AWS 啟動彈性基準

AWS 方案指引

AWS 方案指引: AWS 啟動彈性基準

Table of Contents

簡介 1
指引範圍 1
共同責任模型 2
階段 1：設定目標 3
階段 2：設計和實作 4
階段 3：評估和測試 5
階段 4：操作 6
階段 5：回應和學習 7
後續步驟 8
Resources 9
AWS Well-Architected 架構 9
AWS 方案指引 9
其他 AWS 資源 9
貢獻者 10
編寫 10
檢閱 10
技術寫入 10
文件歷史紀錄 11
..... xii

AWS 啟動彈性基準

Amazon Web Services （貢獻者）

2026 年 3 月 (文件歷史記錄)

每個新創公司創始人都知道緊張局勢：團隊正在賽車運送新功能，而客戶期望每個版本都能正常運作。您一覺起，就會收到有關前晚中斷、客戶抱怨流入的通知，而且您的路線圖已包含承諾的功能。這是新創公司的每日現實，平衡創新壓力與維持穩定服務的需求。

排定新功能優先順序的本能很自然。投資者希望成長、客戶要求更多功能，而且競爭對手無法站立。然而，經驗顯示，即使是最創新的新創公司，復原能力問題也會脫軌。服務中斷不僅表示收入損失；還會侵略您努力與客戶一起建立的信任。事實上，堅若磐石的可靠性可以在擁擠的市場中成為您的靜音差異化因素。

在新創公司中建置彈性並不意味著拖慢或大量基礎設施投資。這是關於儘早做出明智的選擇，以防止日後發生問題。將它想成是建置房子。一開始就放入堅實的基礎比在建置完所有項目後修正結構問題要容易得多。透過將彈性實務融入您的初始架構和操作，您可以建立競爭優勢。您不僅提供產品，還提供建立客戶忠誠度的可靠體驗。

AWS 啟動彈性基準 (AWS SRB) 是專為面臨此挑戰的團隊所建立。它提供實用的路徑，可在系統中建置可靠性，同時維持讓啟動變得特別的速度。沒有繁重的程序或企業規模的複雜性，只是隨著業務成長的實際模式。

本指南說明如何在啟動環境中實作 AWS SRB。它可協助您識別恢復能力真正重要的事項、將有限資源集中於何處，以及如何建立隨著公司成長而擴展的實務。

指引範圍

探索建置啟動彈性的實際步驟時，請務必了解本指南在組織更廣泛的成長軌道中的位置。 AWS 啟動彈性基準符合彈性生命週期架構。它可做為初始路線圖，旨在協助您在部署第一個應用程式時，以最少的開發開銷實作基本彈性措施 AWS。將其視為您的入門套件，用於建置可有效從中斷中復原的可靠系統。

AWS Well-Architected Framework 和此彈性指引互相補充，以協助您建置安全可靠的基礎設施。本指南深入探討彈性實務，而 AWS Well-Architected Framework 提供更廣泛的架構最佳實務集。您可以使用 AWS Well-Architected Tool中的 ，針對這些實務評估您的架構 AWS 帳戶。

共同責任模型

釐清在雲端中建置彈性系統的重要層面至關重要： AWS 和 啟動之間的合作夥伴關係。雲端中的彈性在共同的責任模型上運作，與 AWS 和您的團隊各自在確保系統彈性方面扮演不同的角色。

AWS 負責支援其服務的基礎雲端基礎設施的彈性。將此視為您建置應用程式的基礎。這包括維護資料中心的彈性、聯網元件，以及 AWS 服務 您在架構中使用的核心。

您的新創公司的責任著重於您在此基礎上建置和部署之系統的彈性。本指南中的建議屬於您的責任範圍。透過深思熟慮地實作這些實務，您可以建立使用可靠 AWS 基礎設施並整合強大復原功能的應用程式。

此模型表示您永遠不會獨立建置彈性。您是以證實可靠的基礎為基礎，同時專注於直接影響客戶和業務營運的各個層面。本指南說明如何充分利用此共同責任模型，同時使用可靠的 AWS 基礎設施，同時實作實際措施，以確保您的應用程式從中斷中正常復原。

階段 1：設定目標

假設您的團隊在主要產品推出之前正處於最終衝刺中。新功能是突破性的，並且正在培養投資者的興奮。然後，在例行部署期間，您的核心服務會關閉。當客戶抱怨淹沒您的電子郵件時，兩個問題會變得很痛苦：您可以負擔離線多久？ 您可以承受遺失哪些資料？

希望一切都可以正常運作不是一個好的策略。您需要有系統的方式來決定彈性的最重要位置和不重要位置。這是業務影響分析 (BIA) 變得至關重要的地方。它可協助您對投資彈性的位置做出明智的決策。BIA 可協助您了解系統哪些部分真正需要堅如磐石的可靠性，以及哪些部分可以容忍一些靈活性。

首先映射您的核心使用者旅程。對於每個項目，請自問下列事項：

- 如果中斷，會有什麼影響？
- 我們必須多快還原服務？
- 哪些資料對保護至關重要？

這不只是技術練習；它可協助您了解可靠性問題對業務的影響。收入損失只是一個開始。考慮中斷可能會破壞客戶信任、違反法規要求，或為競爭對手提供邊緣。

在此分析中，您將為每個使用者旅程衍生兩個關鍵數字：復原時間目標 (RTO) 和復原點目標 (RPO)。RTO 定義您必須多快還原該旅程。RPO 會定義您的客戶可容忍的資料遺失量。這些業務驅動型目標接著會引導您選擇的元件，以及建構這些元件的方式，而不會過度設計系統的每個部分。

這種方法的優點是，它可協助您將有限的資源集中在最重要的位置。也許您的核心交易處理需要近乎即時的復原和零資料遺失，但您的建議引擎可以容忍更長的停機時間。透過設定明確目標，您可以建立架構，讓您繼續快速開發功能，同時以策略方式建置彈性。

明確記錄這些目標。它們不僅適用於您的工程團隊。當您向企業客戶推廣或向投資者進行技術盡職調查時，本文件會證明您已批判性地考慮業務持續性。

這些目標會隨著您的新創公司成長而演進。您前千名使用者的彈性需求與您第一個企業用戶端的需求不同。從您今天可實際達成的目標開始，但規劃它們在您擴展時如何收緊。

本指南說明如何實作符合這些目標的彈性措施。設定這些目標是您的關鍵第一步。它們是您駕馭創新和穩定性之間持續緊張關係的指南，可協助您建置可靠地為客戶提供價值的系統。

階段 2：設計和實作

本節討論如何實現您的彈性目標。您已映射出對業務最重要的內容，現在是建置它的時候了。如何在不拖慢創新的情況下建置彈性？

將 AWS 受管服務視為彈性捷徑。使用 服務為您處理備援，而不是消耗寶貴的工程時數來維護基礎設施。例如，考慮 Amazon Simple Storage Service (Amazon S3)。它會自動將資料的多個副本存放在中， AWS 區域 以提供耐久性。它不需要額外的程式碼或深夜分頁器職責。

您的核心應用程式元件呢？ 明智的選擇可能會使團隊的影響倍增。請考慮做為您服務骨幹的資料庫。請考慮使用自動處理容錯移轉的 Amazon Aurora，而不是建置您自己的複寫系統。這些功能可能成本更高，但它們會將團隊的焦點從維護基礎設施轉移到解決業務問題。此成本可以透過更快的功能交付來抵銷，並避免中斷時的收入損失。

有時，新創公司需要建置自訂解決方案。這就是創新新創公司的本質。當您這樣做時，請保持簡單但智慧。使用 Elastic Load Balancing 和 Amazon EC2 Auto Scaling 群組，將您的應用程式分散到多個可用區域。設定 Auto Scaling 群組最小容量，以處理基準流量，即使一個可用區域故障。這可為沒有複雜架構模式的局部故障提供彈性。隨著您的新創公司成長且客戶需要更高的彈性，您可以發展為更複雜的方法。

建議您將生產和開發環境保存在不同的 中 AWS 帳戶。當您快速移動時，混合它們很吸引人，但這個界限是您的安全網。它可防止意義良好的實驗縮減您的生產服務。將其視為「快速移動和破壞事物」開發文化的保險 - 破壞開發中的事物，保持生產穩定。

如果您的應用程式依賴第三方服務，請規劃其失敗。當您的付款處理器發生問題時，您的系統是否可以正常地處理它？ 建置簡單的斷路器和備用選項。可能將這些交易排入佇列，而不是顯示錯誤訊息。您的客戶會感謝您讓一切正常運作，即使 並不完美。

在建置時記錄 ，但請保持實用性。專注於記錄關鍵決策背後的原因，並建立簡單的復原手冊。當事件發生時，請務必備妥這些項目。

您並非為了完美恢復能力而建置；而是為了適當的恢復能力而建置。每個花費在過度工程彈性上的小時都是一個小時，不會花費在客戶要求的功能上。使用 AWS 受管服務作為您的基礎，在最重要的位置新增目標彈性，並建立明確的路徑以隨著業務成長擴展彈性。

下一章會討論如何驗證這些設計選擇，而不會消耗工程資源。對於新創公司，測試應該是合理提升和智慧型投資您應用程式的彈性。

階段 3：評估和測試

您已建置彈性基礎，但如何知道它實際運作？ 當您在競賽中證明產品市場適合度時，測試彈性可能聽起來很豪華。不過，有一種聰明的方法可以這樣做，而不會破壞您的功能開發。本章說明符合啟動速度的精簡實用測試。

從 開始AWS Resilience Hub，並將其視為初始架構評估工具。它提供架構彈性基礎的實用基準檢閱。它透過檢查常見的組態模式和潛在的單點故障，協助您評估基本基礎設施設定是否符合您的復原目標。它可以標記明顯的差距，例如缺少多個可用區域組態或不完整的備份政策。Resilience Hub 補充，但不會取代深思熟慮的架構檢閱和關鍵路徑的目標測試。

若要驗證您記錄的復原目標，請在開發環境中的 AWS Backup 中排程每月還原測試。即使需要工程時間，它可能比發現備份在實際事件期間無法運作更便宜。讓它成為您一般開發週期的一部分，例如執行單元測試或程式碼檢閱。目標並非完美；您可以在需要時復原。

隨著您的新創公司成長，客戶開始更加依賴您，逐漸升級您的測試遊戲。當您部署新功能時，請在管道中包含基本彈性檢查。使用 嘗試簡單的混沌實驗AWS Fault Injection Service。從生產前環境中開始，從小開始。在考慮任何生產實驗之前，測試您的應用程式如何處理開發中的延遲 API 回應。隨著您的可信度增加， 會逐漸擴展這些測試，但一律會先在生產前進行驗證。對於新創公司而言，破壞生產環境中的實物具有足夠的風險，而不刻意這樣做。

金鑰是平衡。用於測試的每小時都是一個小時，而不是用於建置新功能。但是，一些策略測試可以防止失去客戶信任的中斷類型。使用 提供的自動化工具 AWS 來執行繁重工作，並專注於對您的客戶最重要的測試。這可協助您建立對應用程式彈性的信心，而不會拖慢創新速度。

下一章將探討如何在啟動擴展時發展此基礎。

階段 4：操作

您已建置彈性應用程式並對其進行測試。現在，每日現實正在保持執行狀態。但是在啟動時，您無法監看所有操作，也不應該嘗試。關鍵是保持對重要事項的提醒，而不會提供太多指標或讓團隊負擔過重。

從客戶的觀點開始。Amazon CloudWatch Synthetics Canary 充當自動化客戶。他們會持續測試重要的使用者旅程。讓他們登入、使用測試帳戶模擬購買，或存取金鑰功能，尤其是在繁忙的時段。這可協助您了解客戶體驗，並協助您在實際使用者之前發現問題。當 Canary 失敗時，您立即知道從客戶的角度來看，發生錯誤。

在此基礎上建置，並專注於監控支援基礎設施。哪些訊號告訴您有問題？ Amazon CloudWatch 可協助您建置追蹤這些跡象的儀表板。不只是監控技術指標，還要將這些指標與業務影響相關聯。例如，高 CPU 用量很重要，但這可能會降低您使用 Canary 追蹤的客戶體驗。

做為實際的方法，請將您的監控映射至您的客戶旅程。如果您執行軟體即服務 (SaaS) 平台，您可能會關心 API 回應時間、身分驗證成功率和核心功能可用性。設定提醒，告知您這些指標何時偏離。不過，請選擇性。每個提醒都應要求採取動作。如果您的團隊因為「可能什麼都不做」而開始忽略提醒，表示您已設定太多或正在追蹤錯誤的指標。

透過您的團隊已使用的工具路由這些提醒。如果您的工程師住在特定的簡訊應用程式中，請在該處傳送提醒。目標是在不建立新程序的情況下快速感知。當警示觸發時，您的團隊應該確切知道它意味著什麼，以及如何處理它。

保持營運文件精簡且實用。在版本控制中存放具有程式碼的 Runbook，但請記住它們不是小說。當某件事休息時，您的團隊需要明確、可行的步驟。每個提醒都應連結到對應的 Runbook，而且每個 Runbook 應回答三個問題：

- 什麼會中斷？
- 它為什麼重要？
- 我該如何修正這個問題？

實作簡單的事件管理程序。您不需要複雜的架構，只需明確定義什麼構成事件，以及在情況升級時呼叫誰即可。保留事件日誌，因為它們可協助您改善應用程式的彈性。

關鍵在於尋找警示和額外負荷之間的甜甜點。使用 AWS 工具來自動化您可以執行的作業、專注於監控影響客戶的指標，並讓您的程序保持足夠光線，以隨著您的成長而演進。

下一章探討如何培養彈性思維，而不犧牲讓新創公司變得特別的速度和創新。在一天結束時，彈性與人們的彈性一樣多。

階段 5：回應和學習

當您執行啟動時，複雜的事後程序可能會拖慢您的團隊。本章說明如何從事件中學習，而不將其轉換為官僚練習。

將事件學習整合至您現有的節奏。如果您的團隊已有定期會議，請使用十分鐘來討論最近的事件。專注於實際問題，例如：

- Runbook 是否有幫助？
- 提醒是否在正確的時間發生？
- AWS 受管服務可以防止這種情況嗎？

專注於動作，而不是指責。在新創公司中，您並未建置完美的系統；您正在建置一個系統，每當發生問題時就會變得更好。

您可以使用您的票證系統來追蹤事件；不需要專用工具。建立簡單的範本，其中包含事件時間表、客戶影響、採取的復原步驟，以及經驗教訓。如果您主動使用，此攝影機會成為機構記憶體。在加入期間檢閱過去的事件，讓新工程師加快速度。在設計類似的系統時，在架構檢閱中參考它們。將它們拉進遊戲日，根據實際事件建立逼真的失敗案例。範本會擷取發生的情況，並定期使用 將其轉換為組織學習。

隨著新創公司成長，模式就會出現。某些元件可能會更頻繁地失敗，或者特定類型的變更可能會導致問題。使用這些模式來引導彈性投資。如果資料庫容錯移轉造成問題，請考慮改善您的多個可用區域設定。如果第三方服務中斷是常見的主題，請考慮改善斷路器。

目標不是要防止所有可能的失敗。這是不可能的，而且會拖慢您的速度。目標是快速學習、快速調整，並在快速成長的同時保持應用程式的可靠性。使用每個事件作為機會，讓您的系統更具彈性、您的團隊更有知識，以及您的客戶對您的服務更有信心。對於新創公司，速度和學習會打敗完美。建立輕量型程序，協助您從事件中學習，而不會拖慢創新速度。最佳彈性實務是您團隊實際使用的實務。

啟動的後續步驟

還記得讓您的心競賽的深夜部署嗎？ 或者，當主要客戶詢問您的彈性措施時？ 新創公司的彈性旅程不是要消除這些時刻，而是要以自信面對。

本指南說明復原能力的實際路徑，包括：設定逼真的復原目標、使用 AWS 受管服務、實作精簡測試、監控重要事項，以及從事件中學習。這種方法對於新創公司來說非常強大，因為它會隨著您的成長而成長。協助您的新創公司從現在的問題中復原的相同架構，為企業客戶明日提供服務奠定了基礎。

將彈性視為產品藍圖；您不會一次建置一切。從保護核心服務開始。新增關鍵客戶旅程的監控。實作可捕捉明顯問題的基本測試。記錄您學到的內容。每個步驟都是可管理且實用的，最重要的是，它們不會阻止您運送功能。

隨著新創公司的成長，彈性需求也隨之演進。企業客戶可能會提出更嚴格的問題。尖峰流量可能會達到新的高點。國際擴展帶來了新的挑戰。由於您已在基礎中建置彈性，因此已準備好進行調整。

啟動的目標不是完美的運作時間，也不是零事件。目標是建置足夠可靠的系統來獲得客戶的信任，同時維持讓新創公司變得特別的速度。透過遵循此方法，您可以建立比建立完美系統更有價值的內容；您可以建立每天更好的方法，方法是從每個挑戰中學習，同時跟上新創公司的野心。

恢復能力旅程並未結束。它會隨著您運送的每個功能、您獲勝的每個客戶，以及您處理的每個事件而演進。本指南可協助您建置架構，協助您向前邁出自信且實用的步驟。最後，啟動成功與選擇彈性和速度無關。這是關於尋找同時交付兩者的智慧方法。

Resources

AWS Well-Architected 架構

- 可靠性支柱
  - REL05-BP01 實作適度降級，以將適用的硬相依性轉換為軟相依性
  - REL06-BP01 監控工作負載的所有元件
  - REL06-BP03 傳送通知 (即時處理和警示)
  - REL08-BP03 將彈性測試整合為部署的一部分
- 彈性的共同責任模型

AWS 方案指引

- 上的備份和復原方法 AWS
- 彈性生命週期架構：持續改善彈性的持續方法

其他 AWS 資源

- AWS 故障隔離界限 (AWS 白皮書)
- 為雲端應用程式建立 RPO 和 RTO 目標 (AWS 部落格文章)
- 上的部署選項概觀 AWS (AWS 白皮書)

貢獻者

下列個人對本指南有所貢獻。

編寫

- Dylan McCarroll，副解決方案架構師， AWS
- Igor Fil，啟動解決方案架構師， AWS
- Parth Shah，資深解決方案架構師 AWS
- Vrajesh Prajapati，解決方案架構師， AWS

檢閱

- Alice Richey，首席技術人員， AWS
- Bruno Emer，首席技術人員， AWS

技術寫入

- 資深技術作者，Lilly AbouHarb AWS

文件歷史記錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 RSS 摘要。

變更	描述	日期
重大更新	更新了 AWS 服務 整個 的建議和使用。	2026 年 3 月 6 日
初次出版	—	2024 年 1 月 24 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。