



X



Google Kubernetes Engine

WordPress在GKE上的應用

10.10.2022

—

Close Su



概述

WordPress是最受歡迎的網站內容管理系統。

Kubernetes是Google設計出來用於自動化部署、擴展與管理容器化應用程式的開源系統。

本次專題結合雲端化與容器化的趨勢，將WordPress應用佈署在GKE(Google kubernetes Engine)上，並且串接GCP上其他雲端服務。

目標

1. 透過專題來了解k8s的各種服務應用方式，並成功佈署與串接gcp的其他雲端服務
2. 將完整的部署流程做成技術輸出文件提供給他人使用，成為巨人的肩膀

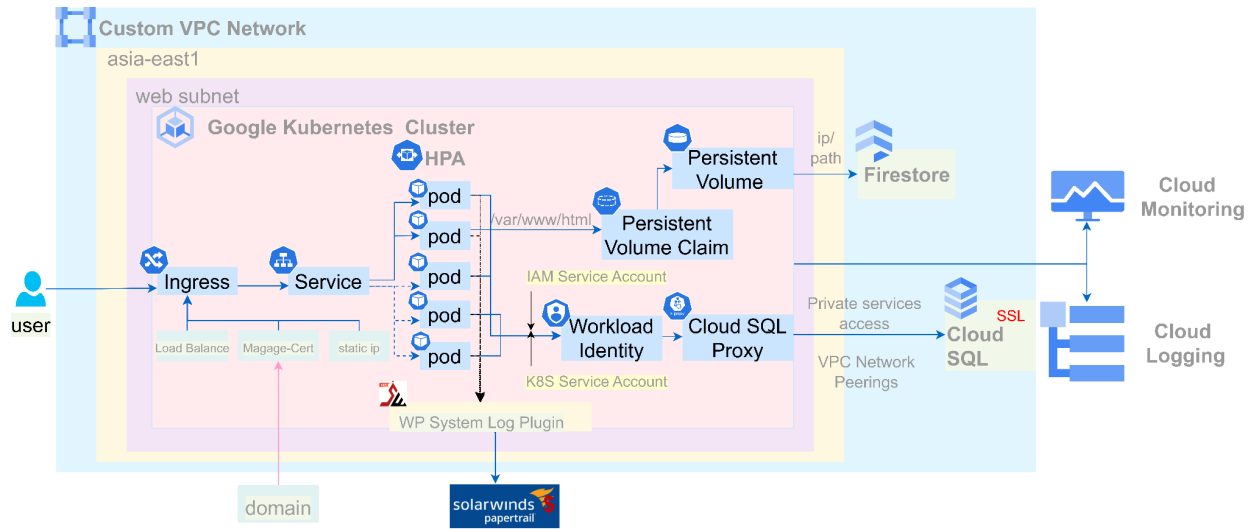
目錄

概述	1
目標	1
目錄	1
總體架構圖	4
K8S架構圖	4
建立VPC以及GKE集群	5
原理	5
VPC:	5
GKE:	6
操作流程	7
注意事項:	7
創建VPC:	7
建立80 port的防火牆規則:	10
建立GKE集群:	13
建立Cloud SQL Proxy	16
原理	16
Cloud SQL Auth:	16

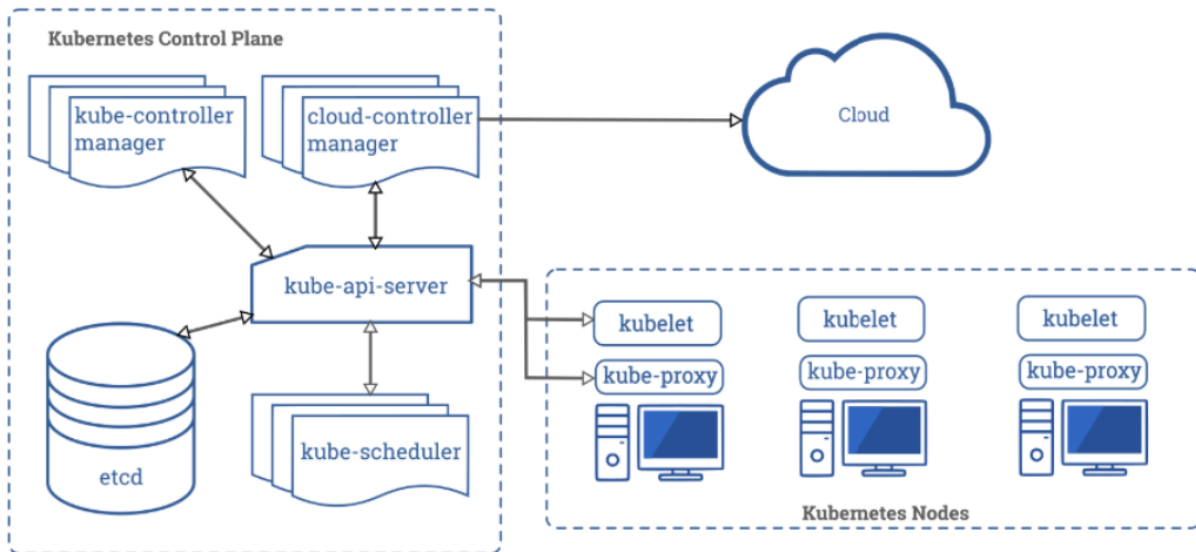
服務帳號:	16
Workload Identity:	18
操作流程	19
開啟API並啟用集群I:	19
建立專屬通道連線:	19
分配通道IP範圍:	19
創建專屬通道:	19
建立資料庫相關資料:	20
建立SQL執行個體:	20
建立資料庫:	23
創建sql帳號:	24
建立服務帳號相關資料:	24
創建服務帳號:	24
創建Kubernetes服務帳號:	26
綁定Kubernetes服務帳號和IAM服務帳號:	27
建立cloud SQL proxy:	28
建立SQL deployment:	29
建立SQL service:	31
將服務改回內部調用:	33
將Google Filestore接入PV和PVC	35
原理	35
NFS:	35
Filestore:	35
PersistentVolume(PV):	35
PersistentVolumeClaim(PVC):	35
操作流程	35
創建filestore:	36
創建持久卷並接入firestore:	38
建立WordPress服務	41
原理	41
pod:	41
Deployment:	41
Service:	41
BackendConfig:	42

操作流程	42
建立Deployment:	42
建立BackendConfig:	45
建立Service:	46
用Auto Scaling來處理高併發	49
原理	49
HorizontalPodAutoscaler:	49
操作流程	49
創建HPA:	49
使用Cloud Monitor和Cloud Logging來做集群狀態監控	51
原理	51
Cloud Logging:	51
Cloud Monitoring:	51
操作流程	51
使用WordPress的Plugin來整合雲端日誌收集系統(可選)	52
原理	52
WP System Log和Papertrail:	52
操作流程	53
接入Ingress, 提供外部訪問、SSL與域名, 成為一個完整的HTTPS網站	54
原理	54
Ingress:	54
操作流程	54
建立完整的ingress:	54
建立靜態IP:	54
建立由Google管理的證書:	56
未來可以增加的功能	60
相關參考資源:	61

總體架構圖



K8S架構圖

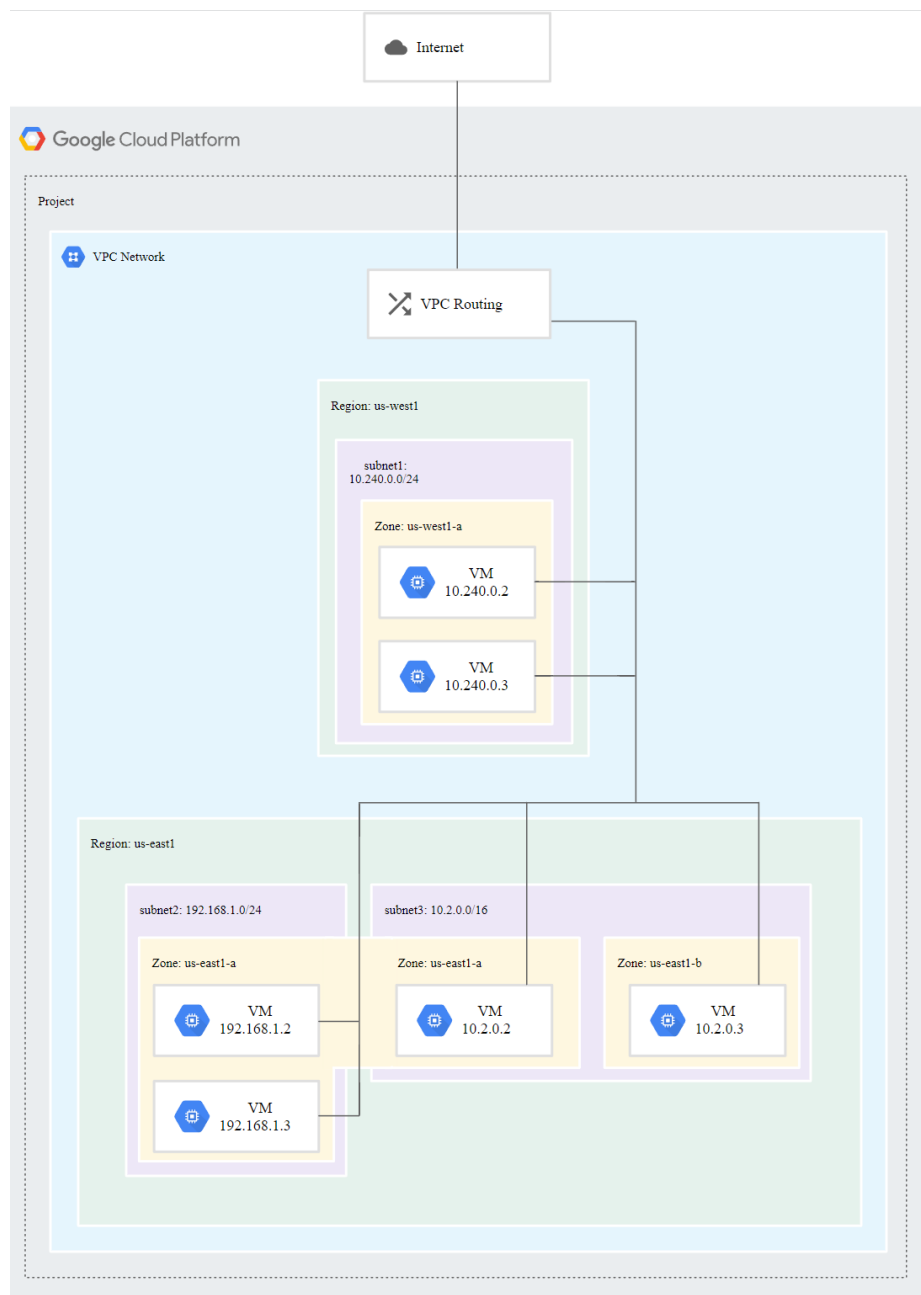


建立VPC以及GKE集群

原理

VPC:

架構圖:

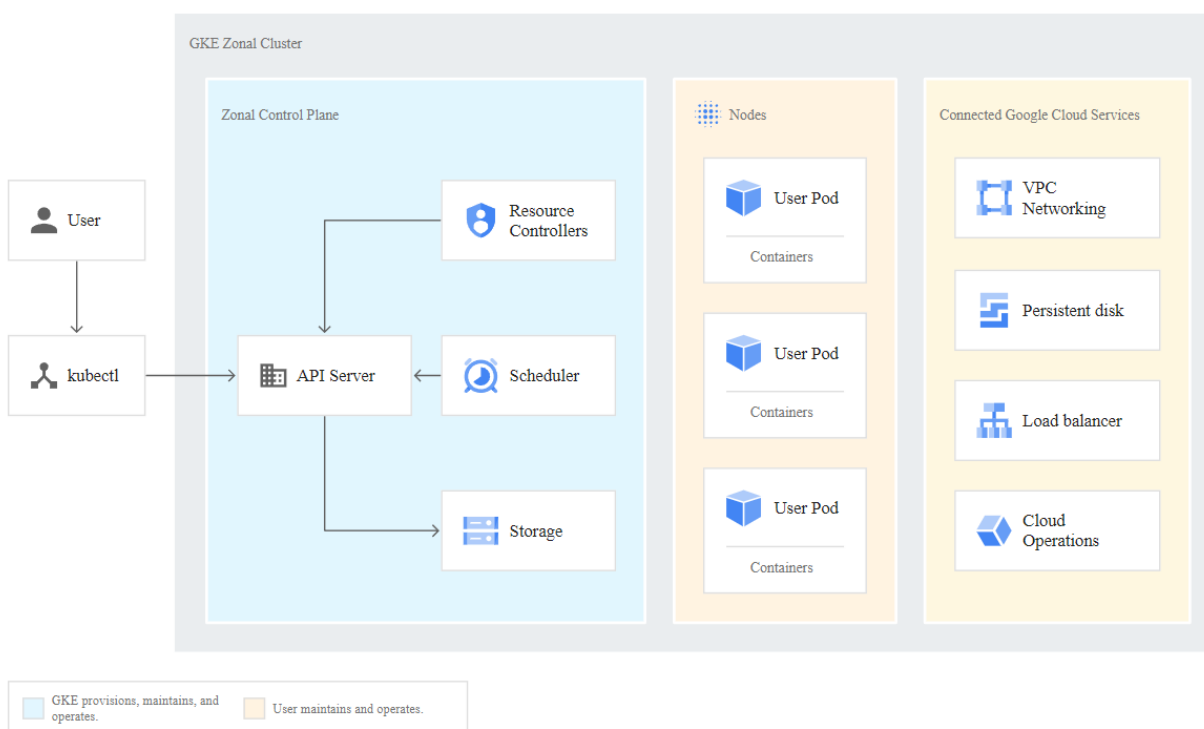


Virtual Private Cloud (VPC) 網絡是物理網絡的虛擬版本，同網路的主網段，可以根據需求再切分出不同的子網段，在google的網路提供以下功能：

- 為 Compute Engine 虛擬機 (VM) 實例提供連接，包括 Google Kubernetes Engine (GKE) 集群、App Engine 環境實例以及 Compute Engine 虛擬機上構建的其他 Google Cloud 產品。
- 為內部 HTTP(S) 負載平衡提供原生的內部 TCP/UDP 負載平衡和代理系統。
- 通過 Cloud VPN 隧道和 Cloud Interconnect 連接，連接到本地網絡。
- 將來自 Google Cloud 外部負載平衡器的流量分配到後端。

GKE:

架構圖：



Google Kubernetes Engine 集群為google提供的Kubernetes託管服務，結合了Kubernetes與google自身服務的優勢，為用戶提供許多優異功能：

- Google Cloud 針對 Compute Engine 實例提供的負載平衡功能。
- 節點池(可用於在集群中指定節點子集以提高靈活性)。
- 自動擴縮集群的節點實例數量。
- 自動升級集群的節點軟件。
- 節點自動修復(以保持節點的正常運行和可用性)。

- 利用 Google Cloud 的運維套件進行日誌記錄和監控，讓用戶可以清楚了解自己集群的狀況
- 操作模式：
 - **Autopilot**: 管理整個集群和節點基礎架構。Autopilot 提供無需人工干預的 Kubernetes 體驗，讓用戶可以專注於工作負載，只需為運行應用所需的資源付費。Autopilot 集群會進行預先配置，並提供可供生產工作負載使用的優化集群配置。
 - 標準: 提供節點配置靈活性以及對集群和節點基礎架構的完全控制。對於使用標準模式創建的集群，用戶需要確定生產工作負載所需的配置，並且需要為用戶使用的節點付費。

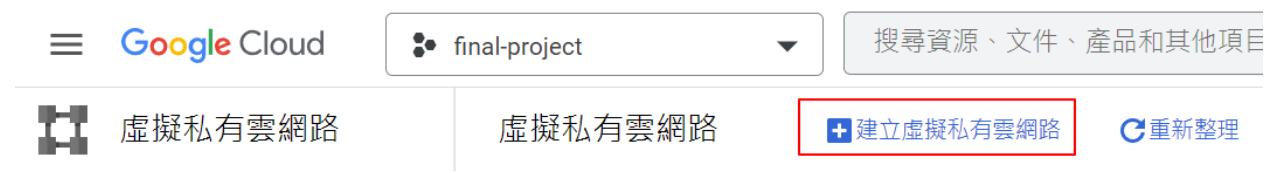
操作流程

注意事項:

- 先在GCP創立一個空白的專案，並且在一個分頁開啟cloud shell的編輯器
- 之後操作其他功能時，一個功能開啟一個分頁，方便檢查用

創建VPC:

- 點選建立VPC
- 填入自定義的命名，本次範例為wordpress-vpc



The screenshot shows the Google Cloud console interface. At the top, there's a navigation bar with the Google Cloud logo, a dropdown menu showing 'final-project', and a search bar. Below this, there are three tabs: '虛擬私有雲網路' (Virtual Private Cloud Network), '虛擬私有雲網路' (Virtual Private Cloud Network), and '+ 建立虛擬私有雲網路' (Create Virtual Private Cloud Network). The third tab is highlighted with a red box. To the right of the tabs is a '重新整理' (Refresh) button.

Below the tabs, there's a list of items. The first item is '填入自定義的VPC命名' (Enter custom VPC name).

Below the list, there's a section titled '← 建立虛擬私有雲網路' (← Create Virtual Private Cloud Network). Inside this section, there's a form with two fields:

- 名稱 *** (Name *): A text input field containing 'wordpress-vpc'. A red box highlights this field. Below the field, there's a hint: '請使用小寫字母、數字或連字號' (Please use lowercase letters, numbers, or hyphens).
- 說明** (Description): A text input field that is currently empty.

- 填入自定義的子網段命名，本次範例為k8s-cluster-subnet
- 區域選asia-east1(台灣)
- IPV4範圍根據建議值填就好，也可以自訂

編輯子網路

名稱 *

k8s-cluster-subnet



請使用小寫字母、數字或連字號

說明

區域 *

asia-east1



IP 堆疊類型

☒ IPv4 (單一堆疊)

☐ IPv4 和 IPv6 (雙重堆疊)

IPv4 範圍 *

10.0.0.0/24



例如：10.0.0.0/24

- VPC內部通訊可以port全開
- VPC外部只要開啟22就好，建立VPC後再回來開啟80port

IPv4 防火牆規則

名稱	類型	目標	篩選器	通訊協定/通訊埠	動作	優先順序 ↑	
<input checked="" type="checkbox"/> wordpress-vpc-allow-custom ?	輸入	套用至所有元件	IP 範圍 : 10.0.0.0/24	all	允許	65,534	編輯
<input type="checkbox"/> wordpress-vpc-allow-icmp ?	輸入	套用至所有元件	IP 範圍 : 0.0.0.0/0	icmp	允許	65,534	
<input type="checkbox"/> wordpress-vpc-allow-rdp ?	輸入	套用至所有元件	IP 範圍 : 0.0.0.0/0	tcp:3389	允許	65,534	
<input checked="" type="checkbox"/> wordpress-vpc-allow-ssh ?	輸入	套用至所有元件	IP 範圍 : 0.0.0.0/0	tcp:22	允許	65,534	
wordpress-vpc-deny-all-ingress ?	輸入	套用至所有元件	IP 範圍 : 0.0.0.0/0	all	拒絕	65,535	
wordpress-vpc-allow-all-egress ?	輸出	套用至所有元件	IP 範圍 : 0.0.0.0/0	all	允許	65,535	

- 其他配置不用改就可以點選建立VPC

動態轉送模式 ?



地區

Cloud Router 只會在當初建立路由器的區域中記錄路徑



全域

全域轉送可讓您透過單一 VPN 或互連網路和 Cloud Router 在所有區域之間動態記錄路徑



如要選取 DNS 政策，請啟用 DNS API

啟用

最大傳輸單位 (MTU)

1460



建立

取消

- 等待一段時間後, VPC已經建立完成

名稱 ↑	區域	子網路	MTU ?	模式	內部 IP 範圍	外部 IP 範圍	次要 IPv4 範圍	開道
▶ default		35	1460	自動	無			
▼ wordpress-vpc		1	1460	自訂	無			
	asia-east1	k8s-cluster-subnet			10.0.0.0/24	無	無	10.0.0.1

建立80 port的防火牆規則:

- 點擊進去VPC

名稱 ↑	區域	子網路	MTU ?	模式	內部 IP 範圍	外部 IP 範圍	次要 IPv4 範圍	開道
▶ default		35	1460	自動	無			
▼ wordpress-vpc		1	1460	自訂	無			
	asia-east1	k8s-cluster-subnet			10.0.0.0/24	無	無	10.0.0.1

- 點擊進去防火牆

[子網路](#)
[靜態內部 IP 位址](#)
[防火牆](#)
[路徑](#)
[虛擬私有雲網路對等](#)

[新增子網路](#)
[流程記錄 ▼](#)

篩選

輸入屬性名稱或值

?

III

<input type="checkbox"/>	名稱 ↑	區域	堆疊類型	內部 IP 範圍	外部 IP 範圍
<input type="checkbox"/>	k8s-cluster-subnet	asia-east1	IPv4	10.0.0.0/24	無

- 點擊新增防火牆規則

< 子網路 靜態內部 IP 位址 **防火牆** 路徑 虛擬私有雲網路對等互連 私人服務連線

新增防火牆規則 刪除

篩選 輸入屬性名稱或值

名稱 ↑	強制執行順序 ↑	類型	部署範圍	規則優先順序
▼ vpc-firewall-rules	1	虛擬私有雲網路	全球	

- 填入自定義的名稱, 本次範例為allow-http

← 建立防火牆規則

防火牆規則會控管執行個體的來往流量。根據預設，系統會封鎖來自所用網路以外的連入流量。[瞭解詳情](#)

名稱 *
 allow-http

請使用小寫字母、數字或連字號

- 填入自定義的標記, 本次範例為allow-http
- IPV4來源選0.0.0.0/0
- 通訊埠填80

目標標記 *
allow-http x

來源篩選器
IPv4 範圍 ▼ ?

來源 IPv4 範圍 *
0.0.0.0/0 x 例如：0.0.0.0/0、192.168.2.0/24 ?

次要來源篩選器
無 ▼ ?

通訊協定和埠 ?

☐ 全部允許

☒ 指定的通訊協定和埠

☒ TCP

通訊埠

80

例如：20, 50-60

- 填完後點選建立

✓ 停用規則

建立

取消

對等指令列

- 檢查防火牆是否有正確建立

名稱	強制執行順序 ↑	類型	部署範圍	規則優先順序	目標	來源	目的地	通訊協定和通訊埠
▼ vpc-firewall-rules	1	虛擬私有雲防火牆規則	全球					
<input type="checkbox"/> allow-http		輸入防火牆規則	全球	1000	標記： allow-http	IPv4 範圍： 0.0.0.0/0	-	tcp:80
<input type="checkbox"/> wordpress-vpc-allow-custom		輸入防火牆規則	全球	65534	全部套用	IPv4 範圍： 10.0.0.0/24	-	all
<input checked="" type="checkbox"/> wordpress-vpc-allow-ssh		輸入防火牆規則	全球	65534	全部套用	IPv4 範圍： 0.0.0.0/0	-	tcp:22

建立GKE集群：

- 選擇建立集群

Kubernetes Engine

Kubernetes 叢集
+ 建立
重新整理

叢集
工作負載
Service 與 Ingress
應用程式
Secret 與 ConfigMap
儲存空間
物件瀏覽器

Kubernetes Engine
Kubernetes 叢集

容器可封裝應用程式，方便應用程式在專屬的獨立環境中部署及執行。容器是在 Kubernetes 叢集中執行。
[瞭解詳情](#)

建立

部署容器
進入快速入門導覽課程

- 選擇建立autopilot集群

建立叢集

選取要使用的叢集模式。



Autopilot: Google manages your cluster (Recommended)

在這種依據 Pod 數量計費的 Kubernetes 叢集中，您僅須選取極少設定即可讓 GKE 管理節點。[瞭解詳情](#)

設定



Standard：由您自行管理叢集

在這種依據節點數量計費的 Kubernetes 叢集中，您可以設定及管理節點。[瞭解詳情](#)

設定



比較叢集模式，藉此進一步瞭解不同之處。

比較

- 填入自定義的集群名稱，本次範例為wordpress-cluster
- 地區選asia-east1

- ✓ 節點：自動化佈建節點、調度資源及維護
- ✓ 網路：在公開或私人叢集中使用虛擬私有雲原生流量轉送功能
- ✓ 安全性：受防護的 GKE 節點和 Workload Identity
- ✓ 遙測：Cloud 作業套件的記錄與監控功能

名稱

wordpress-cluster

叢集名稱開頭須為小寫英文字母，其後最多可接 39 個小寫英文字母、數字或連字號。結尾不得為連字號。叢集名稱一經建立即無法變更。

地區

asia-east1

叢集控制層和節點所在的區域位置。叢集區域一經建立即無法變更。

- 選擇之前建立的VPC
- 選擇之前建立的子網路
- 其他用預設值

網路 *
wordpress-vpc

節點子網路 *
k8s-cluster-subnet

叢集預設 Pod 位址範圍
/17
範例：192.168.0.0/16

服務位址範圍
/22
範例：192.168.0.0/16

隱藏網路選項

- 點擊建立GKE集群

網路選項

進階選項

Click **Create** to create the cluster with these settings turned on.

建立

取消 對等 REST 或 指令列

- 回到功能檢查是否建立成功

Kubernetes Engine

Kubernetes 叢集

+ 建立 + 部署 重新整理

叢集

工作負載

Service 與 Ingress

應用程式

Secret 與 ConfigMap

總覽 觀測能力 成本最佳化

篩選 輸入屬性名稱或值

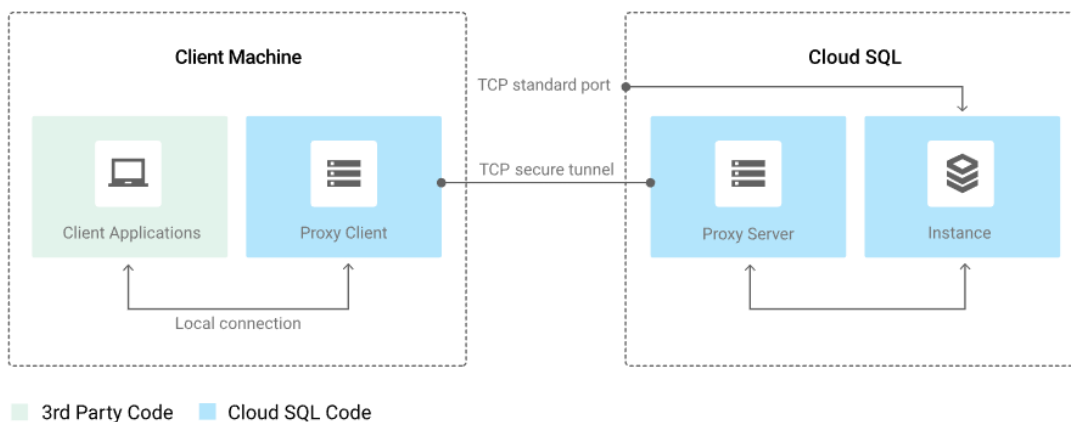
<input type="checkbox"/> 狀態	名稱 ↑	位置	模式	節點數量
<input checked="" type="checkbox"/>	wordpress-cluster	asia-east1	Autopilot	

建立 Cloud SQL Proxy

原理

Cloud SQL Auth:

架構圖:



Cloud SQL Auth 代理通過安全隧道與服務器上運行的代理配套進程進行通信。通過 Cloud SQL Auth 代理建立的每個連接都會創建一個與 Cloud SQL 實例的連接，提供以下優勢：

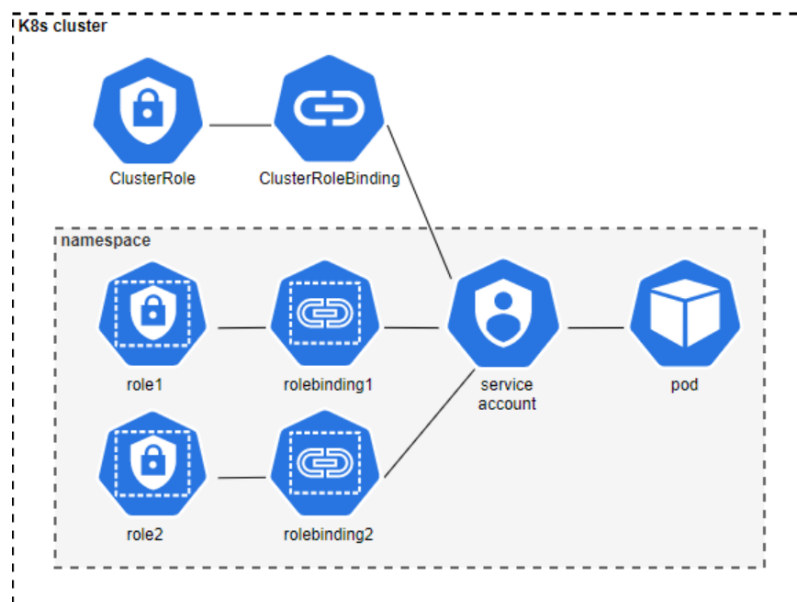
- 安全連接：Cloud SQL Auth 代理使用採用 256 位 AES 加密的 TLS 1.3 自動加密進出數據庫的流量。SSL 證書用於驗證客戶端和服務器身份，獨立於數據庫協議；用戶無需管理 SSL 證書。
- 簡化連接授權：Cloud SQL Auth 代理使用 IAM 權限來控制誰能連接到 Cloud SQL 實例。因此，Cloud SQL Auth 代理會處理 Cloud SQL 的身份驗證，讓用戶無需再提供靜態 IP 地址。
- IAM 數據庫身份驗證。（可選）Cloud SQL Auth 代理支持自動刷新 OAuth 2.0 訪問令牌。如需了解此功能，請參閱 Cloud SQL IAM 數據庫身份驗證。

服務帳號:

服務帳號在GKE上又分為兩種:

- Kubernetes 服務帳號:

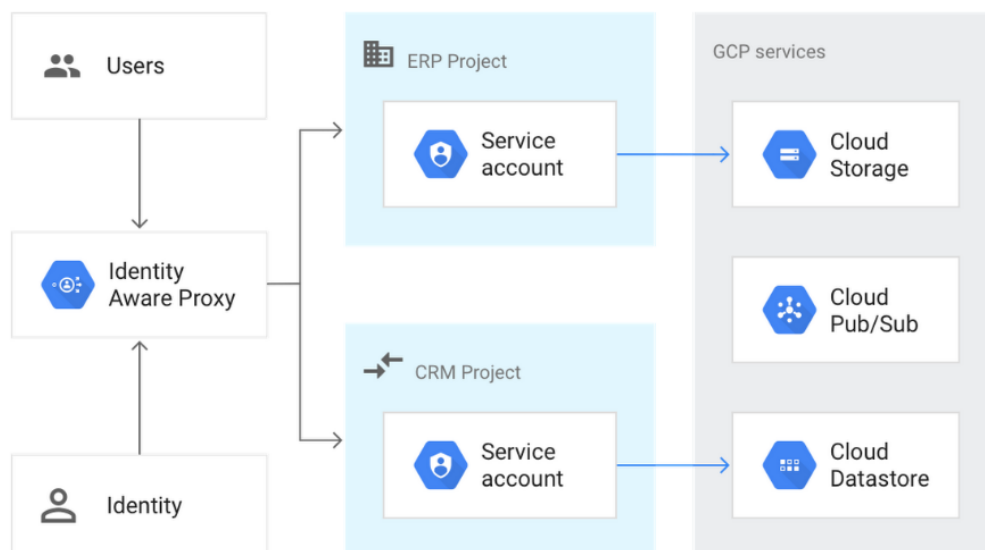
- 架構圖:



- Kubernetes 資源, 用於為 GKE pod 中運行的進程提供身份, 使用RBAC模型

- IAM 服務帳號:

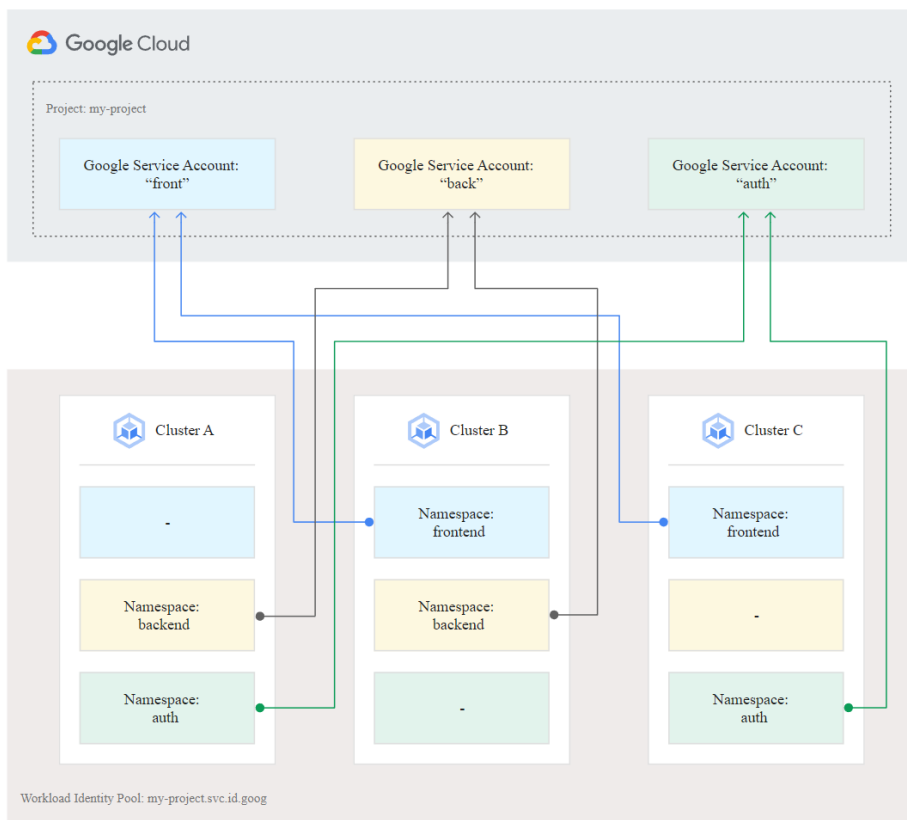
- 架構圖:



- 由應用或計算工作負載(而非真人)使用的特殊帳號。服務帳號由 Identity and Access Management (IAM) 管理。

Workload Identity:

架構圖:



Workload Identity 允許 GKE 集群中的 Kubernetes 服務帳號充當 IAM 服務帳號。訪問 Google Cloud API 時, 使用已配置 Kubernetes 服務帳號的 Pod 會自動作為 IAM 服務帳號進行身份驗證。通過工作負載身份, 用戶可以為集群中的每個應用分配不同的精細身份和授權。

操作流程

前置步驟參考，專用IP部分：

https://cloud.google.com/sql/docs/mysql/connect-instance-kubernetes?hl=zh-cn#go_1

開啟API並啟用集群I:

- 開啟cloud sql所需API:

```
gcloud services enable compute.googleapis.com sqladmin.googleapis.com \
  container.googleapis.com artifactregistry.googleapis.com cloudbuild.googleapis.com
```

- 啟用集群，紅字表示可以替換，必須跟之前建的集群名字一樣

```
gcloud container clusters get-credentials wordpress-cluster --region asia-east1
```

建立專屬通道連線:

分配通道IP範圍:

- 將參數取代成之前步驟創立的VPC名稱，紅字表示可以替換

```
gcloud compute addresses create google-managed-services-default \
  --global \
  --purpose=VPC_PEERING \
  --prefix-length=16 \
  --description="peering range for Google" \
  --network=wordpress-vpc
```

- 看到Created就是建立成功

Created [https://www.googleapis.com/compute/v1/projects/final-project-364908/global/addresses/google-managed-services-default].

創建專屬通道：

- 將參數取代成之前步驟創立的VPC名稱，紅字表示可以替換，遇到詢問則輸入Y

```
gcloud services vpc-peerings connect \
--service=servicenetworking.googleapis.com \
--ranges=google-managed-services-default \
--network=wordpress-vpc
```

- 看到successfully就是建立成功

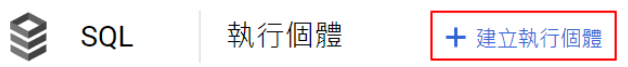
API [servicenetworking.googleapis.com] not enabled on project [517532788734]. Would you like to enable and retry (this will take a few minutes)? (y/N)? y

Enabling service [servicenetworking.googleapis.com] on project [517532788734]...
Operation "operations/acat.p2-517532788734-43c6d8e7-1cb7-49c0-a3f2-1888eec7a670" finished successfully.
Operation "operations/pssn.p24-517532788734-8b8f3f76-a027-4e21-923a-b379a3b25f1c" finished successfully.

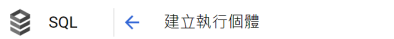
建立資料庫相關資料：

建立SQL執行個體：

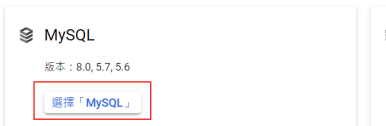
- 建立SQL執行個體



- 選擇MySQL



選擇您的資料庫引擎



需要更多 Cloud SQL 資料庫引擎的相關資訊嗎？[瞭解詳情](#)

- 填入SQL執行個體名稱，本次範例是wordpress-sql
- 填入密碼，本次範例密碼是abc

執行個體資訊

執行個體 ID *

請使用小寫英文字母、數字和連字號，

密碼 *

設定超級使用者的密碼。 [瞭解詳情](#)

- 選擇Development的規格就好

請選擇初始設定

這些建議設定會預先填入這份表單，做為建立執行個體的基礎。日後可視需求進行自訂。

- ☐ Production
Optimized for the most critical workloads. Highly available, performant, and durable.
- ☒ Development
Performant but not highly available, while reducing cost by provisioning less compute and storage.

✓ 設定詳細資料

- 地區選asia-east1
- 可用性選單一區域

選擇區域和可用區供應情形

為了改善效能，請將資料存放在需要這類可用區則可隨時變更。

地區

asia-east1 (台灣)

可用區可用性

☒ 單一區域

如果發生服務中斷情形，不進行容錯移轉

☐ 多可用區 (可用性高)

自動容錯移轉至選定區域內的其他可用區

- 展開設定選項

自訂執行個體

您也可以稍後再自訂執行個體。

☒ 顯示設定選項

- 選擇連線, 選取私人 IP, 並指定之前創立的 VPC

連線

選擇來源與這個執行個體的連結方式, 然後定義哪些網路可以取得連線授權。[瞭解詳情](#)

無論採用哪個選項, 您都可以使用 Cloud SQL Proxy 來進一步提高安全性。[瞭解詳情](#)

執行個體的 IP 指派設定

☒ 私人 IP

指派託管於 Google 的內部虛擬私有雲 IP 位址。您必須具備其他 API 和權限, 才能執行這項操作。IP 位址一經啟用即無法停用。[瞭解詳情](#)

關聯網路

請選取用來建立私人連線的網路

網路 *
wordpress-vpc



已成功建立 **wordpress-vpc** 網路的私人服務存取連線。您現在可以在所有專案的代管服務中使用相同的網路。如要變更這項連線, 請前往 [網路頁面](#)。

- 關閉備份和防刪除

資料保護

自動備份和時間點復原

以最低的費用預防資料遺失。[瞭解詳情](#)

☐ 自動備份

☐ 啟用時間點復原

您可以復原特定時間點的資料 (最小單位不到一秒)。如要複製資料, 您必須啟用二進位檔錄功能。

執行個體防刪除功能

防範意外刪除和資料遺失的情況。[瞭解詳情](#)

☐ 啟用防刪除功能

啟用這項功能之後, 除非將其停用, 否則您將無法刪除這個執行個體

- 執行建立執行個體

標籤

未設定任何標籤

^ 隱藏設定選項

建立執行個體

取消

- 檢查是否建立成功

SQL		執行個體	+ 建立執行個體	⇄ 遷移資料
篩選 輸入屬性名稱或值				
<input type="checkbox"/>	執行個體 ID [?] ↑	類型	公開 IP 位址	私人 IP 位址
<input checked="" type="checkbox"/>	wordpress-sql	MySQL 8.0		10.43.0.3

- 回到cloud shell
- 僅允許此執行個體建立SSL連接, 紅字表示可以替換, 遇到詢問則輸入Y

```
gcloud sql instances patch wordpress-sql --require-ssl
```

- 檢查是否建立成功

```
Patching Cloud SQL instance...done.
Updated [https://sqladmin.googleapis.com/sql/v1beta4/projects/final-project-364908/instances/wordpress-sql].
```

建立資料庫:

- 用命令來建立資料庫, 將instance參數取代成之前步驟創立的SQL執行個體
- 紅字表示可以替換
- 如果變更資料庫名稱, 要記得和wp_deployment.yaml的環境變數要一起改

```
gcloud sql databases create wp --instance=wordpress-sql
```

- 檢查是否建立成功

```
Created database [wp].  
instance: wordpress-sql  
name: wp  
project: final-project-364908
```

創建sql帳號:

- 紅字表示可以替換
 - 如果變更帳號和密碼, 要記得和wp_deployment.yaml的環境變數要一起改
-

```
gcloud sql users create wp-user \  
--instance=wordpress-sql \  
--password=abc
```

- 檢查是否建立成功

```
Creating Cloud SQL user...done.  
Created user [wp-user].
```

建立服務帳號相關資料:

創建服務帳號:

- 用命令來創建IAM服務帳號
-


```
gcloud iam service-accounts create gke-quickstart-service-account \  
--display-name="GKE Quickstart Service Account"
```

- 確認建立結果

Created service account [gke-quickstart-service-account].

- 將服務帳號加上角色
- 紅字的項目ID, 本次範例是final-w(請勿照抄, 位置參考如下圖)

選取專案

 新增專案

搜尋專案和資料夾

Q |

近期專案 已加星號 全部

名稱	ID
✓ ☆ final-paper ?	final-w

gcloud projects add-iam-policy-binding final-w \

--member="serviceAccount:gke-quickstart-service-account@final-w.iam.gserviceaccount.com" \

--role="roles/cloudsql.client"

- 檢查是否建立成功

```
Updated IAM policy for project [final-project-364908].
bindings:
- members:
  - serviceAccount:517532788734@cloudbuild.gserviceaccount.com
  role: roles/cloudbuild.builds.builder
- members:
  - serviceAccount:service-517532788734@gcp-sa-cloudbuild
  role: roles/cloudbuild.serviceAgent
```

創建Kubernetes服務帳號:

- 建立一個服務帳號檔案: service-account.yaml
- 紅字表示可以替換

```
# Copyright 2021 Google LLC
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# [START cloud_sql_mysql_databasesql_gke_quickstart_sa]
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ksa-cloud-sql # TODO(developer): replace this value.
```

```
# [END cloud_sql_mysql_databasesql_gke_quickstart_sa]
```

- 執行創建命令:

```
kubectl apply -f service-account.yaml
```

- 檢查是否建立成功

```
kubectl get serviceaccount
```

NAME	SECRETS	AGE
default	1	122m
ksa-cloud-sql	1	35s

綁定Kubernetes服務帳號和IAM服務帳號:

- 紅字表示可以替換
 - K8s的namespace默認是default, 如果有自訂義namespace則要改
 - ksa-cloud-sql是前面步驟產生的服務帳號
-

```
gcloud iam service-accounts add-iam-policy-binding \
  --role="roles/iam.workloadIdentityUser" \
  --member="serviceAccount:final-w.svc.id.goog[default/ksa-cloud-sql]" \
  gke-quickstart-service-account@final-w.iam.gserviceaccount.com
```

- 檢查是否建立成功

```
bindings:
- members:
  - serviceAccount:final-project-364908.svc.id.goog[default/ksa-cloud-sql]
  role: roles/iam.workloadIdentityUser
etag: BwXqhK58xQs=
version: 1
```

- 紅字表示可以替換
- 用kubectl annotate添加綁定註解
- ksa-cloud-sql是前面步驟產生的服務帳號

kubectl annotate serviceaccount \

ksa-cloud-sql \

iam.gke.io/gcp-service-account=gke-quickstart-service-account@**final-w**.iam.gserviceaccount.com

-
- 檢查是否建立成功

serviceaccount/ksa-cloud-sql annotated

建立cloud SQL proxy:

- 紅字表示可以替換, 為之前步驟建的SQL執行個體名稱

gcloud sql instances describe **wordpress-sql** --format='value(connectionName)'

- 執行結果就是執行個體的全名

```
final-w:asia-east1:wordpress-sql
```

建立SQL deployment:

- 建立一個檔案: sql_deployment.yaml
- 紅字表示可以替換
- 修改執行個體全名, 本次範例為上個步驟的命令結果: final-w:asia-east1:wordpress-sql
- ksa-cloud-sql是前面步驟填的服務帳號

```
# Copyright 2021 Google LLC
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# [START cloud_sql_mysql_databasesql_gke_quickstart_deployment]
apiVersion: apps/v1
kind: Deployment
```

```

metadata:
  name: gke-cloud-sql-quickstart
spec:
  selector:
    matchLabels:
      app: gke-cloud-sql-app
  template:
    metadata:
      labels:
        app: gke-cloud-sql-app
    spec:
      serviceAccountName: ksa-cloud-sql
      containers:
        - name: cloud-sql-proxy
          # This uses the latest version of the Cloud SQL proxy
          # It is recommended to use a specific version for production
environments.
          # See: https://github.com/GoogleCloudPlatform/cloudsql-proxy
          image: gcr.io/cloudsql-docker/gce-proxy:latest
          command:
            - "/cloud_sql_proxy"

            # If connecting from a VPC-native GKE cluster, you can use the
            # following flag to have the proxy connect over private IP
            - "-ip_address_types=PRIVATE"

            # tcp should be set to the port the proxy should listen on
            # and should match the DB_PORT value set above.

```

```

# Defaults: MySQL: 3306, Postgres: 5432, SQLServer: 1433
- "-instances=final-w:asia-east1:wordpress-sql=tcp:0.0.0.0:3306"

securityContext:
  # The default Cloud SQL proxy image runs as the
  # "nonroot" user and group (uid: 65532) by default.
  runAsNonRoot: true

# [END cloud_sql_mysql_databasesql_gke_quickstart_deployment]

```

-
- 執行創建命令:
-

```
kubectl apply -f sql_deployment.yaml
```

- 檢查是否建立成功
-

```
watch -n1 kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
gke-cloud-sql-quickstart-585577b4f-bkrs9	1/1	Running	0	2m27s

建立SQL service:

- 建立一個檔案: sql_service.yaml
-

```

# Copyright 2021 Google LLC
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at

```

```
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# [START cloud_sql_mysql_databasesql_gke_quickstart_service]
# The service provides a load-balancing proxy over the gke-cloud-sql-app
# pods. By specifying the type as a 'LoadBalancer', Kubernetes Engine will
# create an external HTTP load balancer.
apiVersion: v1
kind: Service
metadata:
  name: gke-cloud-sql-app
spec:
  #先使用LoadBalancer來暴露IP來檢查是否正常,確認正常後再註解掉
  type: LoadBalancer
  selector:
    app: gke-cloud-sql-app
  ports:
    - name: gke-cloud-sql-app
      port: 3306
      targetPort: 3306
# [END cloud_sql_mysql_databasesql_gke_quickstart_service]
```

- 執行創建命令：

```
kubectl apply -f sql_service.yaml
```

- 檢查是否建立成功

```
watch -n1 kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
gke-cloud-sql-app	LoadBalancer	10.187.0.187	35.234.49.90	3306:32512/TCP	64s
kubernetes	ClusterIP	10.187.0.1	<none>	443/TCP	69m

- 看到external-ip後用mysql命令(-h = host)登陸來測試是否能連通
- 紅字表示可以替換
- -u為之前步驟建立的sq帳號
- -p為之前步驟建立的sq帳號的密碼, 注意-p後面沒有空格

```
mysql -h 35.236.138.34 -u wp-user -pabc
```

- 連線成功結果會如下圖, 確認成功輸入exit離開

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statemer
mysql> █
```

將服務改回內部調用:

- 把sql_service.yaml中的LoadBalancer註解掉然後重新apply

spec:

#先使用LoadBalancer來暴露IP來檢查是否正常, 確認正常後再註解掉

#type: LoadBalancer

```
kubectl apply -f sql_service.yaml
```

- 此時再去檢查service，確認external-ip已經消失
-

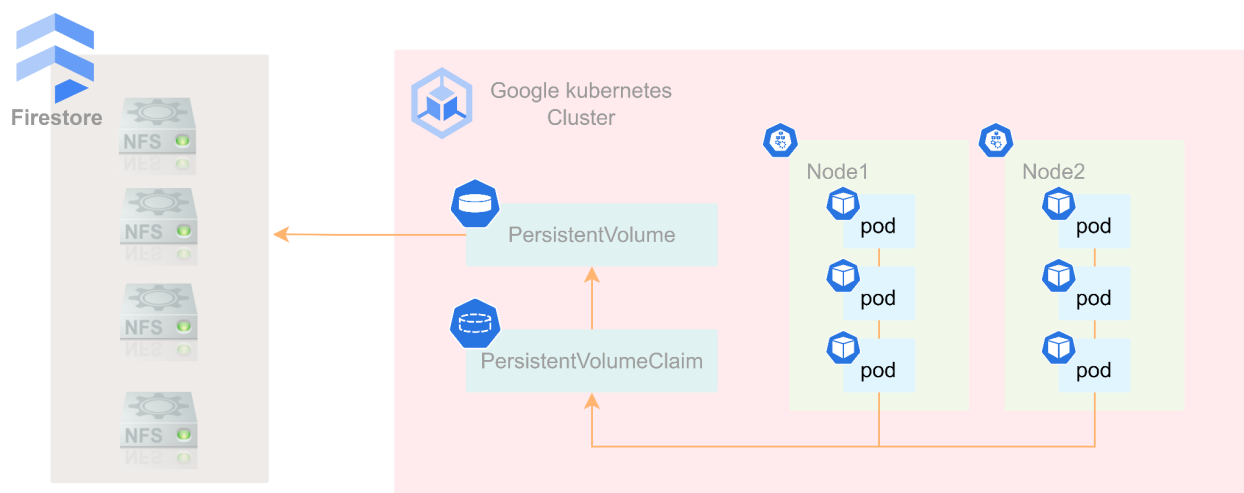
```
watch -n1 kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
gke-cloud-sql-app	ClusterIP	10.187.0.187	<none>	3306/TCP	2m30s
kubernetes	ClusterIP	10.187.0.1	<none>	443/TCP	70m

將Google Filestore接入PV和PVC

原理

架構圖:



NFS:

是一種分散式檔案系統，力求客戶端主機可以存取伺服器端檔案，並且其過程與存取本地儲存時一樣

Filestore:

Cloud Filestore 是一款託管的NFS，用於需要檔案系統介面和共享檔案系統的應用程式，提供低延遲的應用程式儲存空間作業。針對容易受到延遲時間影響的工作負載，例如高效能運算、資料分析或其他需要大量中繼資料的應用程式。

PersistentVolume(PV):

是集群中的一塊存儲，可以由管理員事先製備，或者使用存儲類(Storage Class)來動態製備。持久卷是集群資源，就像節點也是集群資源一樣。PV持久捲和普通的Volume一樣，也是使用卷插件來實現的，只是它們擁有獨立於任何使用PV的Pod的生命週期。此API對象中記述了存儲的實現細節，無論其背後是NFS、iSCSI還是特定於雲平台的存儲系統。

PersistentVolumeClaim(PVC):

表達的是用戶對存儲的請求。概念上與Pod類似。Pod會耗用節點資源，而PVC申領會耗用PV資源。Pod可以請求特定數量的資源(CPU和內存)；同樣PVC申領也可以請求特定的大小和訪

問模式（例如，可以要求 PV 卷能夠以 ReadWriteOnce、ReadOnlyMany 或 ReadWriteMany 模式之一來掛載）。

操作流程

創建filestore:

- 點擊建立執行個體



The screenshot shows the Google Cloud Filestore console. On the left, there is a sidebar with 'Filestore' and '執行個體' (Instances) selected. The main area shows the '執行個體' tab with a description: '執行個體是全代管的網路附加儲存系統，可以與 Google Compute Engine 執行個體搭配使用。' and a link '瞭解詳情'. Below this is a search bar '篩選 輸入屬性名稱或值' and a table with columns: '執行個體編號', '檔案共用區名稱', '建立時間', and '服務級別'. A red box highlights the '建立執行個體' button in the top right corner.

- 輸入執行個體命名，本次範例為wordpress-nfs

為執行個體命名

執行個體 ID *

wordpress-nfs

一經選取即無法變更，而且在其區域中不得重複以英文字母開頭。

說明 (選填)

- 容量保持預設值1TB
- 地區為asia-east1，區域為asia-east1-a
- 選擇之前創立的VPC

分配容量 *

 TiB

可佈建 1 至 63.9 TiB

選取資料的儲存位置

為提升效能並降低網路費用，請讓執行個體與要連結的 VM 位於相同區域。選擇後即無法變更。

地區 * asia-east1 ▼	區域 * asia-east1-a ▼
----------------------	------------------------

設定連線

選取網路和位址範圍，讓用戶端用來存取您的執行個體。[瞭解詳情](#)

虛擬私有雲網路
選擇後即無法變更。[瞭解詳情](#)

網路 *
wordpress-vpc ▼

- 填入檔案共用區名稱，本次範例為wordpress
- 完成後點選建立

設定檔案共用區

檔案共用區名稱 *

wordpress

選擇後即無法變更。請使用小寫英文字母、數字和底線，並以英文字母開頭。

存取權控管

☒ 為虛擬私有雲網路中的所有用戶端授予存取權限☐ 依據 IP 位址或範圍限制存取權

預設移除所有用戶端的存取權，並啟用 Root 權限壓縮

建立標籤

+ 新增標籤

建立

取消

- 過一段時間後重新整理，就可以看到建立的共用區名稱和IP，表示建立成功

篩選 輸入屬性名稱或值

號	檔案共用區名稱	建立時間	服務級別	位置	IP 位址	容量
-	wordpress	2022年 10月8日 晚上 9:10:26	BASIC_HDD	asia- east1- a	10.43.1.2	1 TiB

創建持久卷並接入firestore:

- 建立持久卷檔案，filestore_pv.yaml
- 紅字表示可以替換
- 填入自定義的檔案共用區名稱，本次範例為wordpress(/不能去掉)
- 修改servier的IP為創建成功產生的IP，本次範例為10.140.1.2

apiVersion: v1

kind: PersistentVolume

metadata:

```

name: fileserver

labels:
  name: fileserver

spec:
  capacity:
    storage: 1T
  accessModes:
    - ReadWriteMany
  nfs:
    path: /wordpress
    server: 10.140.1.2

```

-
- 執行創建命令:
-

```
kubectl apply -f filestore_pv.yaml
```

-
- 檢查是否建立成功
-

```
kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
fileserver	1T	RWX	Retain	Available				18s

- 建立持久卷聲明檔案, filestore_pvc.yaml
-

```
apiVersion: v1
```

```
kind: PersistentVolumeClaim
```

```
metadata:
```

```
  name: fileserver-claim
```



```
spec:
  # Specify "" as the storageClassName so it matches the PersistentVolume's
  # StorageClass.
  # A nil storageClassName value uses the default StorageClass. For details, see
  # https://kubernetes.io/docs/concepts/storage/persistent-volumes/#class-1
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 1T
  selector:
    matchLabels:
      name: fileserver
```

-
- 執行創建命令:
-

```
kubectl apply -f filestore_pvc.yaml
```

- 檢查是否建立成功
-

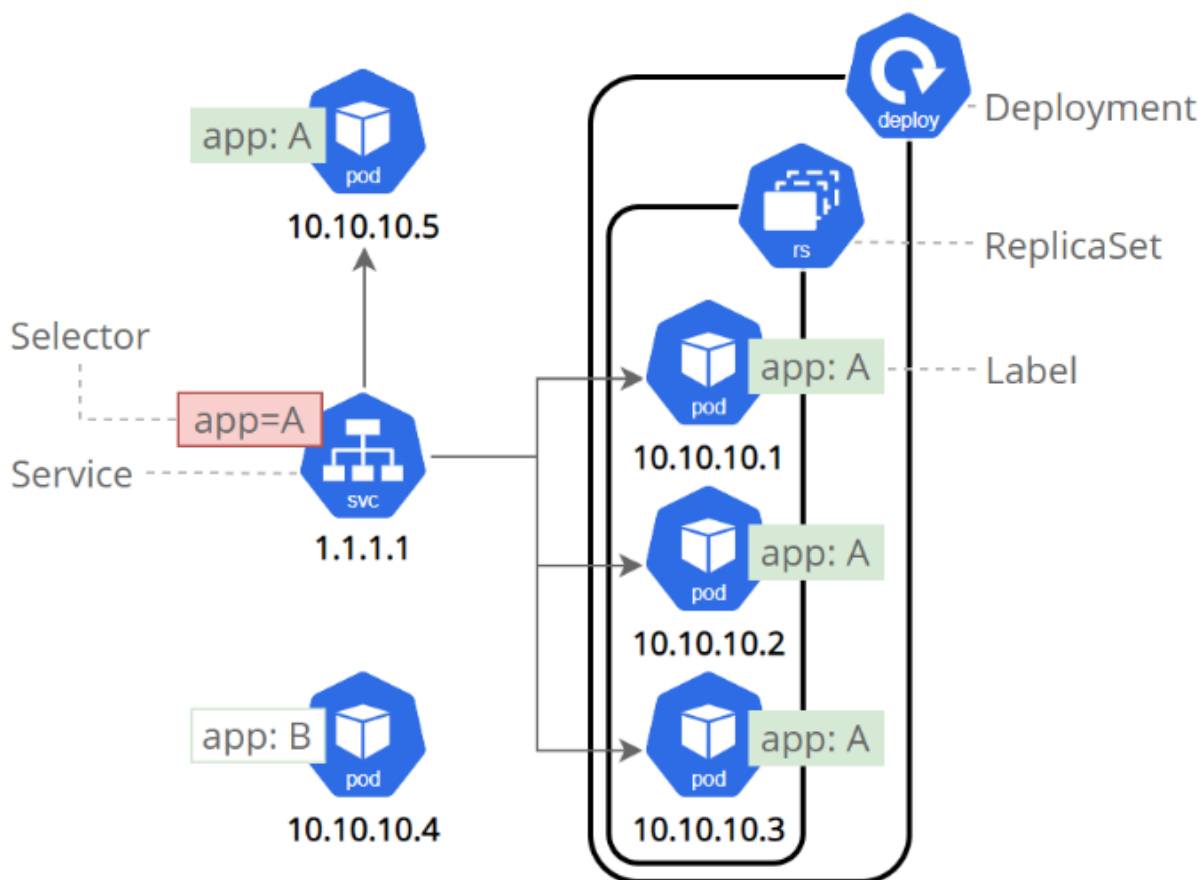
```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
fileserver-claim	Bound	fileserver	1T	RWX		31s

建立WordPress服務

原理

架構圖:



pod:

Pod 是可以在 Kubernetes 中創建和管理的、最小的可部署的計算單元。

Deployment:

一個 Deployment 為 Pod 和 ReplicaSet 提供聲明式的更新能力, pod會根據Deployment定義中的內容來運作服務

Service:

將運行在一組 Pods 上的應用程序公開為網絡服務的抽象方法。

使用 Kubernetes, 無需修改應用程序即可使用不熟悉的服務發現機制。Kubernetes 為 Pod 提供自己的 IP 地址, 並為一組 Pod 提供相同的 DNS 名, 並且可以在它們之間進行負載均衡。

BackendConfig:

通用的一種配置格式, 可以跟不同服務做結合, 達到靈活配置的結果, 在這裡綁定到service上, 作為之後ingress的健康檢查策略配置使用

操作流程

建立Deployment:

- 建立一個yaml檔案: wp_deployment.yaml
- Env為環境變數, 根據需要替換調整
- livenessProbe是存活探針, 不指定WP的特定路徑或導致檢查根目錄(/), 造成容器重啟

apiVersion: apps/v1

kind: Deployment

metadata:

 labels:

 app: wordpress

 name: wordpress

spec:

 selector:

 matchLabels:

 app: wordpress

 strategy: {}

 template:

 metadata:

 labels:

```

    app: wordpress

spec:
  containers:
  - image: wordpress
    name: wordpress
    ports:
    - containerPort: 80
      name: http
  resources:
    #此部分的資源限制是為了觸發HPA用的
    requests:
      cpu: 200m
      memory: 256Mi
    env:
      # wordpress連接資料庫的參數,host就是k8s的service,集群內可以透過service
      # name來內部通訊
      # 此部分為之前創建的SQL相關資料
      - name: WORDPRESS_DB_HOST
        value: gke-cloud-sql-app:3306
      - name: WORDPRESS_DB_USER
        value: wp-user
      - name: WORDPRESS_DB_PASSWORD
        value: abc
      - name: WORDPRESS_DB_NAME
        value: wp
    volumeMounts:
      - mountPath: "/var/www/html"
        name: mypvc

```

```

    livenessProbe:
      httpGet:
        port: 80
        path: /wp-admin/install.php
      periodSeconds: 5
      timeoutSeconds: 30
    volumes:
      - name: mypvc
        persistentVolumeClaim:
          claimName: fileserver-claim
status: {}

```

- 執行創建命令:

```
kubectl apply -f wp_deployment.yaml
```

- 執行檢查命令，看到running就是正常啟動

```
watch -n1 kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
wordpress-78c7f8b6d5-zp8k1	1/1	Running	0	15s

- 如果有問題用describe來看狀態:

```
kubectl describe pod <pod的名稱>
```

建立BackendConfig:

- 建立一個yaml檔案: backend_config.yaml
 - 詳盡配置可以參考:
https://cloud.google.com/kubernetes-engine/docs/how-to/ingress-features#direct_health
-

apiVersion: cloud.google.com/v1

kind: BackendConfig

metadata:

name: wordpress-config

spec:

healthCheck:

checkIntervalSec: 30

timeoutSec: 15

healthyThreshold: 1

unhealthyThreshold: 10

type: HTTP

requestPath: /wp-admin/install.php

port: 80

- 執行創建命令:
-

```
kubectl apply -f backend_config.yaml
```

- 檢查命令:

```
kubectl get BackendConfig
```

```
NAME                AGE
wordpress-config    36s
```

建立Service:

- 建立一個yaml檔案: wp_service.yaml
 - 綁定配置是作為ingress的健康檢查使用
-

```
apiVersion: v1
```

```
kind: Service
```

```
metadata:
```

```
  labels:
```

```
    app: wordpress
```

```
  name: wordpress
```

```
  annotations:
```

```
    cloud.google.com/backend-config: '{"default": "wordpress-config"}'
```

```
    cloud.google.com/neg: '{"ingress": true}'
```

```
spec:
```

```
  ports:
```

```
  - port: 80
```

```
    protocol: TCP
```

```
    targetPort: 80
```

```
  selector:
```

```
    app: wordpress
```

```
#type: NodePort
```

```
#先使用LoadBalancer來暴露IP來檢查是否正常,確認正常後再改回NodePort
```

```
type: LoadBalancer
```

- 執行創建命令:

```
kubectl apply -f wp_service.yaml
```

- 執行檢查命令，一開始會看到pending，等到IP出來就表示完成

```
watch -n1 kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
gke-cloud-sql-app	ClusterIP	10.187.0.187	<none>	3306/TCP	18m
kubernetes	ClusterIP	10.187.0.1	<none>	443/TCP	86m
wordpress	LoadBalancer	10.187.3.50	<pending>	80:30366/TCP	42s

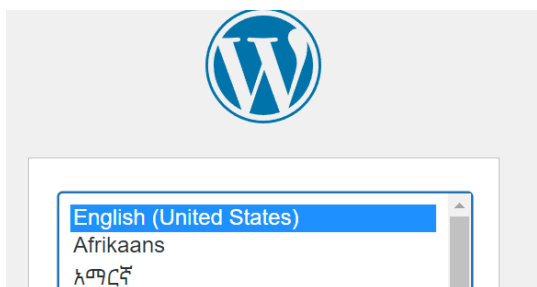
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
gke-cloud-sql-app	ClusterIP	10.187.0.187	<none>	3306/TCP	18m
kubernetes	ClusterIP	10.187.0.1	<none>	443/TCP	87m
wordpress	LoadBalancer	10.187.3.50	35.234.49.90	80:30366/TCP	58s

- 如果有問題用describe來看狀態:

```
kubectl describe service <servie的名稱>
```


- 此時把external-ip貼到瀏覽器，會看到不安全連線但是看的到安裝畫面，表示建立成功

⚠ 不安全 | 34.81.8.187

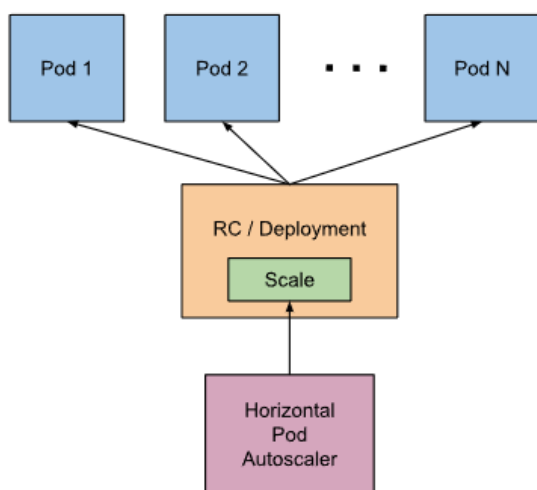


用Auto Scaling來處理高併發

原理

HorizontalPodAutoscaler:

架構圖:



HorizontalPodAutoscaler自動更新工作負載資源（例如 Deployment 或者 StatefulSet），目的是自動擴縮工作負載以滿足需求，可以指定不同擴縮策略滿足不同場景。

操作流程

創建HPA:

- 創建檔案: wp_hpa.yaml
- scaleTargetRef底下的name就是wordpress的service
- minReplicas為最小pod數量
- maxReplicas為最大pod數量
- targetCPUUtilizationPercentage為超過CPU使用率的百分比，因為之前的deployment為200m，也就是使用1/5的CPU使用率，所以只要CPU使用率超過百分之50就或自動擴容，同理用量下降也會自動縮容

```
# wordpress-hpa.yaml
```

```

apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: wordpress-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: wordpress
  minReplicas: 2
  maxReplicas: 5
  targetCPUUtilizationPercentage: 50

```

-
- 執行創建命令:
-

```
kubectl apply -f wp_hpa.yaml
```

- 檢查是否建立成功
-

```
watch -n1 kubectl get pod
```

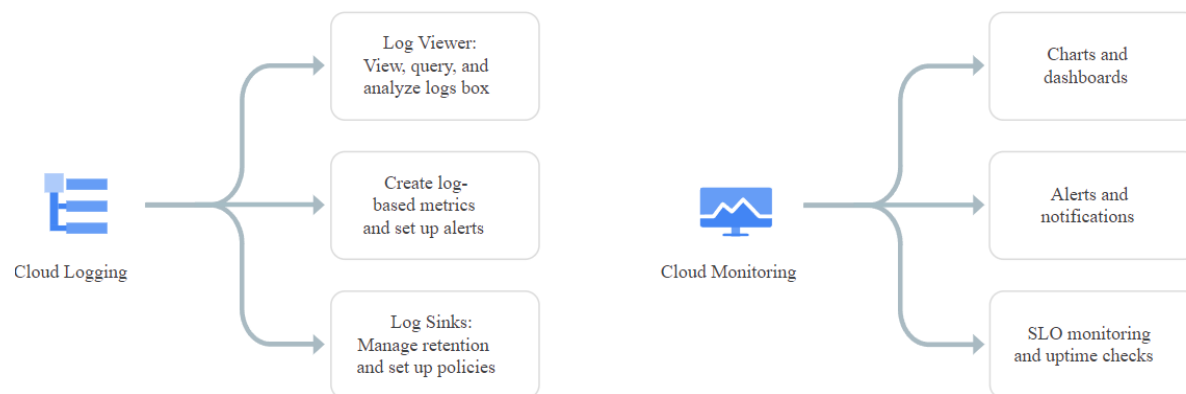
- 此時pod數量已經改為2個
- 這裡跳過壓測, 可以使用siege, 參考連結:
- <https://www.kali.org/tools/siege/>

NAME	READY	STATUS	RESTARTS	AGE
gke-cloud-sql-quickstart-697685db9-gs865	1/1	Running	0	69m
wordpress-74bd7dd446-fkvtw	1/1	Running	0	11m
wordpress-74bd7dd446-zlnp9	1/1	Running	0	23s

使用Cloud Monitor和Cloud Logging來做集群狀態監控

原理

架構圖：



Cloud Logging:

Cloud Logging 是一項全代管服務，不僅可以大規模處理作業，也能擷取 GKE 環境、VM 和 Cloud 內部/外部其他服務中的應用程式與系統記錄檔資料，以及自訂記錄檔資料。透過記錄檔分析功能將 BigQuery 的強大功能整合至 Cloud Logging，以便取得進階效能、疑難排解、安全性和業務深入分析。

Cloud Monitoring:

Cloud Monitoring 可讓用戶掌握雲端應用程式的效能、運作時間和整體健康狀態。這個工具會收集各項 Google Cloud 服務、託管型運作時間探測器、應用程式檢測設備，以及多種常見應用程式元件的指標、事件與中繼資料。透過圖表和資訊主頁以視覺化方式呈現這項資料，並建立快訊，這樣只要指標超出預期範圍，用戶就會收到通知。

操作流程

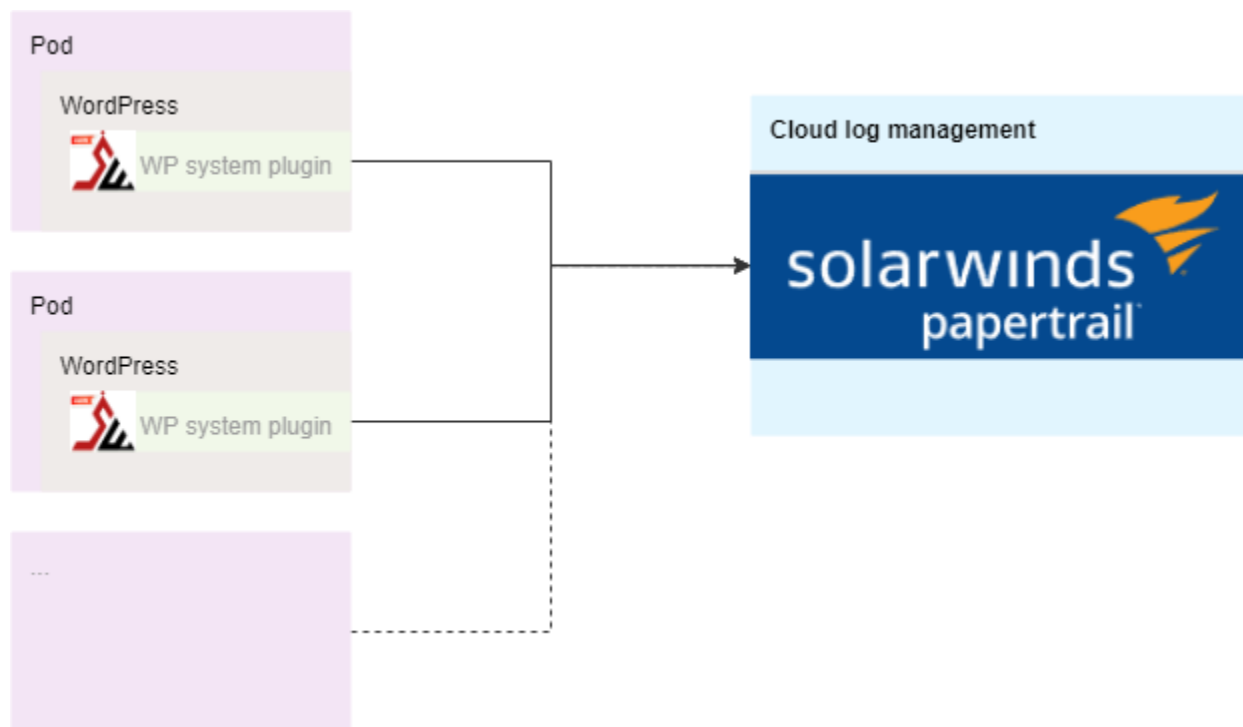
新一個GKE集群後，就可以直接使用Cloud Logging和Cloud Monitoring的功能。

使用WordPress的Plugin來整合雲端日誌收集系統(可選)

原理

WP System Log和Papertrail:

架構圖:



使用WordPress的插件，將日誌輸出到雲端的集中式日誌管理系統Papertrail，但是因為要收費，所以請斟酌使用。

操作流程

安裝插件: WP system log, 就可以看到選項cloud interation



WP System Log

Most detailed WP System Log
with User Requests Tracking...

開發者: *SWIT*

[立即安裝](#)[更多詳細資料](#)

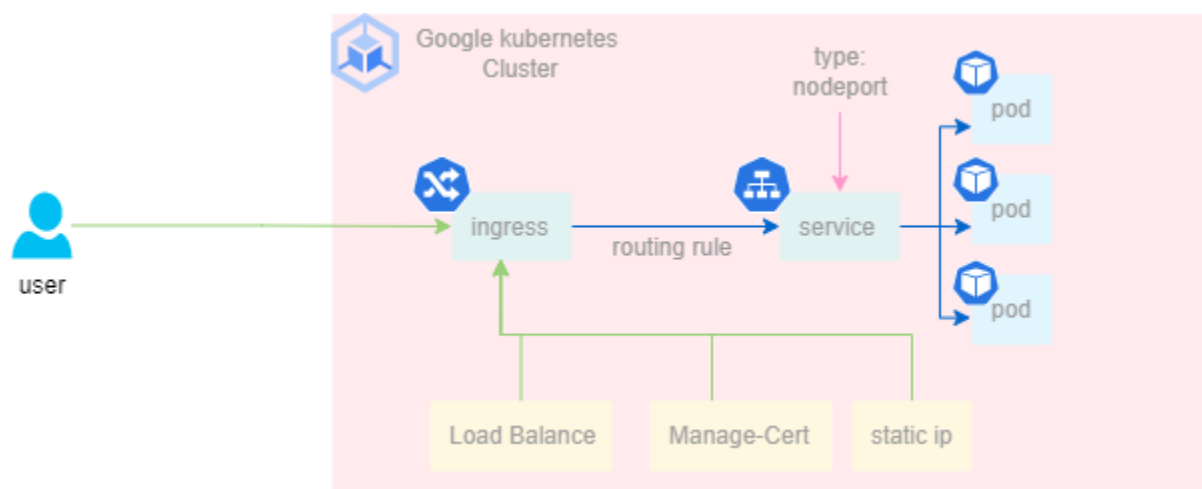
The screenshot shows the WP System Log plugin interface. On the left is a sidebar menu with options: 控制台, 文章, 媒體, 頁面, 留言, WinterLock (highlighted), Activity Log, Favourite Logs, Disabled Logs, Control Security, Log Alerts, History, User Sessions, Reports, Cloud Integration (highlighted with a red box), and Related plugins. The main content area is titled 'Cloud integration' and contains two buttons: '+ Add New Papertrail Cloud' (highlighted with a red box) and '+ Add New External MySQL Connection'. Below these is a section 'Manage Cloud integration Data' with a search bar and a table. The table has columns: #, Title, Component, and Program name. It currently shows 'No matching records found'. At the bottom, a light blue box contains the text: 'Here you can configure cloud logging to Papertrail app'.

接入Ingress, 提供外部訪問、SSL與域名, 成為一個完整的HTTPS網站

原理

Ingress:

架構圖:



Ingress 是對集群中服務的外部訪問進行管理的 API 對象, 典型的訪問方式是 HTTP, Ingress 可以提供負載均衡、SSL 終結和基於名稱的虛擬託管

操作流程

參考資料:<https://cloud.google.com/kubernetes-engine/docs/how-to/managed-certs>

建立完整的ingress:

建立靜態IP:

- 根據命令創建靜態IP
- 紅字表示可以替換

```
gcloud compute addresses create wordpress --global
```

- 根據命令取得的靜態IP
- 紅字表示可以替換

gcloud compute addresses describe **wordpress** --global

- 取得結果如下圖

```
address: 34.111.181.141
addressType: EXTERNAL
creationTimestamp: '2022-10-08T19:34:39.336-07:00'
description: ''
id: '1362733709629820368'
ipVersion: IPV4
kind: compute#address
name: wordpress
networkTier: PREMIUM
```

- 先把原本的wordpress的service的改成Nodeport

type: NodePort

#先使用LoadBalancer來暴露IP來檢查是否正常,確認正常後再改回NodePort

#type: LoadBalancer

-
- 重新執行創建命令:
-

```
kubectl apply -f wp_service.yaml
```

- 檢查結果

```
kubectl get service
```

```
wei33107@cloudshell:~/wordpress/ingress (final-project-364908)$ kubectl get service
NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
gke-cloud-sql-app                  ClusterIP      10.187.0.239   <none>          3306/TCP          133m
kubernetes                         ClusterIP      10.187.0.1     <none>          443/TCP           4h59m
wordpress                         NodePort       10.187.3.98    <none>          80:31030/TCP      4h22m
```

建立由Google管理的證書：

- 建立憑證檔案, managed-cert.yaml
 - 紅字表示可以替換
 - 域名請務必修改
-

```
apiVersion: networking.gke.io/v1
```

```
kind: ManagedCertificate
```

```
metadata:
```

```
  name: managed-cert
```

```
spec:
```

```
  domains:
```

```
    - cgc101.tibame.ga
```

- 先測試域名和IP是否綁定成功

```
Ping cgc101.tibame.ga [34.111.54.138] (使用 32 位元組的資料):
回覆自 34.111.54.138: 位元組=32 時間=31ms TTL=54
回覆自 34.111.54.138: 位元組=32 時間=40ms TTL=54
```

- 執行創建命令:

```
kubectl apply -f managed-cert.yaml
```

- 檢查證書是否創建成功
 - Status狀態是Provisioning表示還在創建, 需要串完ingress才會轉active
-

```
kubectl describe managedcertificate managed-cert
```

```
Status:
Certificate Name:      mcrt-27f26d2b-0d2d-422a-b8d3-c71db3ae1408
Certificate Status:    Provisioning
Domain Status:
  Domain:  cgc101.tibame.ga
  Status:  Provisioning
Events:
  Type      Reason      Age      From                                     Message
  ----      -
  Normal    Create      16s      managed-certificate-controller          Create SslCertificate mcrt-27f26d2b-0d2d-422a-
```

- 創建ingress, 檔案名稱為 managed-cert-ingress.yaml
 - kubernetes.io/ingress.global-static-ip-name為之前步驟創建靜態IP的別名, 可以替換
 - networking.gke.io/managed-certificates為之前步驟創建的憑證檔案的name, 本次範例為managed-cert
-

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  name: managed-cert-ingress
```

```
  annotations:
```

```
    kubernetes.io/ingress.global-static-ip-name: wordpress
```

```
    networking.gke.io/managed-certificates: managed-cert
```

```
    kubernetes.io/ingress.class: "gce"
```

spec:

defaultBackend:

service:

name: wordpress

port:

number: 80

- 執行創建命令:

```
kubectl apply -f managed-cert-ingress.yaml
```

- 在GKE的頁面，可以看到ingress的執行狀態，如果有報錯會顯示，完成則顯示OK

The screenshot shows the Google Cloud Platform console interface for Kubernetes Engine. On the left is a navigation menu with options like '叢集' (Clusters), '工作負載' (Workloads), 'Service 與 Ingress', '應用程式式' (Applications), 'Secret 與 ConfigMap', '儲存空間' (Storage), and '物件瀏覽器' (Object Browser). The main panel is titled 'Service 與 Ingress' and includes a '重新整理' (Refresh) button and a '刪除' (Delete) button. Below this, there are filters for '叢集' (Cluster) and '命名空間' (Namespace). The 'INGRESS' tab is selected, showing a description of Ingress. A table lists the ingress resources, with 'managed-cert-ingress' highlighted. The status of this resource is 'Creating...', which is enclosed in a red box.

名稱 ↑	狀態	類型
managed-cert-ingress	Creating ...	外部 HTTP(S) 負載平衡器

篩選 篩選輸入

名稱 ↑	狀態	類型
managed-cert-ingress	OK	外部 HTTP(S) 負載平衡器

- 檢查證書是否創建成功

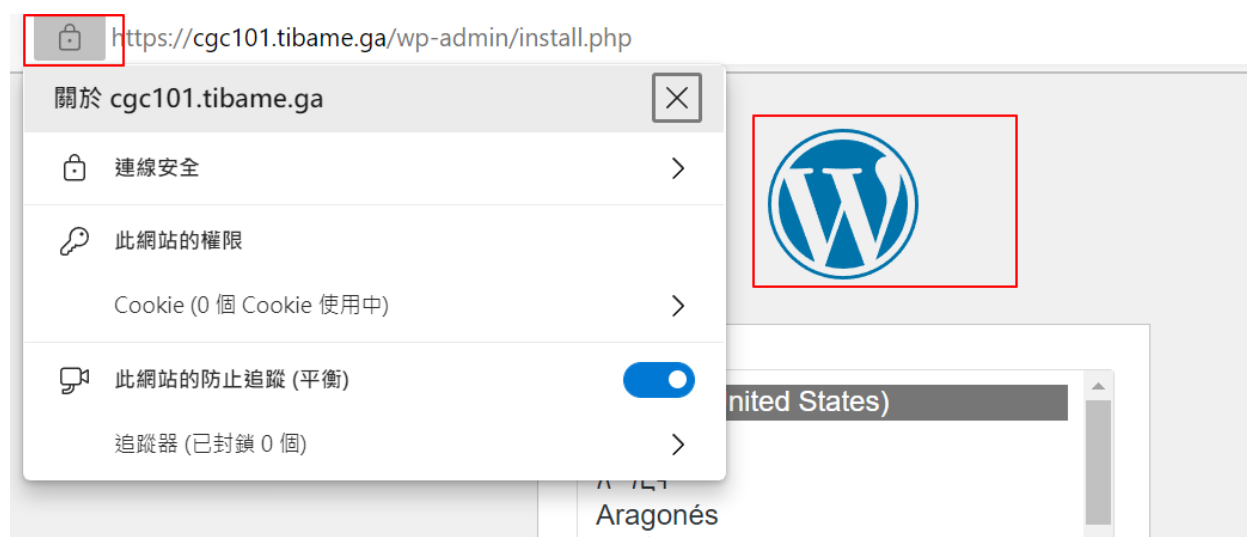
- Status狀態是Provisioning表示還在創建，此段時間會非常久，且需要串完ingress才會轉active

kubectl describe managedcertificate managed-cert

- 看到active表示創建成功

```
cgcl01.tibame.ga
Status:
Certificate Name:      mcrt-b849515b-8bc7-4cf0-86d0-3f5ebb4c2d88
Certificate Status:    Active
```

- 此時到瀏覽器輸入https://<域名>，能進入畫面表示正常



未來可以增加的功能

因為時間的關係，能夠實作的功能很有限，如果以實際上線上功能為考量，則還再增加以下功能，整體運作較為完善：

- 1.使用 Cloud Armor提升資安防護
- 2.改成使用Terraform + Ansible, 用程式實現自動化
- 3.完全自動化CI/CD
4. Cloud SQL與Firestore需啟用備份機制
- 5.滾動升級與降級的測試
- 6.AB test或金絲雀部署
- 7.RBAC與namespace等相關應用
- 8..升級Istio, 使用Service Mesh

相關參考資源:

主要資源來源:

<https://cloud.google.com/>

<https://kubernetes.io/docs/home/>

其他資料來源:

<https://dev.to/stack-labs/securing-the-connectivity-between-a-gke-application-and-a-cloud-sql-database-4d6b>

https://www.youtube.com/watch?v=xguheW_GEi4

<https://ithelp.ithome.com.tw/articles/10197046>

<https://blog.searce.com/multi-tenant-applications-on-single-google-kubernetes-engine-cluster-using-google-cloud-platform-2c36d31e2c17>

<https://stackoverflow.com/questions/71428660/how-to-solve-ingress-error-some-backend-services-are-in-unhealthy-state>