

Go 程式設計課 08

Go 與 K8s



大綱

- ❖ 基於 docker desktop 的 k8s 測試集群
- ❖ 微服務概念
- ❖ 進階容器概念
- ❖ Docker Compose
- ❖ K8s 概念
- ❖ GKE 舉例
- ❖ K8s 練習題
- ❖ 用 K8s 發布後端



基於 docker desktop 的 k8s 測試集群

進入設定介面選擇啟用

<https://docs.docker.com/desktop/kubernetes/>

1. From the Docker Dashboard, select the Settings.
2. Select Kubernetes from the left sidebar.
3. Next to Enable Kubernetes, select the checkbox.
4. Select Apply & Restart to save the settings and then select Install to confirm.
5. note: 自動安裝 `/usr/local/bin/kubectl` 不支援 Linux

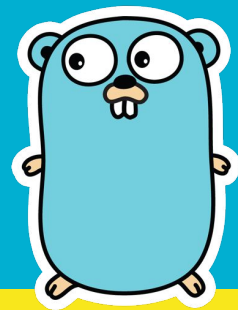


CLI 操作指令

Docker Desktop integration provides the Kubernetes CLI command

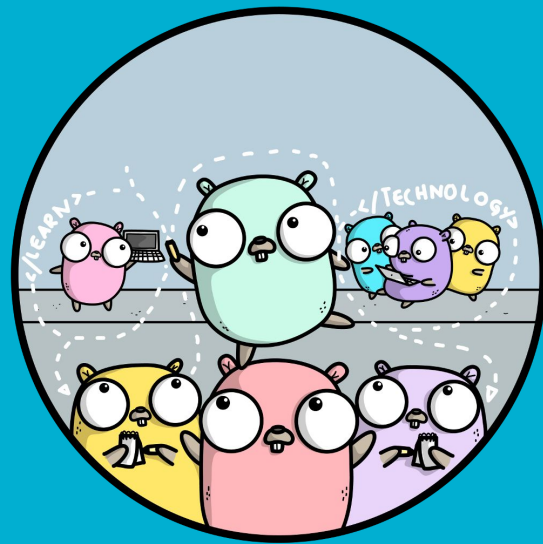
- Mac: `/usr/local/bin/kubectl`
- Windows: `C:\Program Files\Docker\Docker\Resources\bin\kubectl.exe`

範例指令：``/usr/local/bin/kubectl get nodes``



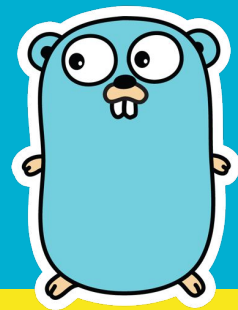
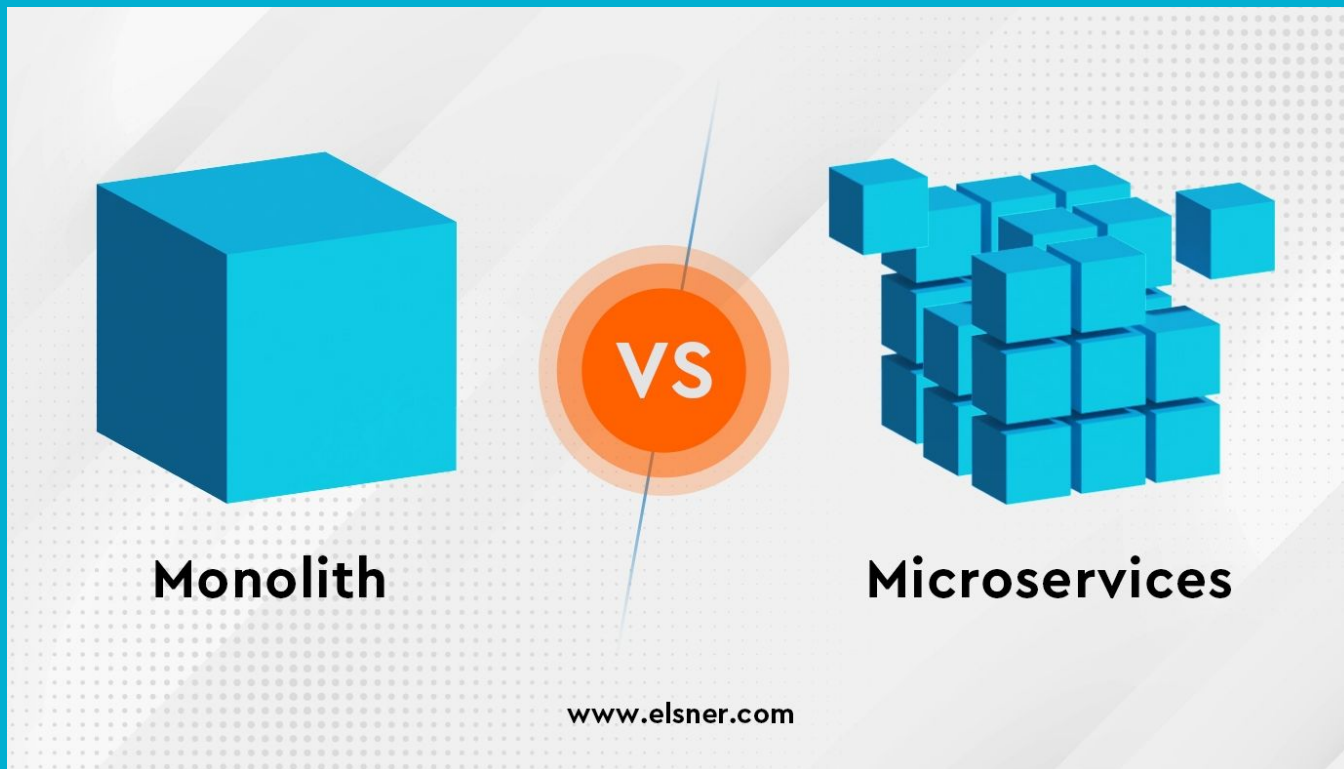
小試身手

—
安裝看看 K8s 測試群集吧！



微服務概念

單體 vs 微服務 (1)



單體 vs 微服務 (2)

單體架構: 將所有功能打包到一個單獨的應用程序中

- 優點: 簡單、易於開發和部署
- 缺點: 難以維護、擴展和更新,一個組件的問題可能導致整個應用程序失敗

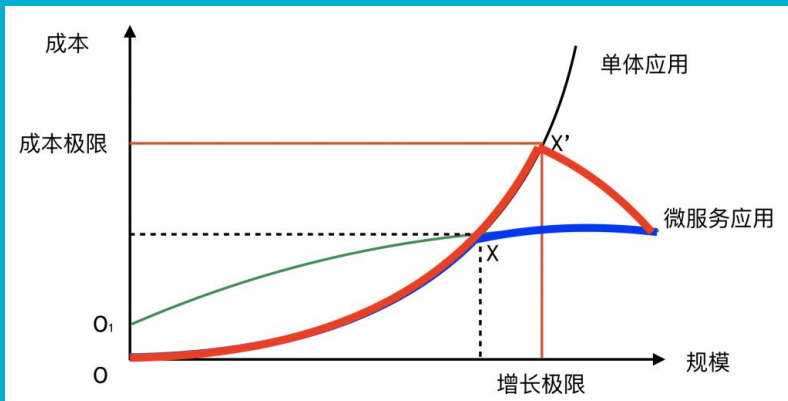
微服務架構: 將應用程序拆分為多個小型服務

- 優點: 每個服務可以獨立開發、部署和擴展,提高了靈活性和可維護性
- 缺點: 增加了複雜性,需要處理服務間的通信、數據一致性等問題



微服務的優點

- 解耦: 服務之間鬆散耦合, 可以獨立開發、部署和擴展
- 可擴展性: 可以針對個別服務進行水平擴展, 提高資源利用效率
- 容錯性: 一個服務的故障不會導致整個應用程序失敗
- 技術多樣性: 每個服務可以選擇最適合的技術棧, 不受其他服務的限制
- 敏捷開發: 小型服務更容易理解和修改, 加速了開發和迭代速度



圖片 ref:

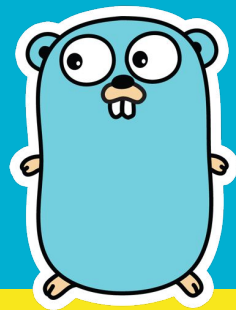
<https://cloud.tencent.com/developer/article/1371055>



微服務的挑戰

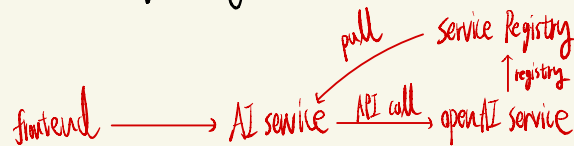
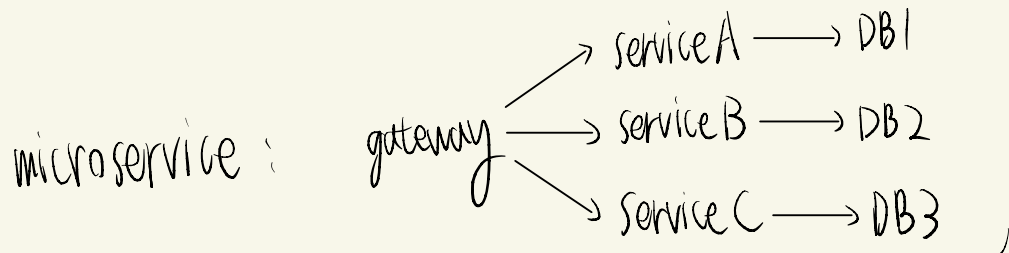
- 複雜性: 服務之間的依賴關係和通信增加了系統的複雜性
- 數據一致性: 每個服務可能有自己的數據存儲,需要處理數據一致性問題
- 測試: 需要對每個服務進行獨立的測試,並進行端到端的集成測試
- 部署: 需要協調多個服務的部署,並處理服務之間的版本相容性
- 監控: 需要對每個服務進行獨立的監控和錯誤跟蹤,並進行全局的性能分析

用 K8s 管理微服務系統可以盡可能克服以上挑戰！



微服務

<https://martinfowler.com/articles/microservices.html>

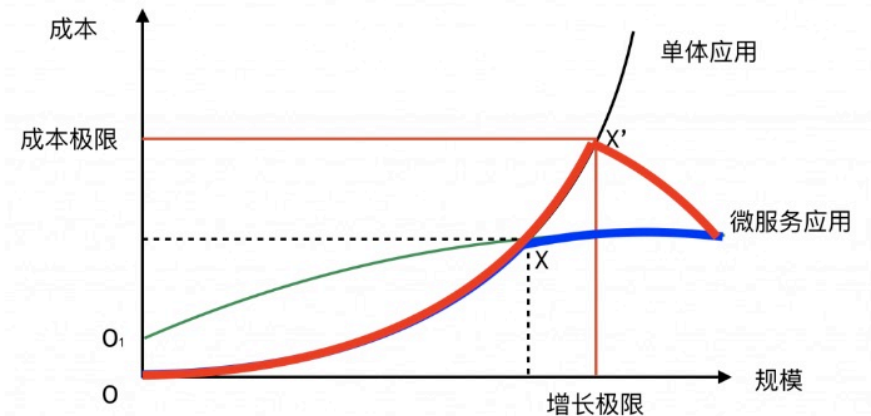


1. 透過服務進行組件化 易分割, 易抽換
2. 圍繞業務能力組織 用 business 拆部門
3. 產品而非項目 產品可以重用
4. endpoints and message queue 易串接
event driven

5. 去中心化治理 服務發現
6. 分散資料管理 1 service 1 DB
7. 基礎設施自動化 ★
↓ microservice 的 cost
8. 為失敗而設計
1. 容忍錯誤
2. log / monitor

微服務的成本

痛點: 跨 *communication* services 的 *data join*
(服務間是 *transaction less*)



圖片 ref: <https://cloud.tencent.com/developer/article/1371055>

總之規模不夠大, 微服務的成本反而會是劣勢

<https://cloud.tencent.com/developer/article/1371055>



1



微服務、敏捷開發成本概念

微服務

小MIS 2024-04-13 09:47:58 · 1407 瀏覽

分享至

我覺得

微服務不是降低成本

而是用時間成本來換取服務的解耦

成本是大增加!

但是網上有很多風氣是降低成本

回答 1 討論 2 邀請回答 追蹤 0

檢舉



浩瀚星空 iT邦大神 1 級 · 2024-04-13 20:04:16

檢舉

你會說這樣的話, 代表你還不懂如何利用。
所以才會有成本增加的感覺。
但你不用擔心, 很多人都是這樣認為。
畢竟不懂, 無法知道如何好好的運用其服務來達到最低成本。
還是守著舊有的觀念使用其服務。

那一定會更貴的。



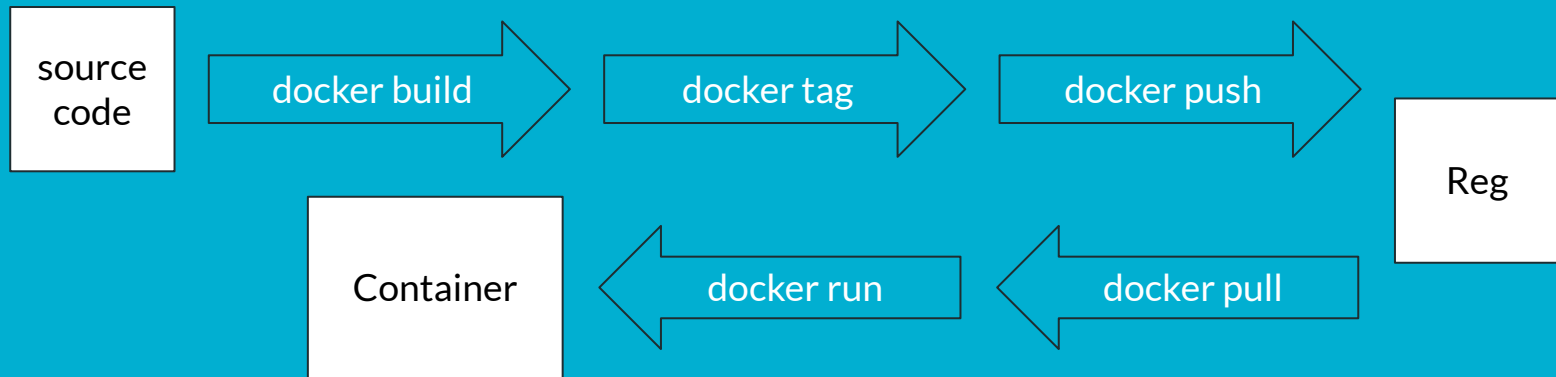
froce iT邦大師 1 級 · 2024-04-13 20:40:00

做的規模不夠大, 規模大的話分拆和合作才會重要

進階容器概念

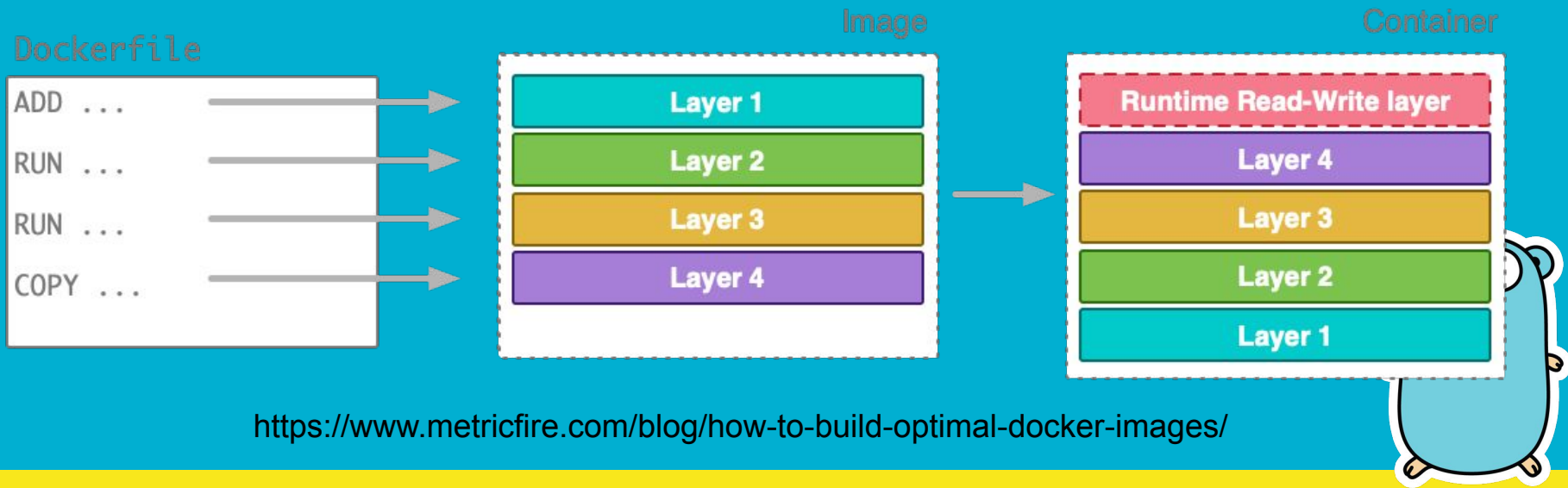
一個容器的生命

- ❖ Build: 創建 Image , 靜態的容器, 可以視為映像
- ❖ Ship: 上傳映像 (image) 至 Registry
- ❖ Run: 利用 Registry 網址下載 image 後執行, 成為 container



容器的存儲機制 Copy-on-Write

- ❖ 一層層的存儲
- ❖ 共用鏡像層, 各自使用容器層
- ❖ 基礎容器亦可在多個容器間共用



Docker Compose

Docker Compose

目的: 用宣告完成複雜的多組 docker 指令

簡介

- Docker Compose 是一個用於定義和運行多容器 Docker 應用程序的工具
- 使用 YAML 文件來配置應用程序的服務
- 通過一個命令就可以從配置中創建和啟動所有服務

主要概念

- 服務(Service): 應用程序中的一個容器, 可以包括多個運行相同鏡像的容器實例
- 項目(Project): 由一組關聯的應用容器組成的一個完整業務單元
- 網絡(Network): Docker 創建的一個虛擬網絡, 用於連接多個容器
- 卷(Volume): Docker 創建的一個虛擬數據卷, 用於持久化數據

跳到 Lab 有練習題!



K8s 概念

what is K8s ?

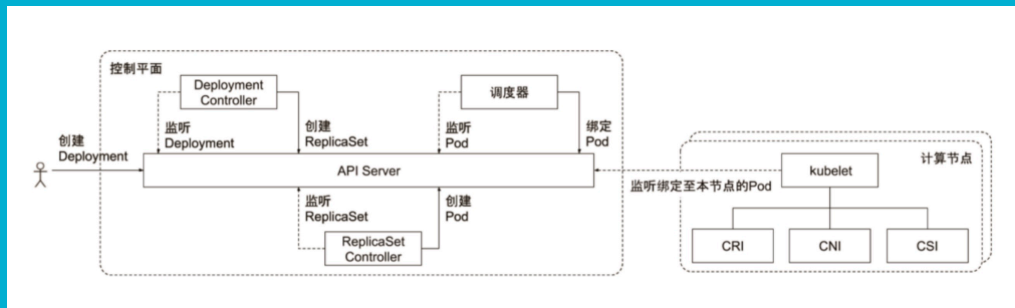
- Kubernetes(K8s)是一個開源的容器編排平台,由Google開發並捐贈給Cloud Native Computing Foundation(CNCF)
- 用於自動化部署、擴展和管理容器化應用程序
- 提供了一個可移植、可擴展和自我修復的平台,使應用程序能夠跨多個主機和雲環境運行



kubernetes



Kubernetes的架構 (1)

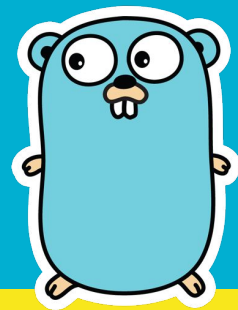


- Master節點:

- API Server: 提供了Kubernetes API,是與Kubernetes交互的主要方式
- etcd: 一個分佈式的鍵值存儲,用於存儲Kubernetes的配置數據
- Scheduler: 負責將Pod調度到合適的Node節點上運行
- Controller Manager: 運行各種控制器,如Replication Controller和Deployment Controller

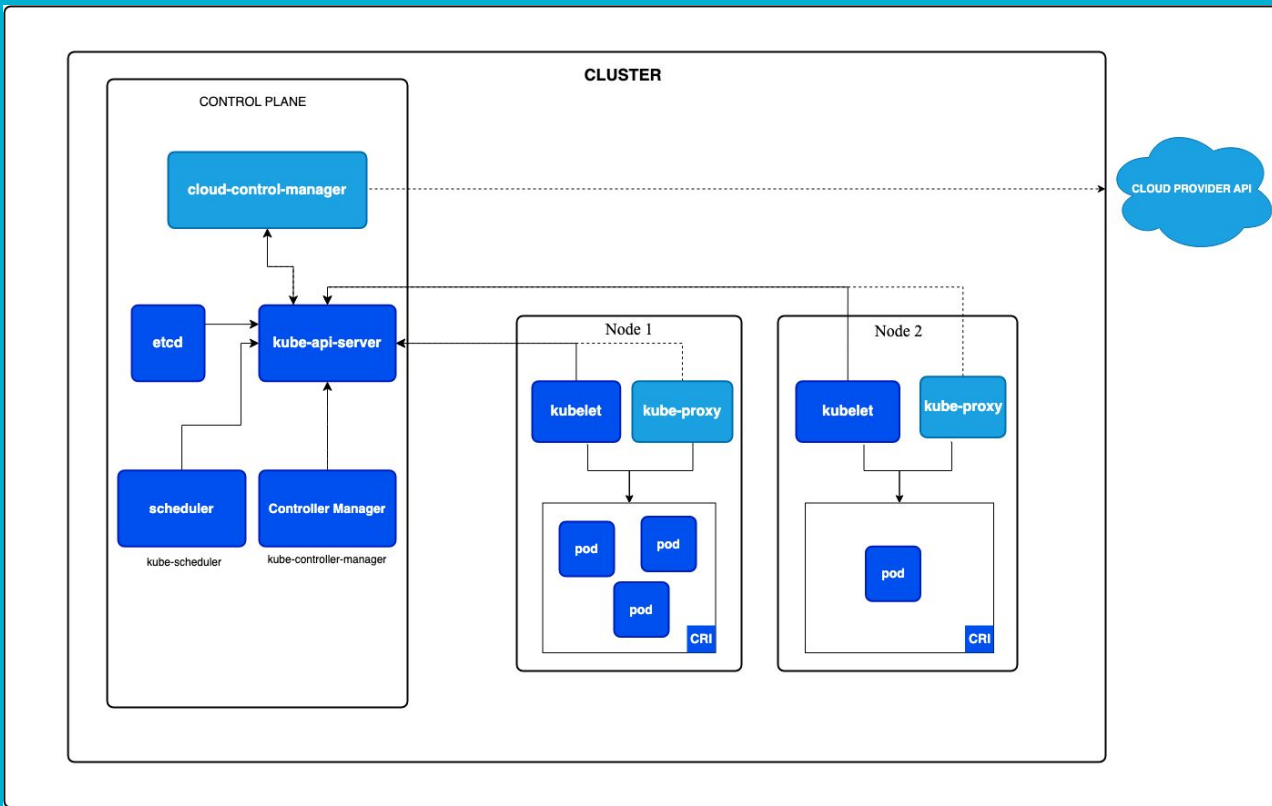
- Worker 節點:

- Kubelet: 運行在每個Node節點上的代理,負責與API Server通信並管理Pod的生命週期
- Container Runtime: 運行容器的軟件,如Docker或containerd
- Kube-proxy: 負責網絡代理和負載平衡,將服務請求轉發到相應的Pod



Kubernetes的架構 (2)

ref:<https://kubernetes.io/docs/concepts/architecture/>

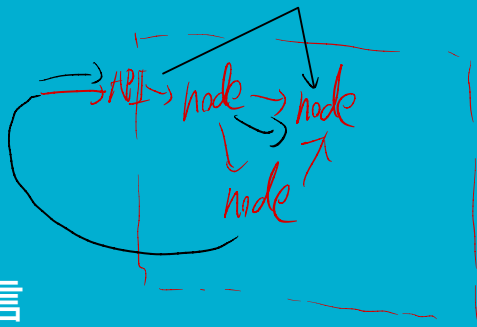


Kubernetes的常見組件

- Pod: 最小的部署單元,包含一個或多個容器,共享網絡和存儲資源
- Service: 定義了一組Pod的訪問策略,提供了一個穩定的IP地址和DNS名稱
- Deployment: 描述了一個應用程序的期望狀態,並管理Pod的更新和擴展
- ReplicaSet: 確保指定數量的Pod副本在運行,提供了故障恢復和擴展的能力
- ConfigMap: 存儲應用程序的配置數據,可以在Pod中以環境變量或文件的形式使用
- Secret: 存儲敏感數據,如密碼和密鑰,並以安全的方式提供給Pod使用
- PersistentVolume: 表示一個存儲資源,可以被Pod掛載和使用
- PersistentVolumeClaim: 表示一個對存儲資源的請求,由Pod使用

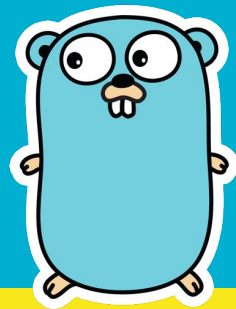


Kubernetes的網絡模型



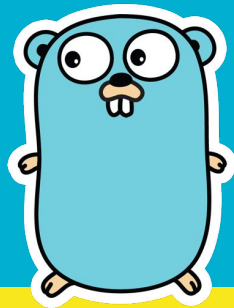
- 每個Pod都有自己的IP地址,可以直接與其他Pod通信
- 同一個Pod內的容器共享網絡命名空間,可以使用localhost通信
- Service提供了一個固定的IP地址和DNS名稱,可以負載平衡到多個Pod
- Ingress提供了外部訪問Service的入口,支持HTTP和HTTPS路由
- 網絡策略(Network Policy)可以控制Pod之間的網絡訪問,提供了細粒度的安全控制

network 實際案例: internal pod 採用外部通信導致 ingress 堵塞



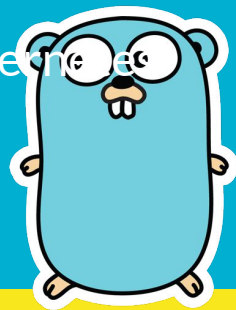
Kubernetes的其它功能

- 水平Pod自動擴展(Horizontal Pod Autoscaler, HPA): 根據CPU使用率或自定義指標自動調整Pod的數量
- 滾動更新(Rolling Update): 逐步替換舊的Pod,實現不中斷的應用程序更新
- DaemonSet: 確保在每個Node節點上運行一個Pod,用於運行系統級服務
- StatefulSet: 用於管理有狀態的應用程序,如數據庫,提供了穩定的網絡標識和持久存儲
- Job和CronJob: 用於運行一次性或定時的任務,如數據備份或報告生成
- K8s 幾乎可以做到任何你想做的事!



Kubernetes的優勢

- 跨平台的可移植性: 可以在各種基礎設施上運行,如本地數據中心、公有雲和混合雲, 避免在單一雲上產生 vendor lock。
- 自動化的擴展和恢復: 根據負載自動調整應用程序的規模,並在故障時自動重啟或重新調度Pod
- 聲明式的配置管理: 使用YAML或JSON文件描述應用程序的期望狀態,Kubernetes負責實現和維護該狀態,長期穩定運維更方便。
- 豐富的生態系統: 有大量的工具和擴展可以與Kubernetes集成,如監控、日誌、CI/CD等
- 強大的社區支持: 有活躍的開發者社區和豐富的文檔資源,不斷推動Kubernetes的發展和創新



Kubernetes的本質

- 最早被 Google 實作與定義
- 後被CNCF(Cloud Native Computing Foundation)定義了一系列組件標準
 - OCI(Open Container Initiative) : 容器 runtime 的標準, 例如: docker, podman,
 - CNI(Container Network Interface): 容器網路介面標準, 允許所有容器對內外的通信共通
 - CSI(Container Storage Interface): 定義了容器存儲接口的標準規範, 允許容器取用存儲
- 延伸標準
 - Prometheus: 一種開放的監控和警報標準, 可以通過 Prometheus 獲取和分析集群的性能指標
 - OpenTracing: 一種分佈式追蹤的標準規範, 定義了一種通用的API 和數據模型,用於跟蹤跨服務的請求流量
 - 還有更多

每個廠商和用戶, 都可以自行組合自己的 K8s 實作, 下一章節介紹 google cloud 的 K8s 實作



K8s 練習題

跳到 Lab 用 K8s 發布應用！



GKE 舉例

除了基本的 K8s 特性外追加功能

1. 全託管服務
2. 自動修復
3. 自動更新
4. 優化的 runtime OS
5. 特殊的 Cluster 級別擴展
6. Google Cloud 整合
 - a. Log
 - b. Monitor
 - c. LB
 - d. WAF/ IPS/ IDS
 - e. GPU
 - f.還很多



客製化 K8s 模塊實現特殊組件

以下舉例: Kueue

```
apiVersion: kueue.x-k8s.io/v1beta1
kind: ResourceFlavor
metadata:
  name: default-flavor # This ResourceFlavor will be used for all the resources
```

透過 apiVersion 引用特殊插件, 實踐特殊功能

ex: 在 K8s 內基於 message queue 完成大量併發排程任務



Lab

設計模式介紹與實戰



K8s 練習題

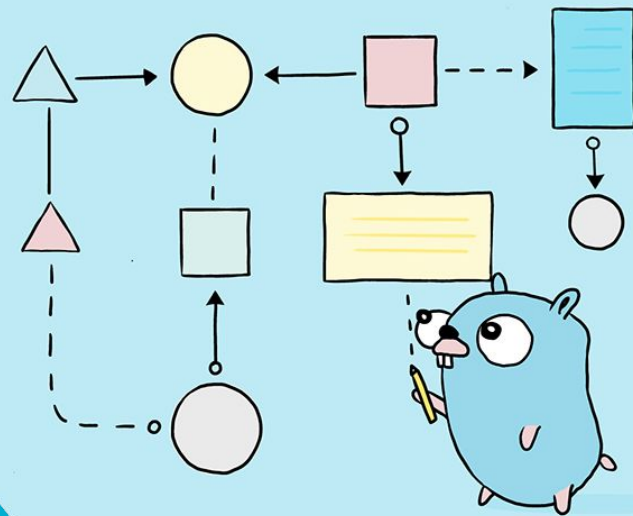
1. 用 Docker Compose 發布應用
2. 用 K8s 發布應用

Link: <https://github.com/leon123858/go-tutorial/blob/main/k8s/Makefile>

補充：用 Azure 發布



Q&A 時間



作業: 後端用 k8s 發布

Link: <https://github.com/leon123858/go-tutorial/blob/main/short-url/k8s.yaml>

