

# 【AWS】【Elastic Beanstalk】常見部署結構推薦指南&費用最佳化建議

## Elastic Beanstalk 常見部署結構推薦指南

Elastic Beanstalk 很方便，但如果搭配得好，會讓你的應用又穩又省錢；搭配得不好，可能會出現：

- 資源不足導致當機
- 成本過高
- 擴展緩慢等問題

這篇針對不同情境，推薦最佳部署結構與選型建議。

### 1. 小型網站 / MVP 初版

#### 適用情境

- 一般公司網站
- 初版產品 Demo
- 內部管理系統
- 低流量 Web API

#### 推薦設定

項目	建議值
Instance Type	t3.micro 或 t3.small (免費額度或低價)
Instance 數量	最小 1 台，最大 1 台 (不開 Auto Scaling)
Database	內嵌式 RDS 或直接外接小型 DB (如 Lightsail)
Load Balancer	不使用 (單台直接對外即可)
部署方式	單一環境，使用 Rolling 更新

#### 原因分析

- 成本極低 (每月不到 10 美金)
- 流量低，單台就夠
- 部署簡單，維運容易
- 適合快速迭代、驗證市場

### 2. 中型後端服務 / API Server

#### 適用情境

- 中等流量的 Web API
- 手機 App 的後端
- 需要支援突發高峰流量

# 推薦設定

項目	建議值
Instance Type	t3.medium 或 t3.large
Instance 數量	最小 2 台，最大 4 台（開啟 Auto Scaling）
Database	外接獨立 RDS（例如 db.t3.medium）
Load Balancer	啟用 Application Load Balancer (ALB)
部署方式	Rolling with additional batch（保證不中斷）

# 原因分析

- 2 台機器可避免單點故障
- Auto Scaling 可以因應突發性流量（例如促銷活動）
- 獨立 RDS 提高資料庫效能與可管理性
- ALB 可支援 HTTPS 與多目標健康檢查

# 3. 高流量大型網站 / 高併發應用

# 適用情境

- 電商網站
- 多人即時服務（如聊天室、直播）
- 高 API 併發應用

# 推薦設定

項目	建議值
Instance Type	m6i.large 或 c6g.large（效能型）
Instance 數量	最小 4 台，最大 10 台（Auto Scaling）
Database	專屬大型 RDS（或 Aurora Serverless）
Load Balancer	Application Load Balancer（或加上 CloudFront）
部署方式	Immutable deployment（零中斷更新）

# 原因分析

- 使用效能型機器降低單位併發成本
- 多台擴展配合 Auto Scaling 可應付大幅波動
- Immutable 部署方式可避免更新失敗影響現有線上服務
- Load Balancer 可搭配 CDN 提升全球加速

# 4. Serverless + Beanstalk 混合模式 (Hybrid)

# 適用情境

- 需要兼顧複雜計算與 Serverless 架構
- 某些任務使用 Lambda 執行（如影像處理、寄信）
- 主系統仍以 Beanstalk EC2 部署

# 推薦設定

- 核心 API / Web server 放在 Elastic Beanstalk
- 背景工作 (cron job、queue consumer) 用 AWS Lambda + EventBridge
- File upload / Static Content 用 S3 + CloudFront
- Serverless 服務與 EC2 協作，共用 IAM role / VPC

## 原因分析

- 彈性最佳化：主服務 EC2 高效能、子任務 Lambda 靈活啟動
- 成本控制佳，避免小工作佔用大型主機資源
- 容錯能力提升，Lambda 失敗也不影響主要 API

## 小結

不同規模、不同需求下，Elastic Beanstalk 最佳部署建議：

規模	Instance建議	Auto Scaling	Load Balancer	Database建議
小型	1台小機器	☐	☐	內嵌 RDS 或外接小型 DB
中型	2-4台中型機器	☐	☐	獨立 RDS
大型	4-10台高效能機器	☐	☐ (加 CDN)	大型 RDS 或 Aurora Serverless
混合	Beanstalk + Lambda	☐	☐	跨服務組合

Elastic Beanstalk 的強大就在於  
從個人開發到企業級系統，都可以靈活搭配出適合自己的方案！

## Elastic Beanstalk 費用最佳化建議

(含 Auto Scaling 節省技巧)

Elastic Beanstalk 本身不收管理費，真正產生費用的是底層的 EC2、Load Balancer、RDS、S3、傳輸流量等資源。

所以如果想讓 Beanstalk 部署既穩定又省錢，關鍵在於：

- 控制 EC2 成本
- 適當調整擴展策略
- 避免無意義的浪費

這篇就帶你掌握 Elastic Beanstalk 最有效的省錢方法！

## 1. 使用 Reserved Instance (RI)

### ☐ 做法

如果你的應用是「穩定長期運行」（例如公司網站、後端 API），建議購買 **Reserved Instances** (RI) 來取代 On-Demand。

- 可省下 **最高 72%** 成本（依付款方案而定）
- 1 年期或 3 年期，分為 All Upfront / Partial Upfront / No Upfront 付款選項

“☐ 注意：RI 是鎖定在「Region + Instance Type family」（例如 t3 family）

### ☐ 適用時機

- 流量穩定的長期應用
- 預期至少持續使用 1 年以上

## 2. 配合 Auto Scaling 使用 Spot Instances

### □ 做法

Elastic Beanstalk 的 Auto Scaling Group 可以設定混合策略：

- 混合使用 On-Demand 與 Spot Instances
- 例如：
  - 30% On-Demand 保底
  - 70% Spot 便宜又彈性擴展

在環境設定裡選擇：

“Capacity → Auto Scaling → Instances → Purchase Options → "Combine purchase options and instance types"

這樣可以極大降低「額外擴展台數」的費用。

Spot 便宜很多（常常是 On-Demand 價格的 10%~30%），超適合做大量讀取、彈性擴展的情境！

### □ 適用時機

- 流量不穩定、會爆量但可以短期容忍失敗（如直播、促銷活動）
- 不要求 100% 可用性的非關鍵服務（如統計、備援服務）

## 3. 適當設定 Auto Scaling 門檻

Auto Scaling 是節省資源的重要機制，但如果設定不當，反而可能讓你爆量開機器！

### □ 推薦設定策略

- **Scaling Up (擴展)**：CPU 使用率 > 60% 才新增機器
- **Scaling Down (縮減)**：CPU 使用率 < 30% 才關閉機器
- **Cooldown Time (冷卻時間)**：
  - 擴展冷卻時間：180 秒
  - 縮減冷卻時間：300 秒

這樣可以避免瞬間小波動就導致無謂的開關機，減少 Auto Scaling 的震盪效應（flapping）。

## 4. 選擇適合的 Instance Type

不要一開始就開很大的機器，建議從小型機型開始測試：

- 小型 / 中型服務：選 `t3.medium` 或 `t4g.medium`（ARM 架構更便宜）
- 記憶體吃重服務：選 `r6g.large`
- 計算密集服務：選 `c6g.large`

**t 系列機型**特別便宜，還有 **CPU Credit** 機制，非常適合大部分輕中量應用。

“□ 如果有資格使用 AWS Savings Plan，搭配起來成本又能再下降！

# 5. 適時關閉閒置資源

Elastic Beanstalk 的環境如果沒有在使用，底層資源（EC2、EBS、ALB）還是會一直產生費用！

## 建議

- 測試環境（staging / dev）用完記得直接 **terminate**。
- 用 Lifecycle Policy 自動清除超過 30 天未使用的 Application Versions。
- RDS 如果是測試用，也記得關掉（RDS 比 EC2 還貴！）。

# 6. 用 CloudFront 做流量加速，減少 ALB 費用

如果你的 Elastic Beanstalk 是做靜態檔案 / API Server，流量高峰期可以前面加一層 **CloudFront CDN**，減少直接打到 Load Balancer 的次數。

- 減少 ALB Request 數量 → 降低 ALB 成本
- 加速全球存取速度
- 保護背後應用不被 DDoS 打爆（搭配 AWS Shield）

# 小結

Elastic Beanstalk 節省費用重點整理：

方法	核心效果
Reserved Instances	穩定運作環境省錢最大化 (>50%)
Spot Instances混搭	彈性負載環境大幅降低成本
正確設 Auto Scaling	避免不必要的機器啟動/關閉
選對 Instance Type	減少過度規格浪費
定期清理閒置資源	減少測試環境與多餘資源費用
加上 CloudFront	減少 ALB 成本與提高存取效率

Elastic Beanstalk 不只是簡單，只要稍微搭配得好，也能做到「低成本、高可靠、高效能」的部署！

🕒修訂版本 #2  
★由 treeman 建立於 25 🕒🕒🕒 2025 19:00:09  
✍由 treeman 更新於 25 🕒🕒🕒 2025 19:04:56