# Google Antigravity 實作實驗室：Agent-First 雲原生部落格部署指南

本手冊旨在指導使用者利用 **Google Antigravity** 的 Agent 能力，在不手寫大量設定檔的情況下，快速重現「雲端原生部落格終極藍圖」架構。我們將從零打造一個基於 Hugo、Docker (Chainguard)、Terraform 與 Cloud Run 的高可用性部落格系統。

## 📋 目錄



## 1. 專案簡介與架構

本實驗室將透過 Antigravity Agent 協調以下技術堆疊，實現 100% 代碼管理與自動化部署：

- 應用層：Hugo (靜態網站生成器) + Congo Theme (Tailwind CSS)
- 容器層：Multi-stage Build + Chainguard Nginx Images (最小權限安全性)
- 基礎設施層：Terraform (IaC) 管理 Cloud Run, Artifact Registry, Load Balancer, CDN
- 開發流程：Agent 驅動的本地熱重載 (Hot-reload) 與 GitOps 自動化

## 2. 先決條件

在開始之前，請確保您已具備以下條件：

- **Google Antigravity IDE**：已安裝並更新至最新版本。
- **Google Cloud Platform (GCP)** 帳號：
  - 擁有一個已啟用帳單 (Billing) 的專案。
  - 擁有專案的 Owner 或 Editor 權限。
- **Antigravity** 設定：
  - 已切換至 **Enterprise Mode**。
  - 已連結目標 GCP 專案。

## 階段 1：環境初始化與專案建置

目標：讓 Agent 準備開發環境並生成 Hugo 網站骨架。

### 操作步驟

1. 開啟 **Antigravity**。
2. 按下 Cmd+I (macOS) 或 Ctrl+I (Windows/Linux) 開啟 **Manager View (Agent Chat)**。
3. 環境連線：複製並貼上以下 Prompt："請確認我目前連接的 GCP Project ID。接著，請幫我檢查並啟用以下 API：Cloud Run, Artifact Registry, Cloud Build, Compute Engine (for Load Balancer)。如果未啟用，請幫我啟用它們。"
4. 等待 Agent 確認 API 啟用完成。
5. 建立 **Hugo** 站點：複製並貼上以下 Prompt："我需要建立一個新的 Hugo 部落格。請在當前目錄下執行 hugo new site my-blog --format yaml。接著，請幫我下載 'Congo' theme (基於 Tailwind CSS) 並將其設定為預設主題。最後初始化一個 git repository。"

### ✅ 檢核點

- 檔案瀏覽器中應出現 my-blog 資料夾。
- my-blog/themes/congo 資料夾存在。
- .git 資料夾已建立。

## 階段 2：容器化與本地預覽

目標：生成符合安全最佳實踐的 Dockerfile 並進行本地測試，取代傳統的 Cloud Code Emulator。

### 操作步驟

1. 生成 **Dockerfile**：複製並貼上以下 Prompt："請幫我為這個 Hugo 網站撰寫一個 Dockerfile。請遵循以下最佳實踐：
    1. 使用 **Multi-stage build**：第一階段用 hugo image 生成靜態檔案。
    2. 安全性：第二階段請使用 **Chainguard Nginx image** (cgr.dev/chainguard/nginx:latest) 來提供服務，以最小化攻擊面。
    3. 確保 Nginx 聽取 $PORT 環境變數 (Cloud Run 需求)。"
2. 代碼審查 **(Review)**：
    - 點擊對話框中的 Review Changes。
    - 確認 FROM 指令使用了 cgr.dev/... 映像檔。
    - 點擊 Accept 套用變更。
3. 本地預覽：複製並貼上以下 Prompt："現在我想在本地預覽網站。請執行 hugo server -D 並開啟瀏覽器預覽。"

### ✅ 檢核點

- IDE 內建瀏覽器或 Chrome 應自動彈出。
- 網址為 localhost:1313，畫面顯示 Congo 主題的預設頁面。

## 階段 3：基礎設施即代碼 (Terraform 生成)

目標：由 Agent 生成定義 Cloud Run、Load Balancer 與 CDN 的 Terraform 代碼。

### 操作步驟

1. 定義基礎設施：複製並貼上以下 Prompt："請在專案根目錄建立一個 infra/ 資料夾，並生成 Terraform 設定檔 (main.tf, variables.tf, services.tf) 以部署以下架構：
    1. 一個 **Artifact Registry** Repository (Docker format)。
    2. 一個 **Cloud Run Service**，部署我們剛寫好的 Container。
    3. 設定 **Global External HTTPS Load Balancer** 並啟用 **Cloud CDN**。
    4. 使用 **GCS Backend** 來儲存 Terraform State 以支援團隊協作。"
2. 執行部署：複製並貼上以下 Prompt："請先執行 terraform init。如果 GCS bucket 不存在，請引導我創建它。接著執行 terraform plan 讓我確認資源，確認無誤後執行 terraform apply。"

### 注意事項

- Agent 可能會詢問您要設定的 GCS Bucket 名稱（必須全球唯一）。
- terraform apply 可能需要幾分鐘時間，請耐心等待 Agent 回報 Apply complete。

### ✅ 檢核點

- GCP Console 中出現了新的 Cloud Run Service。
- Artifact Registry 中建立了新的 Repository。

## 階段 4：CI/CD 自動化管線

目標：設定 GitOps 流程，讓 Git Push 自動觸發 Cloud Build 進行部署。

### 操作步驟

1. 建立 **Cloud Build** 設定：複製並貼上以下 Prompt："請建立一個 cloudbuild.yaml 檔案。步驟應包含：
    1. Build Docker image。
    2. Push image 到我們剛建立的 Artifact Registry。
    3. 使用 terraform apply 或 gcloud run deploy 更新 Cloud Run service。"
2. 設定觸發器 **(Trigger)**：複製並貼上以下 Prompt："利用 gcloud 指令，幫我設定一個 Cloud Build Trigger。當我 push 代码到 main 分支時，自動執行這個 build。(請假設我已經將代碼推送到 GitHub/GitLab 並連結了 Repository)。"

### ✅ 檢核點

- 專案根目錄出現 cloudbuild.yaml。
- GCP Cloud Build 頁面中出現新的 Trigger。

## 階段 5：驗證與體驗

目標：確認全域部署成功，並體驗「熱重載」開發流程。

### 操作步驟

1. 取得公開網址：「請給我目前 Load Balancer 的公開 IP 或 Cloud Run 的 URL。」
2. 瀏覽器驗證：打開該網址，確認網站已上線且 CDN 運作中。
3. 體驗熱重載 **(Hot-reload)**：
    - 在 Antigravity 編輯器中，開啟 content/posts/hello.md (或建立一個新貼文)。
    - 修改標題為 "Hello Antigravity!" 並存檔。
    - 觀察本地預覽視窗（localhost:1313）是否即時更新。
4. 體驗 **GitOps**：
    - 執行 Git Commit & Push。
    - 前往 Cloud Build Console 觀察 Build 是否自動開始。

# 8. 清理資源

為避免產生不必要的 GCP 費用，實驗結束後請務必清理資源。

1. 執行銷毀指令：「請執行 terraform destroy 移除所有 GCP 資源。完成後，請刪除 Terraform State 使用的 GCS Bucket。」
2. 確認：輸入 yes 確認銷毀操作。

文件版本*: 1.0* | 適用於 *Google Antigravity Enterprise*