【AWS】Day 7：運算服務 - Elastic Beanstalk（自動化部署服務）

在前幾篇，我們介紹了 EC2，了解了怎麼自己開一台雲端主機、自己設定環境、自己管防火牆、自己擴展、自己備份。

但如果你覺得自己管這些基礎建設很麻煩，希望可以更專注在「寫程式」跟「部署應用程式」上面，那 AWS 其實也有更方便的選擇：
今天要介紹的就是 —— **Elastic Beanstalk**

什麼是 Elastic Beanstalk？

Elastic Beanstalk 是 AWS 提供的一種 **PaaS（平台即服務）**。

它的目標很簡單：

- 你只要準備好應用程式
- 把應用程式丟給 Beanstalk
- Beanstalk 幫你處理所有底層繁瑣的事情

包括：

- 建好 EC2 instance
- 建好負載平衡器（Load Balancer）
- 建好 Auto Scaling
- 建好 Security Group、防火牆設定
- 幫你佈署、監控、收集日誌
- 還可以一鍵回滾版本

你專注在開發跟部署，AWS 幫你顧好其他基礎設施。

支援的應用類型

Elastic Beanstalk 支援超多種語言與平台：

- Node.js
- Python
- Java
- Ruby
- PHP
- .NET
- Go
- Docker（完整支援 Container）
- 靜態網站（HTML/CSS/JS）

如果你是 Web 應用開發者，基本上都可以直接使用 Beanstalk 部署。

Elastic Beanstalk 的架構概念

雖然 Beanstalk 幫你管理底層資源，但其實背後還是會幫你建立 AWS 其他服務，包括：

- EC2（跑你的應用）
- Elastic Load Balancer（分流）
- Auto Scaling Group（自動擴展）
- CloudWatch（監控、日誌）
- RDS（如果有需要資料庫）

這些資源都是 Beanstalk 幫你「托管」，你可以選擇要不要深入調整。

> 你仍然可以進入 EC2 頁面看到自己應用用到的實例，但建議不要直接手動修改，避免破壞 Beanstalk 的管理邏輯。

建立 Elastic Beanstalk 應用的流程

1. 進入 AWS Console，打開 Elastic Beanstalk
2. 點選「Create Application」
3. 填入：
   - Application Name（應用名稱）
   - Platform（選語言與環境）
   - Upload your code（上傳程式碼壓縮檔）
4. 設定部署設定（可以預設）
5. 點下「Create Application」

幾分鐘後，AWS 就會幫你建好整個環境，還會提供一個 URL，直接可以訪問你的應用！

部署更新與版本管理

Elastic Beanstalk 很貼心地內建了「版本管理」功能：

- 每次部署新程式碼，都會建立一個新的版本
- 可以快速回滾（Rollback）到前一個版本
- 可以在不同環境（例如 dev、prod）快速切換程式版本

常見 Elastic Beanstalk 的使用情境

- 快速啟動 MVP、小型專案
- 開發階段測試不同版本部署
- 中小型團隊的 Web 應用架設
- 需要簡單 Auto Scaling、Load Balancing 但不想自己架設

注意事項

- **不是完全免維運**：雖然簡化很多，但還是要稍微理解 VPC、Security Group 等 AWS 基本概念
- **資源會產生費用**：Beanstalk 本身不收費，但底層建立的 EC2、RDS、S3 等都會收費
- **適合標準架構**：如果應用有非常特殊的環境需求（例如自訂作業系統 kernel）可能還是要回去用 EC2 自己建

小結

Elastic Beanstalk 是 AWS 上手最簡單的運算服務之一：

- 幫你管理底層基礎建設
- 讓你快速部署、快速擴展、快速回滾
- 適合中小型 Web 應用，也適合剛開始學 AWS 的人嘗試

當然可以！這裡直接幫你補上
Elastic Beanstalk 簡易實作教學（以 Node.js 舉例），延續前面文章的語氣與結構，實作內容也保持簡單明瞭，讓讀者可以直接跟著做。

Elastic Beanstalk 簡易實作教學（以

Node.js 範例）

剛剛介紹了 Elastic Beanstalk 是什麼，接下來就實際帶大家操作一次：

- 我們會快速做出一個 Node.js 小應用
- 壓成 ZIP 檔
- 部署到 Elastic Beanstalk 上

目標是：**10 分鐘內完成一個雲端 Web App！**

---

1. 建立一個簡單的 Node.js 專案

首先，在你的電腦上開一個新資料夾，例如 `eb-demo-app`，然後建立最基本的 Node.js 應用。

```
mkdir eb-demo-app
cd eb-demo-app
npm init -y
npm install express
```

接著建立一個 `app.js`，內容如下：

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;

app.get('/', (req, res) => {
  res.send('Hello from Elastic Beanstalk!');
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

這樣就完成了最基本的 Node.js Web 應用。

注意：這裡我們用 `process.env.PORT` 是 Elastic Beanstalk 的部署要求，讓它自動分配正確的 port。

---

2. 建立必要的設定檔

Elastic Beanstalk 在 Node.js 環境中，需要有一個叫做 `package.json` 的設定檔（剛剛 `npm init` 已經自動生成），還需要明確標出啟動指令。

請確認 `package.json` 中有這一段：

```
"scripts": {
  "start": "node app.js"
}
```

如果沒有，手動加上去，這樣 Elastic Beanstalk 才知道要怎麼啟動你的應用程式。

---

3. 壓縮成 ZIP 檔案

回到 `eb-demo-app` 目錄，將所有檔案壓縮成一個 `.zip` 檔。

注意：**不要把整個資料夾壓縮進去！**

- ZIP 檔裡面應該是直接包含 `app.js`、`package.json`，不是包一層資料夾！

例如正確的內容：

```
app.js
package.json
node_modules/
```

如果你想省空間，其實可以只壓 `app.js` 和 `package.json`，讓 Elastic Beanstalk 自己在雲端跑 `npm install`。

---

4. 登入 AWS Elastic Beanstalk 控制台

打開 Elastic Beanstalk 主控台

1. 點選「Create Application」
2. 填上：
   - **Application name**：例如 `demo-node-app`
   - **Platform**：選 `Node.js`
   - **Application code**：選擇「Upload your code」上傳剛剛的 `.zip` 檔
3. 點選「Create Application」

接下來 AWS 會自動：

- 建 EC2
- 建 Load Balancer
- 建 Auto Scaling Group
- 佈署你的應用程式

大概 3-5 分鐘後，就可以從 Elastic Beanstalk 頁面拿到一個網址，直接訪問你的應用！

---

5. 驗證部署結果

打開 Elastic Beanstalk 頁面給你的 URL，比如：

```
http://demo-node-app.ap-northeast-1.elasticbeanstalk.com/
```

你應該會看到網頁上寫著：

```
Hello from Elastic Beanstalk!
```

部署成功 ！

小結

這就是最簡單的 Elastic Beanstalk 實作流程，總結一下步驟：

1. 建立應用（Node.js / Python / etc）
2. 加好 `start` 指令
3. 壓成 ZIP 檔
4. 上傳到 Elastic Beanstalk
5. 幾分鐘內自動建立並上線

對於想快速上線 MVP 或測試環境的人來說，Elastic Beanstalk 真的是一個超省力的好選擇！

沒問題！這裡幫你補上
Elastic Beanstalk 簡易實作教學（Python / Flask 範例）
延續剛剛 Node.js 範例的風格和節奏，一樣清楚、快速能跟著做！

Elastic Beanstalk 簡易實作教學（以

Python / Flask 範例）

如果你是使用 **Python** 的開發者，想要快速把 Flask App 部署到 Elastic Beanstalk，也是非常簡單的！

這篇教學會帶你從零建立一個最簡單的 Flask App，並且成功部署到 Elastic Beanstalk 上運行。

---

1. 建立一個 Flask 專案

首先，在你的電腦上新建一個資料夾，例如 `eb-python-demo`：

```
mkdir eb-python-demo
cd eb-python-demo
python3 -m venv venv
source venv/bin/activate  # Windows 用 venv\Scripts\activate
pip install flask
```

接著新增一個 `application.py`（注意：**檔名是 `application.py`，Elastic Beanstalk 預設會找這個名字**）

```
from flask import Flask

application = Flask(__name__)

@application.route("/")
def hello():
    return "Hello from Elastic Beanstalk with Python!"

if __name__ == "__main__":
    application.run(host='0.0.0.0', port=5000)
```

這樣就完成一個超簡單的 Flask Web App！

---

2. 建立 requirements.txt

Elastic Beanstalk 需要知道要安裝哪些套件，所以要建立一個 `requirements.txt` 檔案，內容如下：

```
Flask
```

（未來如果有更多套件，就加在這個檔案裡。）

指令快速產生：

```
pip freeze > requirements.txt
```

---

3. 建立 `.ebextensions` 設定檔（可選）

如果你想在 Elastic Beanstalk 上做更多進階設定（例如自動安裝系統套件），可以用 `.ebextensions`。
不過這次我們簡單部署，這步可以先跳過。

---

4. 壓縮成 ZIP 檔案

跟剛剛 Node.js 一樣，把你的檔案壓縮成一個 `.zip` 檔案。

⚡ **注意：**

- ZIP 裡面應該包含：`application.py`、`requirements.txt`
- 不要壓整個資料夾進去

正確的結構是：

```
application.py
requirements.txt
```

---

5. 部署到 Elastic Beanstalk

打開 AWS Elastic Beanstalk Console

操作步驟：

1. 點選「Create Application」
2. 填寫：
   - **Application name**：例如 `demo-python-app`
   - **Platform**：選 **Python**
   - **Platform Branch**：例如 Python 3.11
3. 上傳剛剛壓好的 `.zip` 檔案
4. 點選「Create Application」

Elastic Beanstalk 會自動：

- 幫你建立 EC2
- 幫你佈署 Flask
- 幫你設定 Load Balancer

大約 3-5 分鐘部署完成。

---

6. 驗證部署結果

部署好之後，Elastic Beanstalk 會給你一個 URL，比如：

```
http://demo-python-app.ap-northeast-1.elasticbeanstalk.com/
```

打開之後應該會看到：

```
Hello from Elastic Beanstalk with Python!
```

大成功！

小結

Python / Flask 的 Elastic Beanstalk 快速部署流程：

1. 建好 Flask 應用（注意 `application.py`）
2. 建立 `requirements.txt`
3. 打包成 `.zip`
4. 上傳到 Elastic Beanstalk
5. 等待幾分鐘，直接有一個可以訪問的 Web App

Elastic Beanstalk 進階教學

（環境變數、eb CLI 部署、自動串接 RDS）

一、設定環境變數（Environment Variables）

在實際應用中，很常需要讓應用程式讀取一些敏感資訊（像資料庫連線字串、API 金鑰等等），這些不適合硬寫在程式碼裡。
這時就需要使用「環境變數」。

在 Elastic Beanstalk 上設定環境變數

操作步驟：

1. 打開 Elastic Beanstalk Console，進入你的應用 Environment
2. 左邊選單找到「Configuration」
3. 找到「Software」，按下「Edit」
4. 在 Environment properties 區塊新增你的變數，例如：

Key	Value
`DB_HOST`	mydb.xxxxxxxxx.rds.amazonaws.com
`DB_USER`	admin
`DB_PASSWORD`	supersecure123

儲存後，Elastic Beanstalk 會自動重啟環境，讓你的應用可以透過環境變數讀取設定。

在程式中讀取環境變數（Python Flask 範例）

```
import os

db_host = os.getenv('DB_HOST')
db_user = os.getenv('DB_USER')
db_password = os.getenv('DB_PASSWORD')
```

這樣就能安全地使用環境設定，不用把敏感資訊寫死在程式碼裡！

二、使用 eb CLI 進行快速部署

如果你覺得每次都開 Console 點來點去很煩，AWS 提供了專門給 Elastic Beanstalk 用的指令工具：
Elastic Beanstalk CLI (eb CLI)

可以直接從 Terminal 完成初始化、部署、查看環境狀態等等，非常方便！

安裝 eb CLI

先安裝 AWS EB CLI：

```
pip install awsebcli --upgrade
```

> 有時候 MacOS 或 Linux 需要加上 `--user`，或用 virtualenv 安裝。

安裝好後，試試：

```
eb --version
```

如果能正常顯示版本號，代表安裝成功。

□ eb CLI 操作基本流程

1. 登入 AWS（要先設定好 AWS CLI credentials）

```
aws configure
```

2. 初始化 eb 專案

在你的應用資料夾內：

```
eb init
```

過程中會問你：

- 要使用哪個 Region？
- 是哪個 Platform？（Node.js、Python 等）
- 是否需要設定 SSH？

3. 建立 Environment

```
eb create dev-env
```

（可以自己取環境名字）

4. 部署更新

```
eb deploy
```

5. 查看環境資訊

```
eb status
```

6. 開啟瀏覽器查看應用

```
eb open
```

超級方便，可以完全不開網頁直接部署更新！

三、自動串接 RDS 資料庫（一起建立）

如果你的應用需要資料庫（例如 MySQL、PostgreSQL），Elastic Beanstalk 可以在建立 Environment 時**順便幫你建立 RDS**！

□ 如何設定

1. 建立 Environment 時，選擇「Configure more options」
2. 找到「Database」選項，點選「Edit」
3. 選擇：
   - 資料庫引擎（MySQL、PostgreSQL）
   - 資料庫版本
   - 使用者名稱、密碼
   - 硬碟空間大小

建立好後：

- Elastic Beanstalk 會自動把 RDS 與你的應用放在同一個 VPC、Security Group
- 會自動把資料庫連線資訊存到環境變數（`RDS_HOSTNAME`、`RDS_USERNAME`、`RDS_PASSWORD`、`RDS_PORT`）

你的 Flask / Node.js 應用只要讀取這些環境變數，就能連上資料庫！

□ Flask 範例（連接 Elastic Beanstalk RDS）

```
import os
import pymysql
```

```
db_host = os.getenv('RDS_HOSTNAME')
db_user = os.getenv('RDS_USERNAME')
db_password = os.getenv('RDS_PASSWORD')
db_name = os.getenv('RDS_DB_NAME')

conn = pymysql.connect(
    host=db_host,
    user=db_user,
    password=db_password,
    database=db_name
)
```

就可以在應用程式中直接操作資料庫啦！

小結

Elastic Beanstalk 除了快速部署應用，進階玩法還可以：

- **設定環境變數，安全管理敏感資訊**
- **使用 eb CLI，極速 Terminal 部署**
- **一起建立 RDS，讓應用直接串接雲端資料庫**

Elastic Beanstalk 常見錯誤排除指南

Elastic Beanstalk 幫我們自動化了很多事，但實際使用過程中還是很容易遇到一些常見錯誤。

這篇幫大家整理了：

- 常見的 Elastic Beanstalk 部署問題
- 原因說明
- 解決方法

給你一份遇到問題時能快速排雷的工具包！

---

1. 部署後出現 502 Bad Gateway

問題現象

- **部署成功，但一打開應用的網址就跳出 502 Bad Gateway**
- **或者頁面載入超級慢然後 timeout**

常見原因

- **應用程式沒有正常啟動**
- **聽錯了 Port**（Elastic Beanstalk 預設會轉送到 80 或由 proxy pass）

解決方法

 檢查你的應用程式是否有監聽正確的 Port：

- Node.js / Flask / Django 等應用，要監聽環境變數給定的 port
- 例如 Flask：

python

複製 **編輯**

```
application.run(host='0.0.0.0', port=int(os.environ.get('PORT', 5000)))
```

 查看 Logs：

- Elastic Beanstalk Console → Logs → Request logs → Last 100 lines
- 查看應用啟動過程是否有錯誤訊息，例如 module not found、port 綁定錯誤等

---

2. 部署失敗，顯示「Instance deployment failed」

問題現象

- 上傳 ZIP 包後，環境 Health 直接變紅色
- console 顯示 deployment failed

常見原因

- 上傳的 ZIP 包結構錯誤
- 必要檔案（像 `application.py` 或 `package.json`）缺少或錯誤
- requirements.txt 有誤（例如版本衝突、缺失）

解決方法

 確認 ZIP 檔內部是正確的結構：

複製 **編輯**

`application.py`
requirements.txt
`（不要包一層資料夾）`

 如果是 Python，確保有 `requirements.txt`，且內容正確。

 部署失敗可以看 Instance logs：

- Console → Logs → Instance logs → 全部下載下來
- 查看 `/var/log/eb-engine.log` 或 `/var/log/web.stdout.log` 裡的錯誤訊息

---

3. 無法 SSH 連線到 EC2

問題現象

- 按下「Connect」或者手動 SSH 連線失敗
- 卡在 timeout，無法進去 Instance

常見原因

- Security Group 沒有開放 port 22
- 沒有設定 Key Pair
- 錯誤使用 Public IP 或 DNS 名稱

解決方法

 確認 Environment → Configuration → Security → EC2 Key Pair 有設定。

 Security Group 要開 TCP 22 port，允許你的 IP 連進去。

 連線指令正確，例如：

bash

複製 編輯

```
ssh -i my-key.pem ec2-user@your-public-dns.amazonaws.com
```

---

4. 資源遺留：環境刪掉但 EC2、RDS 還在

問題現象

- 刪除 Elastic Beanstalk 環境後
- 發現 EC2、RDS、Security Group、EBS Volume 還殘留在帳號中

常見原因

- 預設刪除環境時，只會刪除 Beanstalk 直接管理的資源
- 外掛式 RDS、手動建立的資源不會跟著刪

解決方法

 手動到 EC2、RDS、VPC 頁面，檢查並刪除殘留的資源。

 未來建立環境時，如果有開 RDS，**選「內嵌式 RDS」**（跟環境生命週期綁在一起），才會自動清掉。

---

5. 更新應用程式後，環境變紅，回不去

問題現象

- 部署新版本應用後
- Elastic Beanstalk 環境變成 Severe（紅色）
- 想緊急回滾但不知道怎麼做

常見原因

- 新版程式出錯，導致服務無法正常啟動
- Elastic Beanstalk 的 Health Check 判定失敗

解決方法

 使用 Elastic Beanstalk 的**版本回滾功能**：

- Console → Application → Versions
- 找到上一個成功部署的版本，按「Deploy」

 部署失敗時，通常 Console 會自動記錄上一次成功的狀態，可以快速 Rollback。

---

6. 頻繁超過資源限制（Instance CPU 爆表、Memory 爆滿）

問題現象

- Elastic Beanstalk 環境 Health 不穩
- 應用載入變慢或直接掛掉

常見原因

- Instance 選太小（例如 t2.micro 記憶體太小）
- 程式設計有記憶體洩漏（Memory Leak）
- 同一台機器上負載過高

解決方法

 調整 Instance Type：

- 將 EC2 Instance 換成 t3.medium、t3.large 或更高階型號
- Elastic Beanstalk 支援一鍵調整 Scaling 設定

 設定 Auto Scaling Policy：

- 當 CPU > 60% 自動擴展一台
- 當 CPU < 30% 自動縮減

 用 CloudWatch 監控 CPU、Memory 使用率，設通知（Alarm）

小結

Elastic Beanstalk 雖然大幅降低了管理負擔，但遇到問題時：

- **善用 Logs**
- **善用 Environment Health Status**
- **善用版本控制回滾機制**

這三個是最快速排除問題的關鍵！

下一篇，我們將進入 AWS 的 **Serverless 運算世界 —— Lambda！**
看看如果連 EC2、Load Balancer 都不想管，只想「寫 function」，AWS 能怎麼幫你做到！

Day 8 再見

修訂版本 #1
★由 treeman 建立於 25 ��|�� 2025 18:45:46
由 treeman 更新於 25 ��|�� 2025 19:00:04