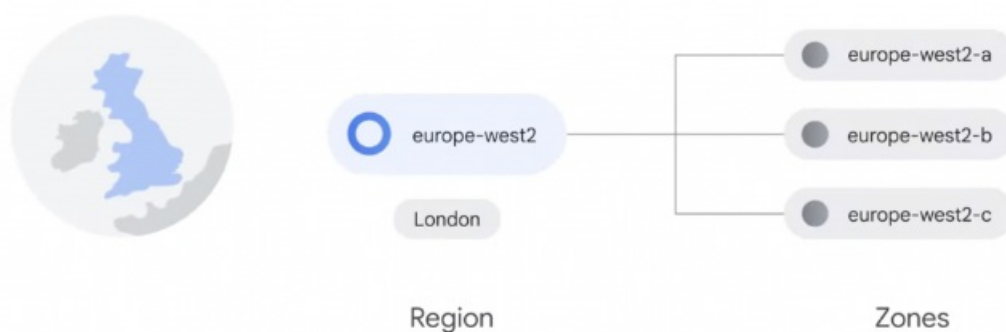


【CCP】Google Cloud Platform 服務總覽

出處：<https://ithelp.ithome.com.tw/users/20151036/ironman/6131>

Region 和 Zone 的差別

在 Google Cloud 裡面，使用了 Region 和 Zone 的概念，分別去區分不同地理區塊的服務



管理 Google Cloud 的四種方法

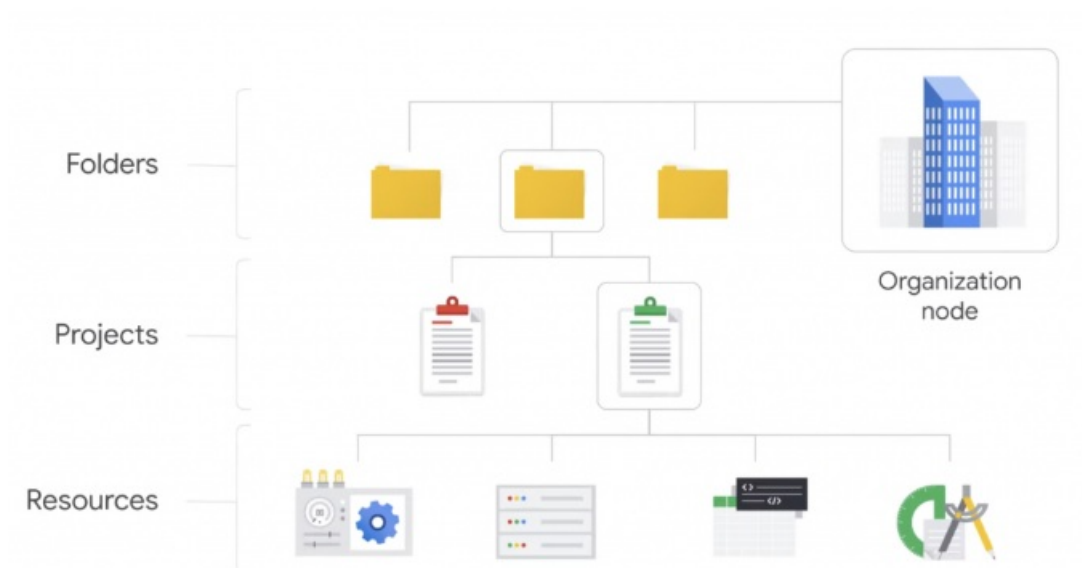
介紹完了 Region 和 Zone 的差別，接著介紹一下管理 Google Cloud 中的服務的四種方法



工具	介面類型	功能強度	使用時機
Google Cloud Console	圖形 UI	中等	建立資源、設定、報表
gcloud SDK	命令列工具	最強	自動化部署、開發整合、CI/CD
Cloud Shell	雲端終端機	高	快速部署/測試指令，不佔本地資源
Cloud APIs	程式介面	高	系統對接、自動化服務整合
Cloud Mobile App	手機 App	輕量	查看狀態、接收通知、緊急操作

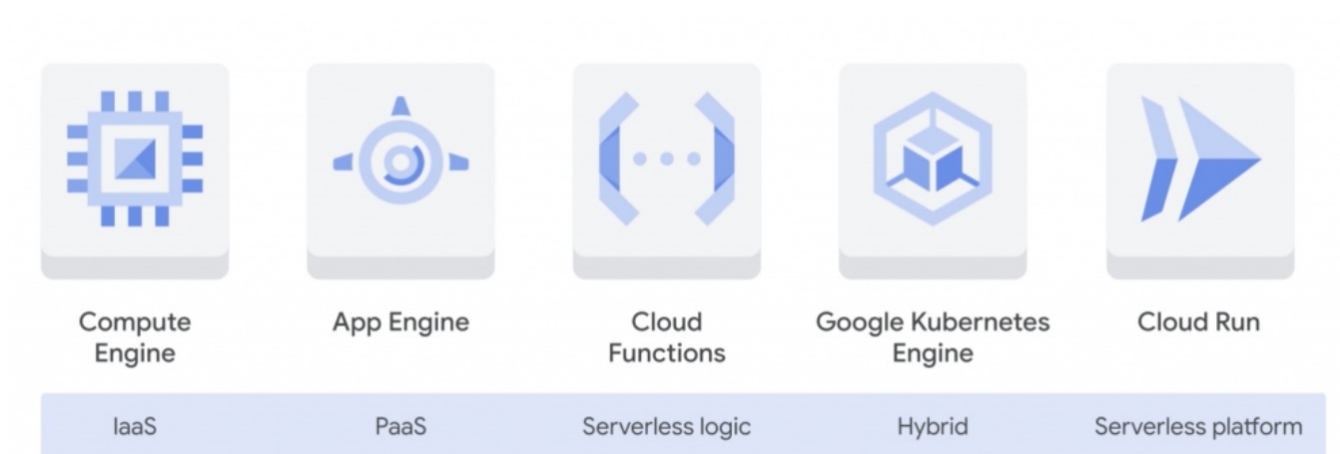
Google Cloud 中 Project 結構

下面這張圖描述了 Google Cloud 中的 Project 結構，不過這張圖要從最下層往上看



Google Cloud 中的運算服務

- Compute Engine
- App Engine
- Cloud Functions
- Google Kubernetes Engine
- Cloud Run



Google Cloud Run、App Engine、Cloud Functions 三者的完整比較：

三種 Serverless 服務比較表



項目	Cloud Run	App Engine	Cloud Functions
適用層級	容器 (Container) 層級	應用 (Application) 層級	函數 (Function) 層級
啟動方式	HTTP 請求觸發	HTTP、排程、Task Queue	HTTP、事件觸發 (如 Storage、PubSub)

項目	Cloud Run	App Engine	Cloud Functions
開發模式	自行打包容器，任何語言框架皆可	GCP 支援語言 (Java, Python, Go, Node)	限制在特定語言框架 (Node, Python, Go 等)
彈性調整	自動擴展至零	自動擴展 (可設定最小/最大執行數)	自動擴展至零 (每次執行為全新環境)
是否支援容器	可自定義容器	(App Engine Flex 支援，但較複雜)	(只支援函數級)
計費方式	請求數 + CPU + 記憶體 + 時間	請求數 + 時間 + 記憶體	函數呼叫次數 + 執行時間 + 資源
冷啟動	較低，若設定最小實例可避免	標準環境會冷啟，高級環境可控制	冷啟明顯 (尤其低頻率觸發)
狀態保持	不可保留 (Stateless)	不可保留	不可保留
部署單位	整個容器 (任意語言)	應用程式檔案 + config.yaml	單一函數檔案 (Function)
自訂網域 / HTTPS 支援	可設定自訂網域與 HTTPS		
IAM 權限控管	精細控制		

簡易選擇建議



你要的是...	建議使用
可以自由使用任何語言與架構	Cloud Run
最少設定、支援完整應用部署	App Engine
僅需要處理單一任務 / 事件驅動功能	Cloud Functions
想用 Docker 容器、可相容本地環境	Cloud Run
想部署 Flask、Express 但不想處理容器細節	App Engine
想做圖片上傳通知、排程簡訊等小任務	Cloud Functions

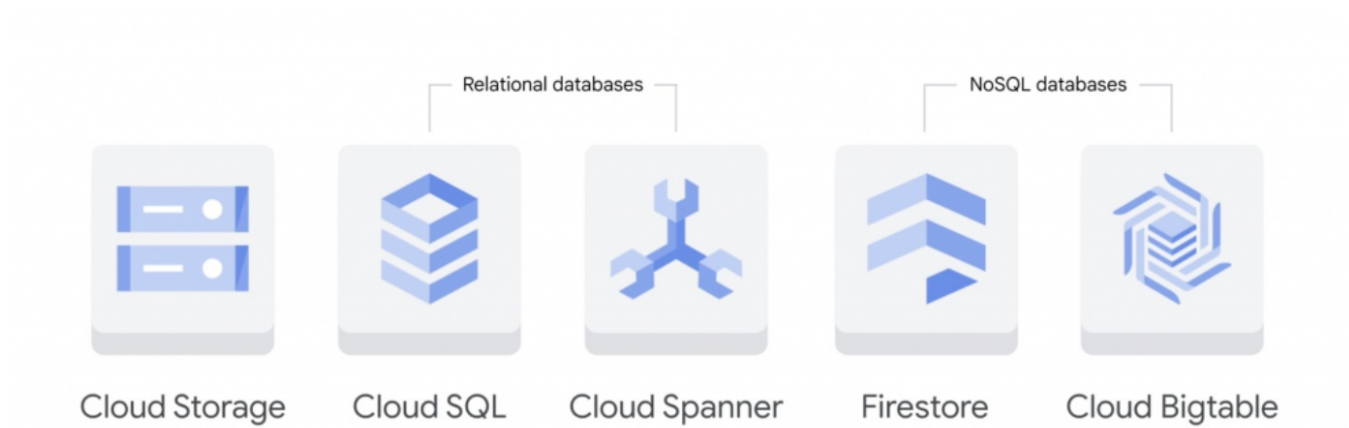
實際開發場景舉例：



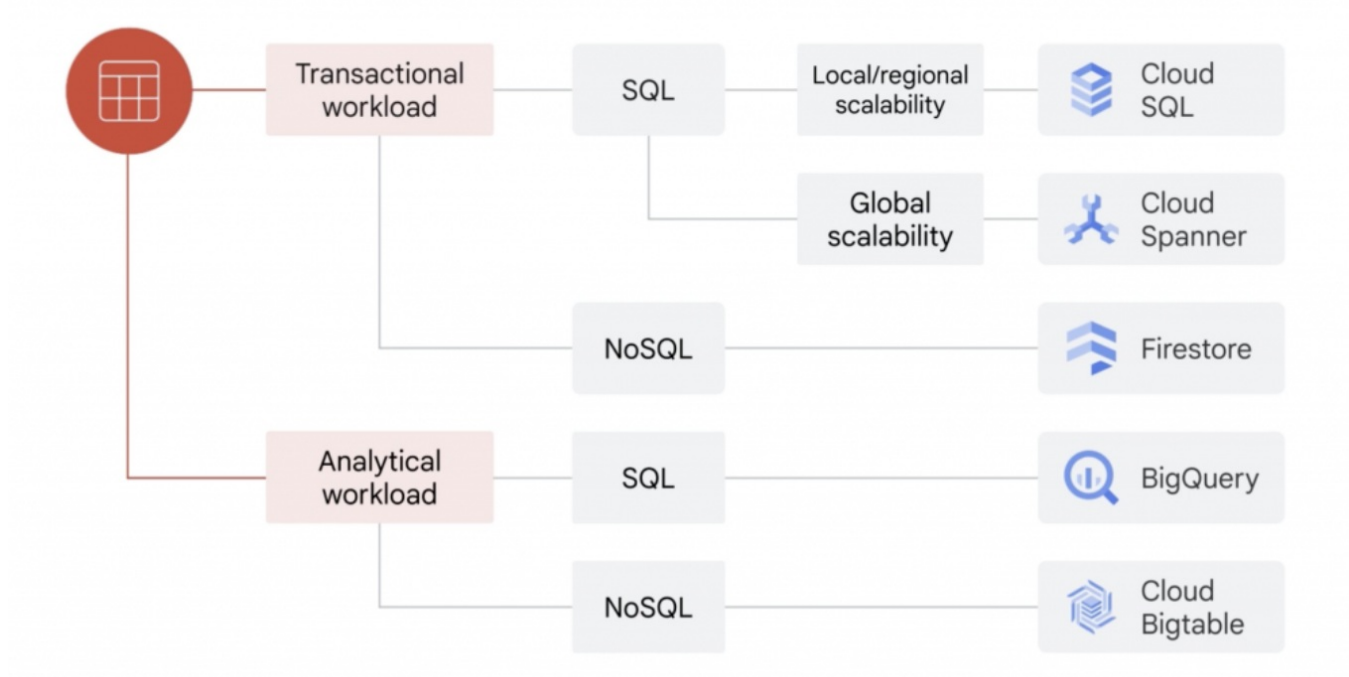
使用情境	適合服務
部署 Spring Boot 應用，並自定義 Dockerfile	Cloud Run
部署 Python Web (Flask) 快速上線	App Engine
使用 GCS 上傳觸發縮圖處理	Cloud Functions
開發 RESTful API，架構使用 FastAPI + Redis + Gunicorn	Cloud Run
設定每日 08:00 發送報表	Cloud Functions (排程)

Google Cloud 中的數據儲存服務

Google Cloud 支援五種數據的儲存服務，而這五種服務，又可以分成三大類，分別是「一般檔案的儲存服務」、「SQL 的儲存服務」以及「NoSQL 的儲存服務」



數據儲存服務總結



Bigtable 與 Bigquery 比較

基本說明



名稱	說明
Bigtable	分散式 NoSQL 資料庫，適合高頻率讀寫、低延遲的應用（類似 HBase）
BigQuery	分析型資料倉儲，專門處理超大資料集的查詢與分析（類似 Redshift、Snowflake）

詳細比較表



項目	Bigtable	BigQuery
資料類型	NoSQL，Key-Value / Wide-column store	SQL-based，結構化資料表
適用場景	即時讀寫、大量設備資料、時序資料（IoT、日誌）	大數據分析、商業智慧報表、ETL 處理

項目	Bigtable	BigQuery
□ 查詢方式	API / gRPC / HBase client	SQL (支援標準 SQL)
□ 查詢靈活度	限制多 (只能依 row key 查詢)	高 (可聚合、JOIN、Window Function)
✗ 效能焦點	寫入快、讀取延遲低	掃描大量資料效率高
□ 儲存設計	儲存在 Colossus + SStable (HBase-like 分散式設計)	儲存在 Dremel 架構 (專為資料分析優化)
□ 索引支援	□ 沒有二級索引，僅支援 row key 前綴	□ 支援複雜查詢與分析
□ 每秒處理量	百萬級 TPS (高併發)	適合大批量 (TB~PB) 分析
□ 查詢延遲	毫秒等級	秒 ~ 分鐘等級
□ 計費方式	根據儲存容量 + 節點數 (可橫向擴充)	根據掃描的資料量計價 (可選定價模式)
□ 管理需求	類似資料庫 (需設計 schema、選 row key、規劃節點)	類似報表工具 (丟資料就能查，幾乎零維運)

□ 使用情境建議



需求/情境	建議使用
IoT 裝置每秒回傳大量資料	□ Bigtable
要統計每月業績報表、跑分析 SQL	□ BigQuery
需要實時查詢某一筆用戶記錄	□ Bigtable
想跑 SELECT count, avg, group by	□ BigQuery
需要和 Data Studio 或 Looker 整合報表	□ BigQuery
想儲存百億筆資料但需低延遲查詢	□ Bigtable

□ 搭配使用範例

很多系統會這樣搭配：

1. **Bigtable：實時寫入**
 - 寫入裝置或行為記錄 (如瀏覽行為、IoT 資料)
2. **定期匯入到 BigQuery**
 - 跑 ETL，進行分析與報表產出

Google Cloud 中的 API 管理服務

影片中有提到三種 Google Cloud 中常見的 API 管理服務，分別是：

- Apigee API Management
- API Gateway
- Cloud Endpoints

□ 快速對比表：Apigee vs API Gateway vs Cloud Endpoints



項目	Apigee	API Gateway	Cloud Endpoints
□ 定位	企業級 API 管理平台	簡化、輕量的 GCP 原生 API Gateway	開發者導向的輕量級 API Proxy
□ 建立方式	使用 Apigee UI 建立 API Proxy	部署 <code>api-gateway.yaml</code> + GCP service	使用 ESP (Extensible Service Proxy) 與 OpenAPI 配置
□ 功能完整性	□ 完整 API 生命週期管理 (分析、流量管控、金鑰等)	△ 僅基本 API Proxy 與驗證	△ 基本功能 + Firebase / Auth0 整合

項目	Apigee	API Gateway	Cloud Endpoints
□ 分析與報表	□ 內建精細分析報表	□ 無分析功能	△ 可搭配 Stackdriver/Cloud Logging
□ 驗證/授權支援	□ OAuth2, API Key, JWT, SAML, 多種認證 ◀ <input type="text"/> ▶	□ JWT / API Key / IAM	□ JWT / API Key / Firebase
⚙ API 開發方式	提供 GUI、Flow 編輯器、邏輯處理	使用 OpenAPI 定義 + 設定 YAML	OpenAPI / gRPC 定義 (ESP Proxy)
□ 效能	高 (企業級 SLA)	中等, 適合輕量流量	輕量, 設計給內部/開發測試用
□ 收費模式	依照 API 呼叫數、功能計費 (需額外購買)	依流量計費 (依照 GCP 定價)	依照 ESP Proxy 的 GKE/GCE/Cloud Run 運行費用計費
□ 運行平台	Apigee X (GCP原生)、Hybrid	Cloud Run / GKE / GCE	GKE / Cloud Run / App Engine
□ 複雜功能 (限流、轉換)	□ 支援 (flow policy 可自定義 JSON/XML 轉換等)	□ 不支援	□ 不支援
□ 部署簡易性	中等 (企業級 GUI + 設定需學習)	高 (YAML + CLI)	中等 (需熟 ESP/OpenAPI)

□ 應用場景建議

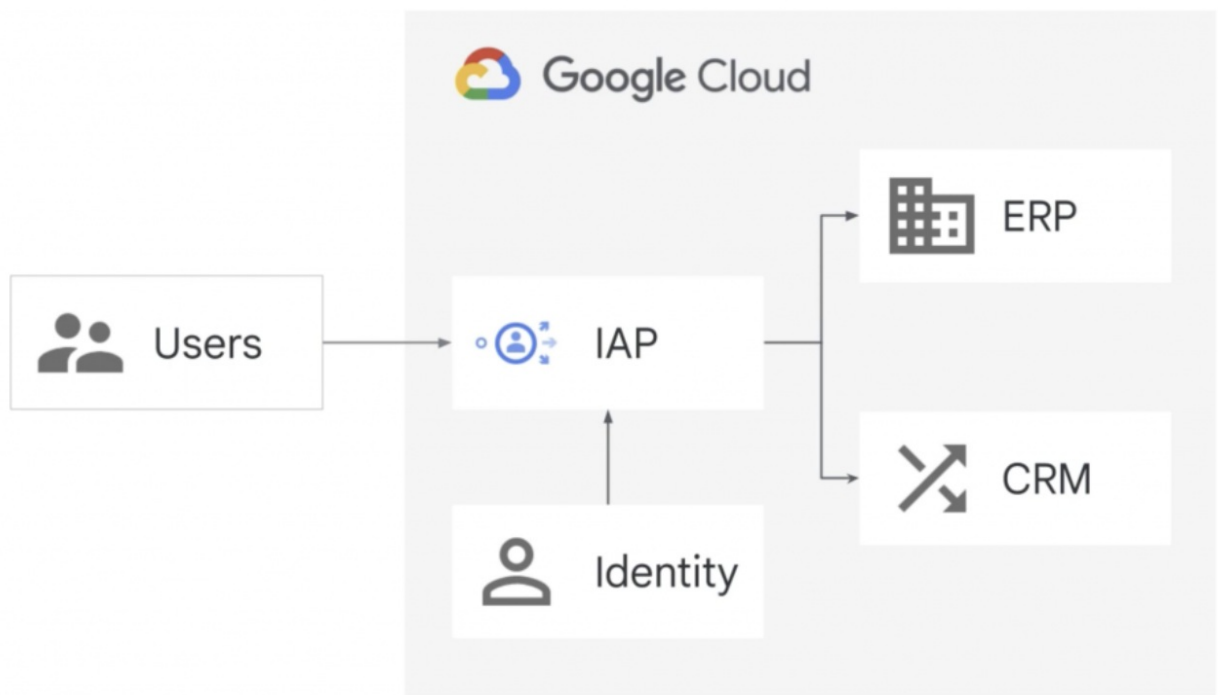


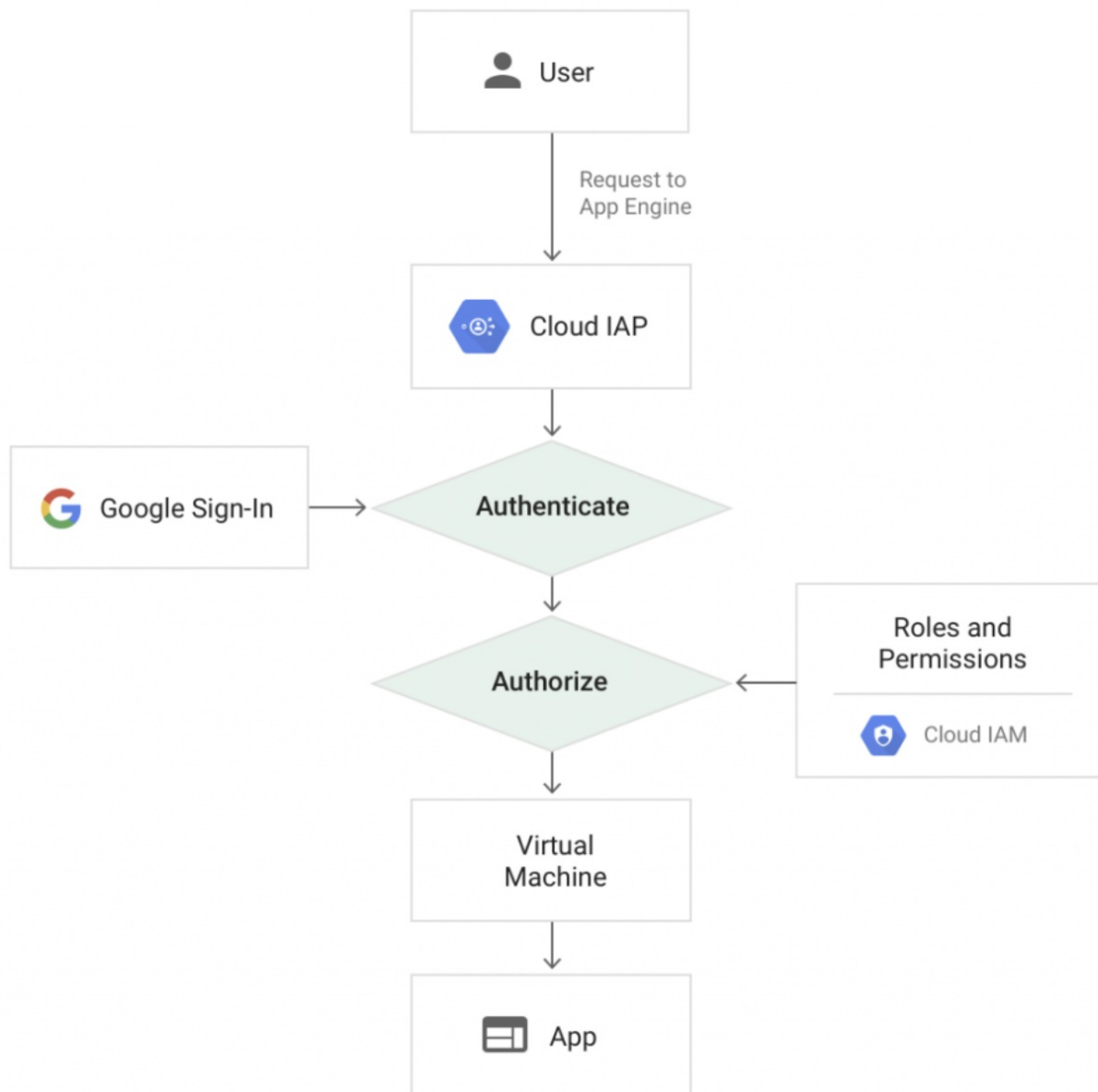
需求 / 狀況	建議使用
企業需要完整 API 管理功能 (流控、報表等)	□ Apigee
GCP 環境輕量部署 + IAM 驗證 API	□ API Gateway
開發階段/原型開發 + 自訂認證方式	□ Cloud Endpoints
需要整合 Firebase 認證	□ Cloud Endpoints
多租戶 API 管理、API 商業化 (API Monetization)	□ Apigee
想快速部署在 Cloud Run + IAM 控制	□ API Gateway

□ 技術選擇心法：

- □ 選 **Apigee** 如果你是企業客戶、要做 API 授權金鑰控管、頻率限制、報表、開發者入口網站 (dev portal) 等。
- □ 選 **API Gateway** 如果你只是想快速幫 Cloud Run 上的 API 加上 HTTPS、安全性。
- □ 選 **Cloud Endpoints** 如果你熟 OpenAPI、只想讓 API 有認證與基本日誌記錄，不需要 GUI。

權限管理服務





說明 Iaas Paas Saas Faas

這四個名詞是雲端服務的不同層級與服務模型，分別為：

□ IaaS (Infrastructure as a Service) 基礎設施即服務

- **提供內容：** 虛擬機 (VM)、儲存空間、網路、作業系統等基礎資源
- **使用者負責：** 自行安裝應用程式、中介軟體、設定安全性等
- **彈性高，自由度大**
- **常見例子：**
 - AWS EC2
 - Google Compute Engine
 - Microsoft Azure VM

“ □ 適用於：需要完整控制權限的系統管理員與 DevOps 團隊

☐ PaaS (Platform as a Service) 平台即服務

- **提供內容：** 開發平台 (含OS、Runtime、DB、Framework)
- **使用者負責：** 上傳應用程式，開發、部署、維護應用邏輯
- **省去基礎建設與系統維護負擔**
- **常見例子：**
 - Google App Engine
 - Heroku
 - AWS Elastic Beanstalk

“☐ 適用於：開發人員專注寫程式，不想管底層架構

☐ SaaS (Software as a Service) 軟體即服務

- **提供內容：** 完整的軟體應用，通常透過瀏覽器存取
- **使用者負責：** 只需要使用，不用管安裝、更新、維護
- **完全託管、快速上手**
- **常見例子：**
 - Gmail / Outlook
 - Google Docs / Office 365
 - Salesforce
 - Dropbox

“☐ 適用於：終端使用者或企業快速取得現成解決方案

✂ FaaS (Function as a Service) 函數即服務

- **提供內容：** 運行特定函數或任務的執行環境
- **使用者負責：** 撰寫函數邏輯，其他都交給雲端處理 (包含資源擴展)
- **屬於 Serverless 架構的一種**
- **常見例子：**
 - AWS Lambda
 - Google Cloud Functions
 - Azure Functions

“☐ 適用於：事件驅動的任務 (如：圖檔轉換、資料處理、API 回應等)

對照圖表總覽：



層級	你負責的部分	例子
IaaS	App + OS + Middleware + DB	AWS EC2、GCP VM
PaaS	App + DB	Heroku、GCP App Engine
SaaS	僅使用	Gmail、Google Docs
FaaS	僅寫 function	AWS Lambda、GCF

說明 IaaS PaaS SaaS FaaS

★由 treeman 建立於 25 2025 16:07:55

✎由 treeman 更新於 25 2025 17:50:44