅能在Dev命名空間監控容器應用。根據管理者發給使用者的kube-config定義檔

■ 平台擴充性 (Cluster Scalability)

Kubernetes叢集中，以Node為主要運算節點。當使用者的資源需求增加 (CPU、RAM、GPU……)，而叢集運算資源不足時，維運人員能夠輕易地擴充運算節點。擴充的節點只要能安裝相應版本之Kubernetes元件即可，故無論是硬體規格或作業系統皆相當彈性。維運人員可以因應當下不足的資源，自由配置新的節點設備。

Kubernetes 管理特性 (2)

■ 高可用性管理介面 (Master HA)

無論是管理者或使用者，皆需要取得管理節點 (Master) 的核心服務 (API Server、Scheduler……) 才能對叢集進行服務部署及管理。而管理節點的核心元件中，Etcd尤為重要。Etcd保存了叢集中所有的網路配置及節點狀態訊息，而部分訊息是由複數個Etcd服務進行投票獲得，故Etcd服務數量 (即Master節點數量) 須為單數。

■ 多元儲存介面 (Storage Classes)

根據使用者的容器應用，不同應用所搭配的資料或檔案儲存系統，可能有各式各樣的偏好。Kubernetes支援多種儲存介面，供容器應用介接

■ 自定義叢集資源 (Custom Resource Definitions)

Kubernetes原生的叢集資源包括Pod、Deployment、DaemonSet、StatefulSet等等……。藉由Kubernetes的apiserver-builder，用戶可以自行定義簡單的叢集資源，用自定義的Controller模式部署容器應用

Kubernetes 應用特性 (1)

■ 滾動式升級 (Rolling Update)

Kubernetes自動以纍進方式更新容器，並且同時監控更新程序，避免在更新工作中一次性停擺所有舊版本容器，而造成服務中斷。若更新過程中發現錯誤訊息，Kubernetes也能夠自動回朔容器至更新前的穩定版本

■ 應用服務擴展性 (Service Scalability)

根據服務需求，管理人員能夠輕鬆設置該服務的自動擴展 (Auto Scaling)。在服務負載量超出正常預期時，Kubernetes將即時增加相應的容器應用，大幅減少維運人員面對臨時超載的工作負擔

■ 自動化負載平衡 (Service Discovery and Load Balancing)

在部署容器應用時，Kubernetes會發配給容器相應的IP位址，並藉Service叢集資源實現負載平衡的工作。使用者將無需自行設計負載平衡的功能架構與機制。

Kubernetes 應用特性 (2)

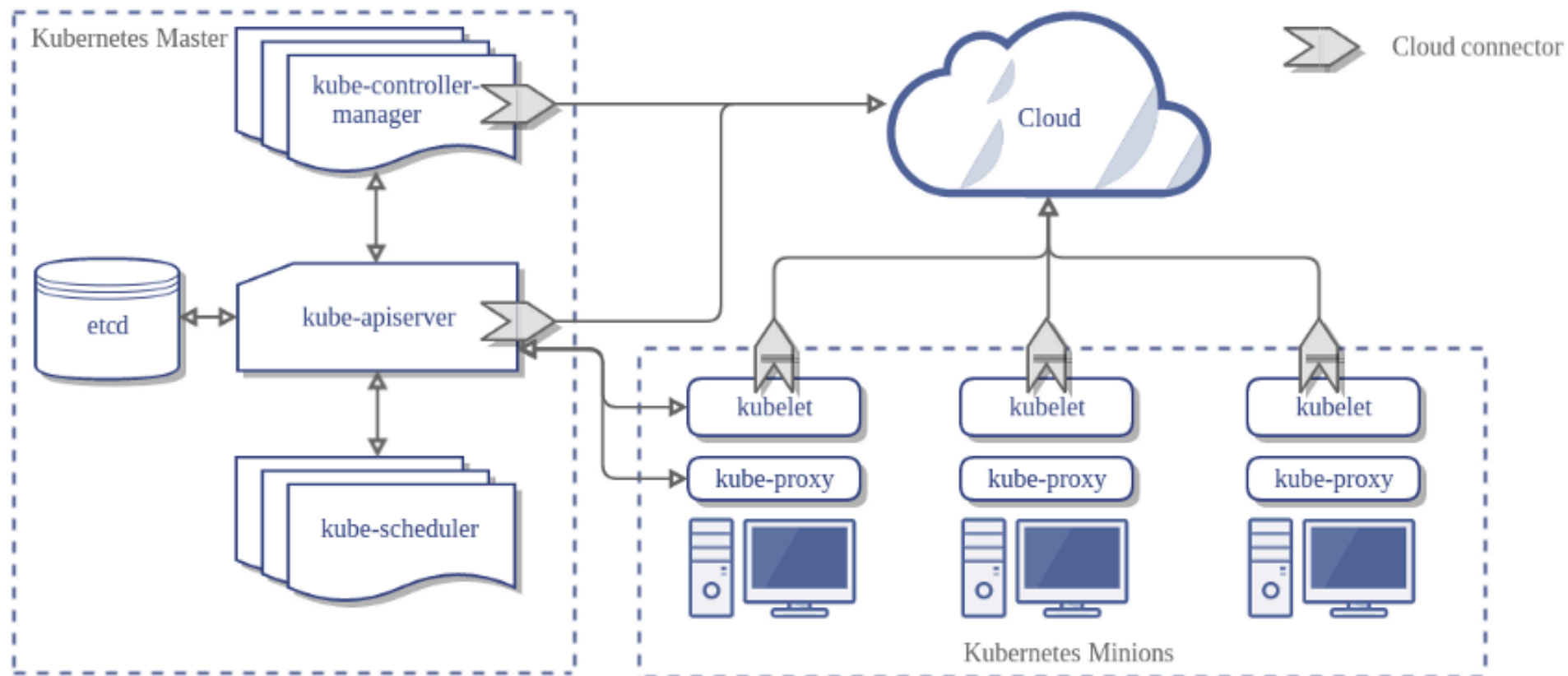
■ 自動重啟失能服務 (Self-healing)

當容器應用出現錯誤訊息並崩潰，Kubernetes能自主利用相同的容器映像檔及部署定義（YAML檔），排程該容器的重啟，以修復服務。若計算節點出錯當機，Kubernetes也能即時在其他計算節點重新部署容器應用，以恢復服務的提供

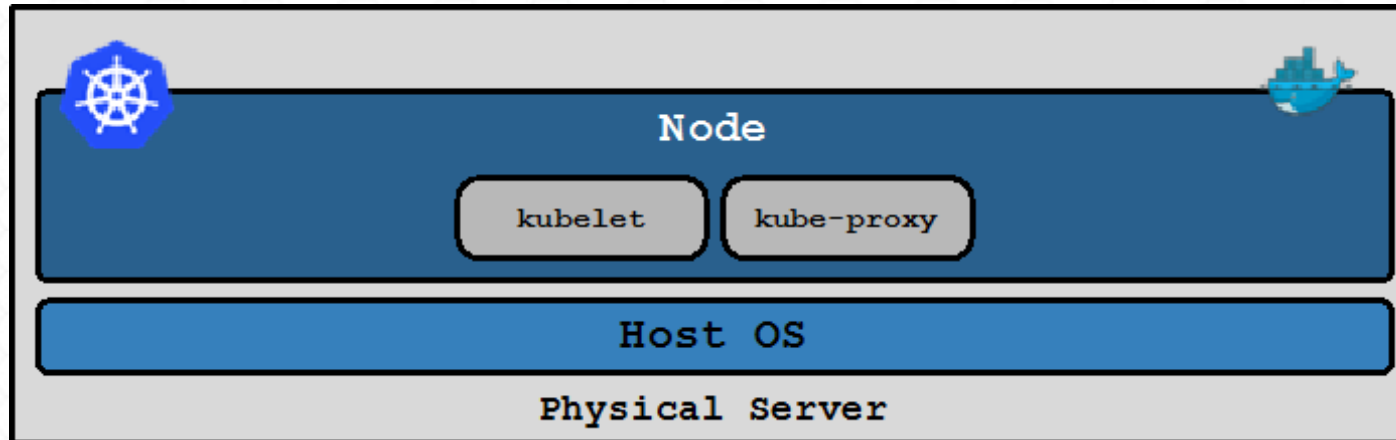
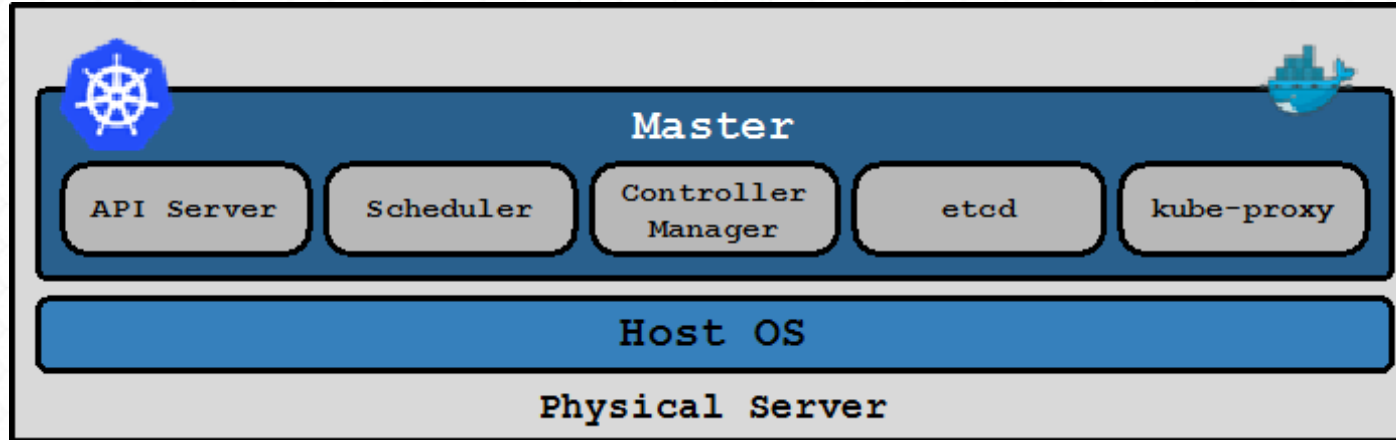
■ 容器網路管理 (Network Policy)

Kubernetes支援多種容器網路介面（CNI），並根據選擇的CNI方案，管理人員能夠自行設置網路管理規則，一方面進一步實現容器間相互溝通的管理，也提供網路安全的解決方案

Kubernetes 架構剖析



Master & Slave



Kubernetes Controller Manager

- **Kubernetes Controller Manager (KCM)** 位於 Kubernetes Master 中，負責所有的控制功能，其中包含了以下四個元件：
 - **Node Controller 節點控制器**：K8s Cluster 由許多 Node 所組成，Node Controller 負責管理這些 Node，包含新增刪除、Hostname 與 IP Address 管理、Node Type 與 Size 等等資源管理，當 Node 發生錯誤造成離線時，Node Controller 也會自動進行處理，將 Node 移除 Cluster。
 - **Route Controller 路由控制器**：這個 Route Controller 只有在 Google Compute Engine 才有作用，用來讓不同的雲節點可以互相通信。
 - **Service Controller 服務控制器**：在 K8s 中所有的網路是與外界隔離的，必須透過 Service 來註冊服務，主要是提供一個統一的入口，將網路封包派送到正確的 Container 中，也會扮演 Load Balance 的角色。
 - **PersistentVolumeLabels Controller**：由於所有 Container 在 K8s Cluster 都是 Stateless，但有時我們在實現服務的同時可能會需要儲存持久性的資料，像是資料庫等等服務，為了讓 Container 在任何一个 Node 重新執行的時候可以讀取之前的資料，所以必須提供 Volume 進行掛載。

Kubelet

- Kubelet 安裝於每一個 Node 上，負責與 API Server 溝通，也包含初始化並且將自己納入到整個 Cloud Cluster 的管理
- Kubelet 像是 Node 上面的 Docker 代理人，負責管理自己所分派的 Container

Kubernetes API server

- 所有的 K8s 操作都是透過 API Server，
- API 透過標準的 HTTP WebService 實現，包含了 Rest 與 WebSocket 等等 API 設計
- 無論是負責在節點管理 Container 的 kubelet 或開發者用來操作 K8s 的 kubectl，甚至 K8s Dashboard 都是透過這個 API Server 進行串接。

etcd

- etcd 是一個分散式資料庫系統，由於我們的 Master Node 可以由多個節點組成，好讓某個節點發生故障的時候可以有其他 Master Node 接手管理 Container，因次透過 etcd 會隨時同步每一個 Master Node 的資料

kube-scheduler

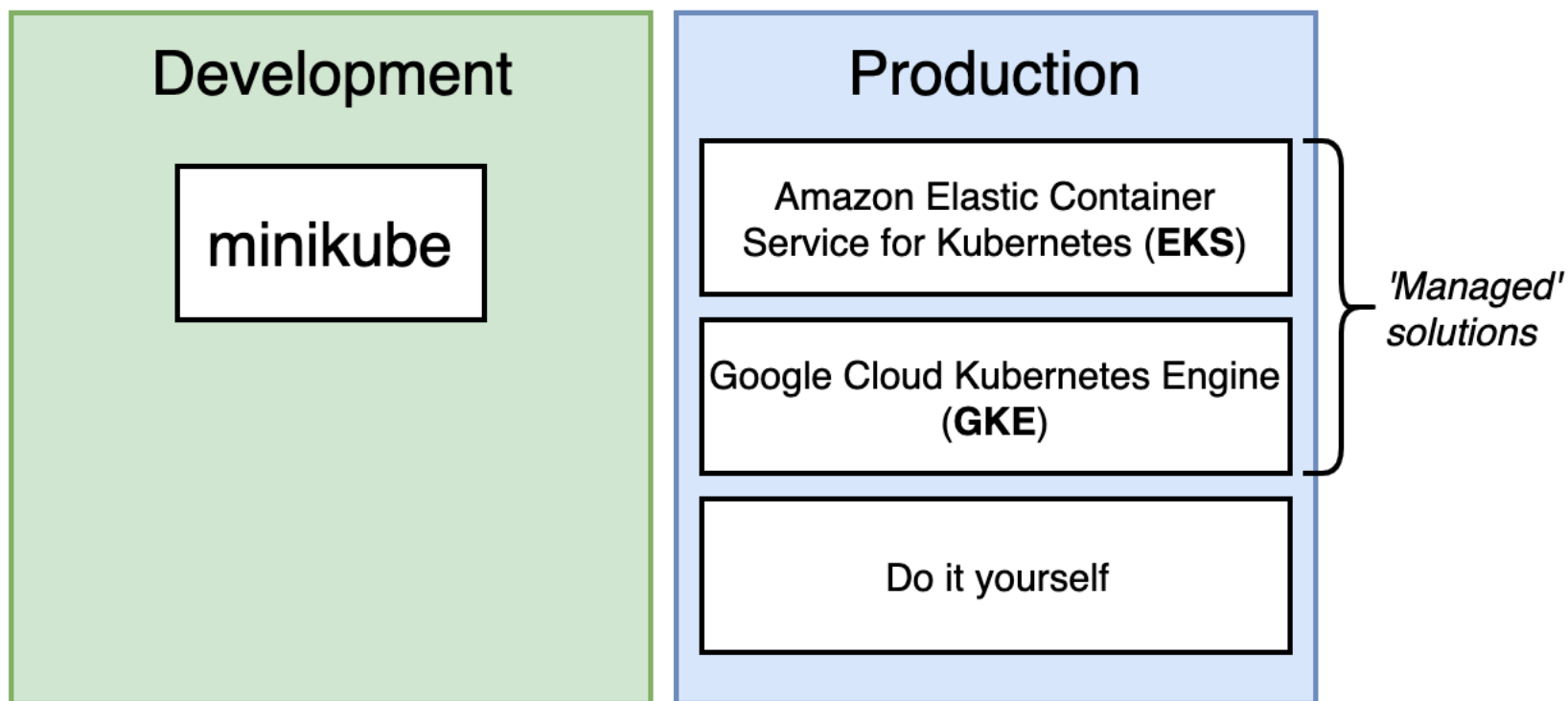
- K8s 所有 Container 的管理都是非同步的，當 API Server 收到資源管理需求時，會交由 kube-scheduler 進行調度，分配最適合的 Node 來執行 Container
- kube-scheduler 的工作包含
 - ▣ 在 Node 拓撲中選擇適合的 Node
 - ▣ kube-scheduler 還需要將 pod 進行重啟，其中有內建一個評分演算法，避免 pod 搞掛整台 Node

kube-proxy

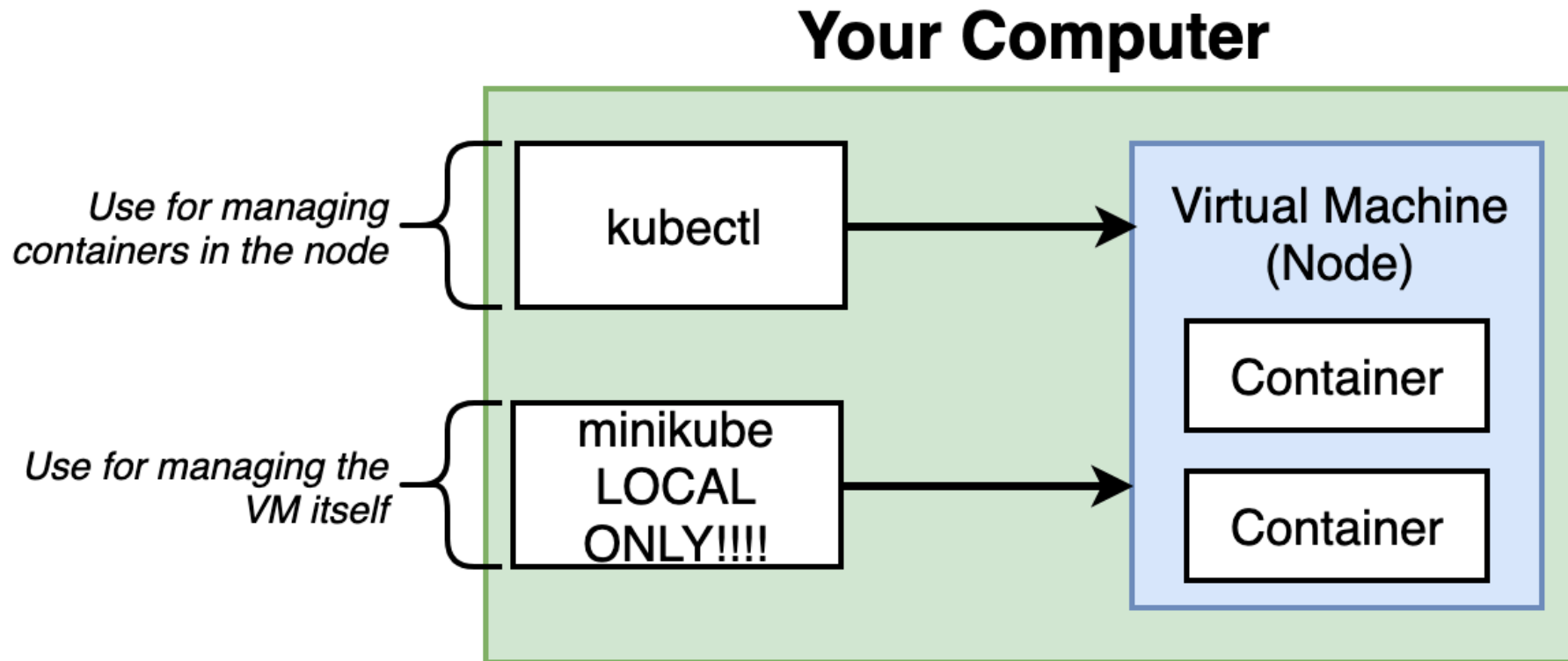
- kube-proxy 提供給 kubectl 或 kubelet 進行 API Server 連線，
kube-proxy 會讀取認證設定檔，方便同時管理不同的 K8s Cluster

使用 Kubernetes

Working with Kubernetes



Kubectl v.s. minikube



安裝 kubectl

■ 安裝步驟

- ❑ `curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl`
- ❑ `chmod +x ./kubectl`
- ❑ `sudo mv ./kubectl /usr/local/bin/kubectl`

安裝 minikube

■ 安裝步驟

- ❑ `curl -Lo minikube`

- `https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && chmod +x minikube`

- ❑ `sudo install minikube /usr/local/bin`

啟用 minikub

■ 啟用 minikub

□ `sudo minikube start --vm-driver=none`

需要至少兩顆 CPU

```
david@ctbc1:~$ sudo minikube start --vm-driver=none
🌟 minikube v1.5.2 on Ubuntu 18.04
👤 Running on localhost (CPUs=2, Memory=13014MB, Disk=49446MB) ...
🔑 OS release is Ubuntu 18.04.3 LTS
🐳 Preparing Kubernetes v1.16.2 on Docker '18.09.7' ...
   ▪ kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
📁 Downloading kubeadm v1.16.2
📁 Downloading kubelet v1.16.2
🚚 Pulling images ...
🚀 Launching Kubernetes ...
👤 Configuring local host environment ...
```

檢視minikub 狀態

■ sudo minikube status

```
host: Running  
kubelet: Running  
apiserver: Running  
kubeconfig: Configured
```

Kubernetes Playground

■ <https://www.katacoda.com/courses/kubernetes/playground>

Welcome!

Kubernetes Playground

★ **Difficulty:** Beginner

🕒 **Estimated Time:** 10 minutes

This is a Kubernetes playground, a safe place designed for experimenting, exploring and learning Kubernetes. The playground has a pre-configured Kubernetes cluster with two nodes, one configured as the master node and a second worker node.


What are playgrounds?

Playgrounds give you a configured environment to start playing and exploring using an unstructured learning approach. Playgrounds are great for experimenting and trying samples.

To learn more about different Cloud Native technologies start with one of our [labs](#).

START SCENARIO

Playground 用作指令測試使用

 **Katacoda**

Katacoda Overview & Solutions Search Log In Sign Up

Kubernetes Playground

Launch Cluster

```
launch.sh ↵
```

This will create a two node Kubernetes cluster using WeaveNet for networking.

Health Check

```
kubectl cluster-info ↵
```

Interested in writing your own Kubernetes scenarios and demos? Visit www.katacoda.com/teach

CONTINUE

Terminal Host 1 + 🔍 ⌵ ⚙

Your Interactive Bash Terminal.

master \$

Terminal Host 2

Your Interactive Bash Terminal.

node01 \$

Kubernetes 教學網站

- <https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-interactive/>

The screenshot shows the Kubernetes documentation website's interactive tutorial for creating a cluster. The page has a blue header with the Kubernetes logo and navigation links: Documentation, Blog, Partners, Community, Case Studies, English, and v1.16. The main content area is titled 'Module 1 - Create a Kubernetes cluster' and 'Step 1 of 3'. The section is 'Cluster up and running'. The text explains that minikube is already installed and instructs the user to verify it by running 'minikube version'. Below this, it says 'OK, we can see that minikube is in place.' and instructs the user to start the cluster by running 'minikube start'. A terminal window on the right shows the command 'minikube ver' being entered.

Module 1 - Create a Kubernetes cluster

Step 1 of 3 ▶

Cluster up and running

We already installed minikube for you. Check that it is properly installed, by running the *minikube version* command:

```
minikube version ↵
```

OK, we can see that minikube is in place.

Start the cluster, by running the *minikube start* command:

```
minikube start ↵
```

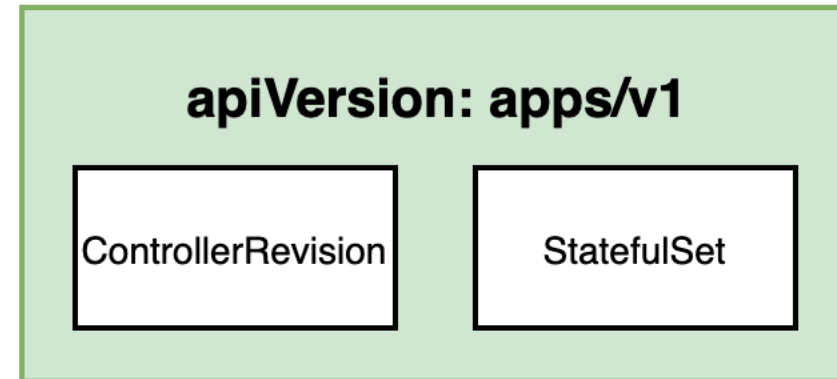
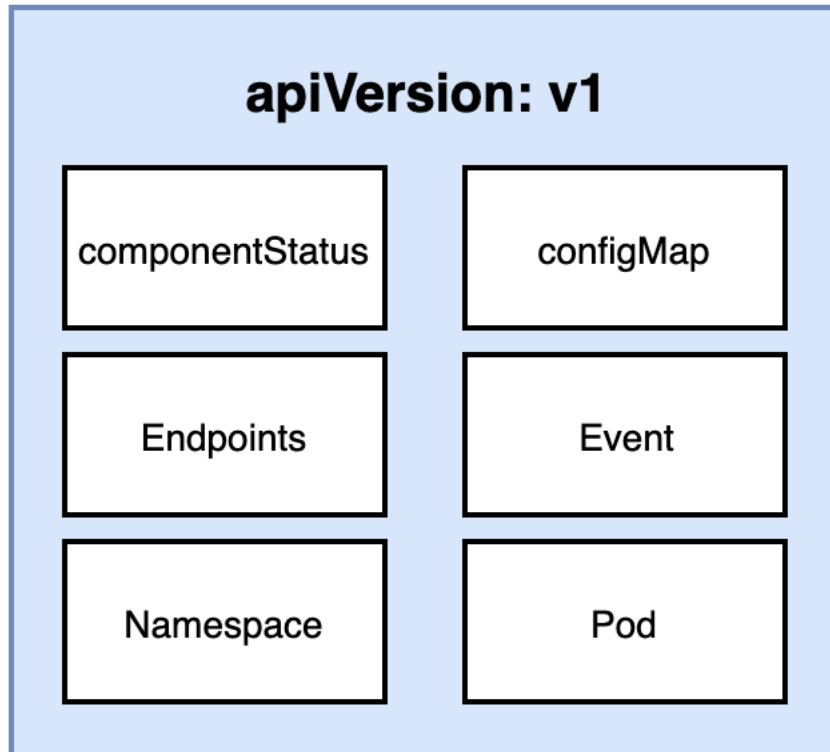
Great! You now have a running Kubernetes cluster in your online terminal. Minikube started a virtual machine for you, and a

Terminal +

Kubernetes Bootcamp Terminal

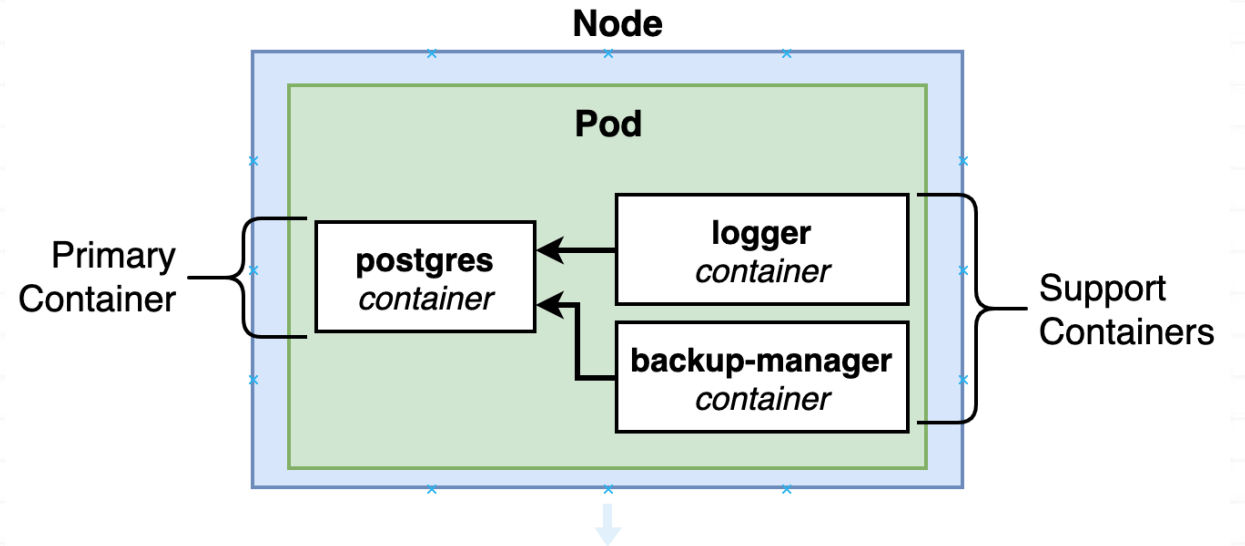
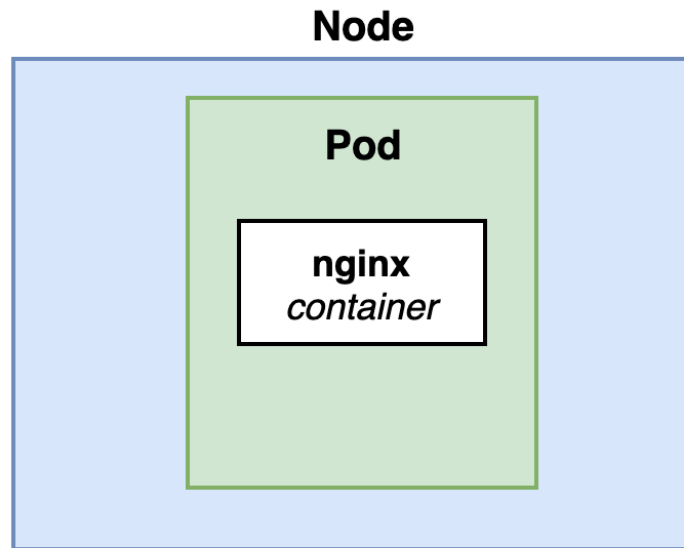
```
$  
$ minikube ver
```

API

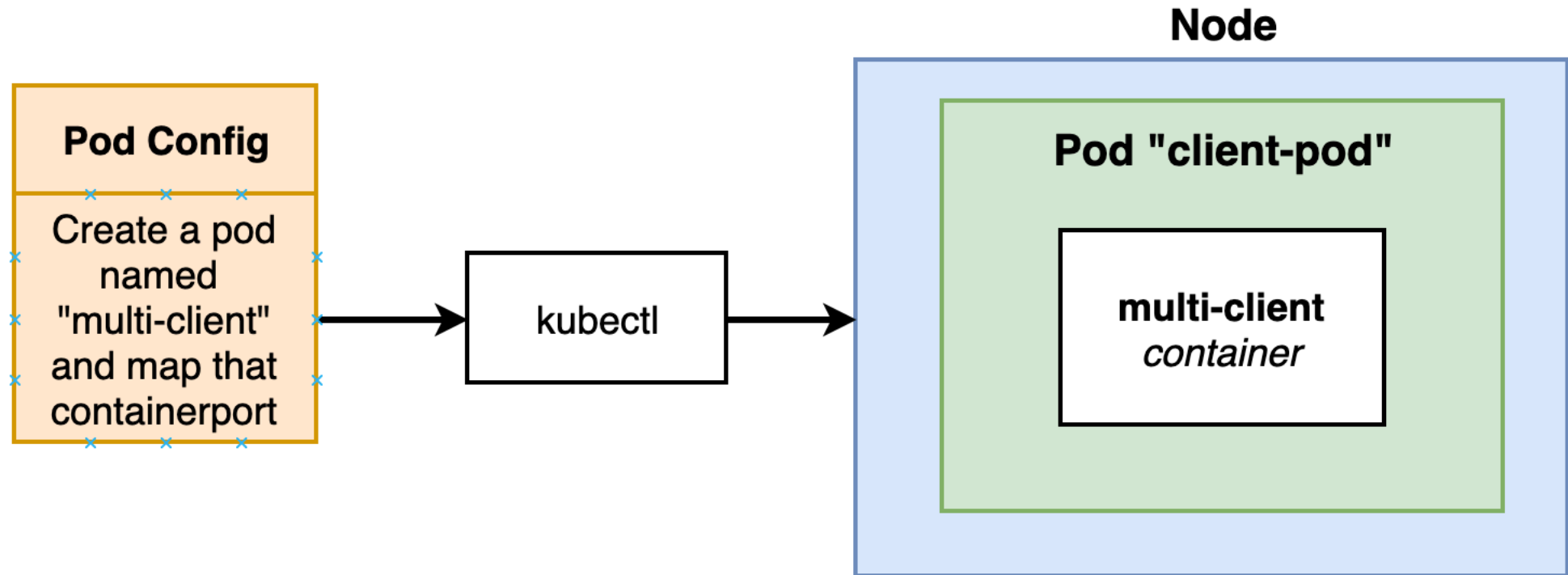


Each API version defines a different set of 'objects' we can use

POD



Enable Pod



The background features a light blue hexagonal grid pattern. Overlaid on this are several concentric, semi-transparent circles in shades of blue and white. The circles have a slightly irregular, hand-drawn appearance. A solid dark blue horizontal line runs across the top of the image, and a similar, slightly textured dark blue line runs across the bottom.

THANK YOU