對本書的讚譽

對於任何想要在軟體即服務（SaaS）業務中建構、維持並蓬勃發展的人來說，本書提供物超所值的指導。不僅有現實世界中的常見挑戰解決方案，書中的模式與實踐也都能禁得起時間的考驗。

— *Adrian De Luca*，*AWS* 雲端加速部門總監

這是一本全面的 SaaS 概念參考書，深入探討真實世界中的架構模式，涵蓋安全性、租戶隔離、可擴展性等各個方面。對於任何想要建構多租戶 SaaS 解決方案的人來說，都是一本不可或缺的指南。

— *Tony Pallas*，*ShyTouch Technology* 首席商務暨技術長

本書精采地聚焦於關鍵的領域概念和槓桿原理，這些都是在打造成功的 SaaS 或 PaaS 產品時，必須掌握的核心要素。

— *Russ Miles*，*Clear.Bank* 平台工程師

Tod 多年來與各類客戶合作的豐富實戰經驗，在這本書中發揮得淋漓盡致。對於想要在 AWS 平台建構可擴展且安全的 SaaS 解決方案開發者來說，這本書將是寶貴資源。

— *Anubhav Sharma*，*AWS* 首席解決方案架構師

第三章

多租戶部署模型

作為 SaaS 架構師，選擇多租戶部署模型是需要優先考慮的關鍵決策之一。在這個階段，應該暫時擱置多租戶實現的技術細節，轉而從更宏觀的角度思考 SaaS 環境的基礎架構。您對應用程式部署模型的選擇，將對成本結構、營運效率、服務分級策略以及諸多其他關鍵因素產生深遠影響，而這些因素都將直接決定 SaaS 業務的成功與否。

本章將深入探討一系列多租戶部署模型，分析運用這些模型來應對各種技術和業務挑戰的方法。在討論過程中，我將詳細闡述每種模型的優劣，幫助您深入理解所選模型對 SaaS 產品複雜度、擴展性、性能和靈活性的影響，透徹理解這些模型的核心價值和取捨，將是制定有效架構策略的關鍵，這種策略需要在業務現實、客戶需求、時間壓力和長期 SaaS 目標之間取得平衡。儘管這些模型中存在許多 SaaS 團隊共同關注的主題，但沒有什麼解決方案放之四海而皆準，您的任務是權衡各種部署模型，選擇一個或多個模型的組合，以滿足當前和未來的需求。

本章還將持續擴充 SaaS 專業詞彙，為這些模型及其支援結構引入新的術語，它們將貫穿本書後續章節，能幫助您更精確地描述 SaaS 環境的特性，從而更清晰、細緻地闡述多租戶架構的各個組成部分。這些術語和概念將使您能夠進一步描述和分類 SaaS 架構，以適應實際應用中各式各樣的多租戶布建。

開始部署模型後，就會看到特定技術在探索這些模式中逐漸顯現。雖然部署模型模式與特定技術並沒有直接對應關係，但開始以更具體的技術實現時，也將看出成形的過程，在這個階段，會開始看到更多 Amazon Web Services（AWS）的服務和機制浮現。不過一般來說，無論使用什麼工具和技術，都很可能找到與這些 AWS 組件功能相對應的替代方案。

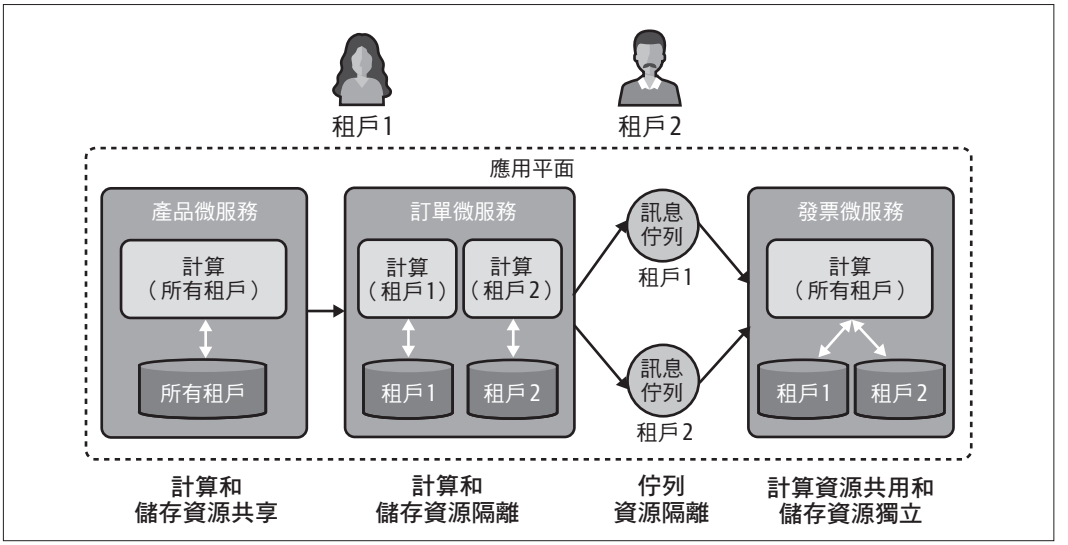

圖 3-2　獨立和共享資源模型

從產品微服務開始，此服務採用將所有租戶的運算和儲存資源以共享模式部署的策略，在這個情境中，我認為共享方法最能滿足該服務在隔離性和效能方面的需求。接著看訂單微服務，您會發現我選擇一個截然不同的模型，這裡的每個租戶都擁有獨立運算和儲存資源，這個決策同樣基於環境的特定需求，可能是受某些 SLA 要求或合規需求所驅動。

從訂單服務開始，您會發現系統將訊息傳送至一個負責準備訂單計費的佇列，之所以特別設計這個情境，是為了強調獨立式和共享式的概念不僅適用於微服務，還能擴展到環境中的任何資源；如這個解決方案中，我選擇為每個租戶設定專屬佇列。最後，右側的發票服務會從這些佇列中提取訊息並產生發票。為了滿足解決方案的需求，我在這個微服務中採用獨立式和共享式模型的混合策略，具體而言，運算資源採用共享模式，而儲存資源則是獨立布建。

「獨立」和「共享」這兩個核心術語主要用於描述資源的架構布建，不僅能精確描述租戶與架構中特定元素的對應關係，還可以更廣泛地用來說明租戶資源集的部署策略。因此，不應將獨立和共享僅僅限制在特定結構上，而應該將它們視為描述單一或多個資源租戶模式的通用辦法。

這個概念對於理解本章的部署模型至關重要，讓我們能在多租戶架構的各個層面靈活運用獨立和共享的概念。

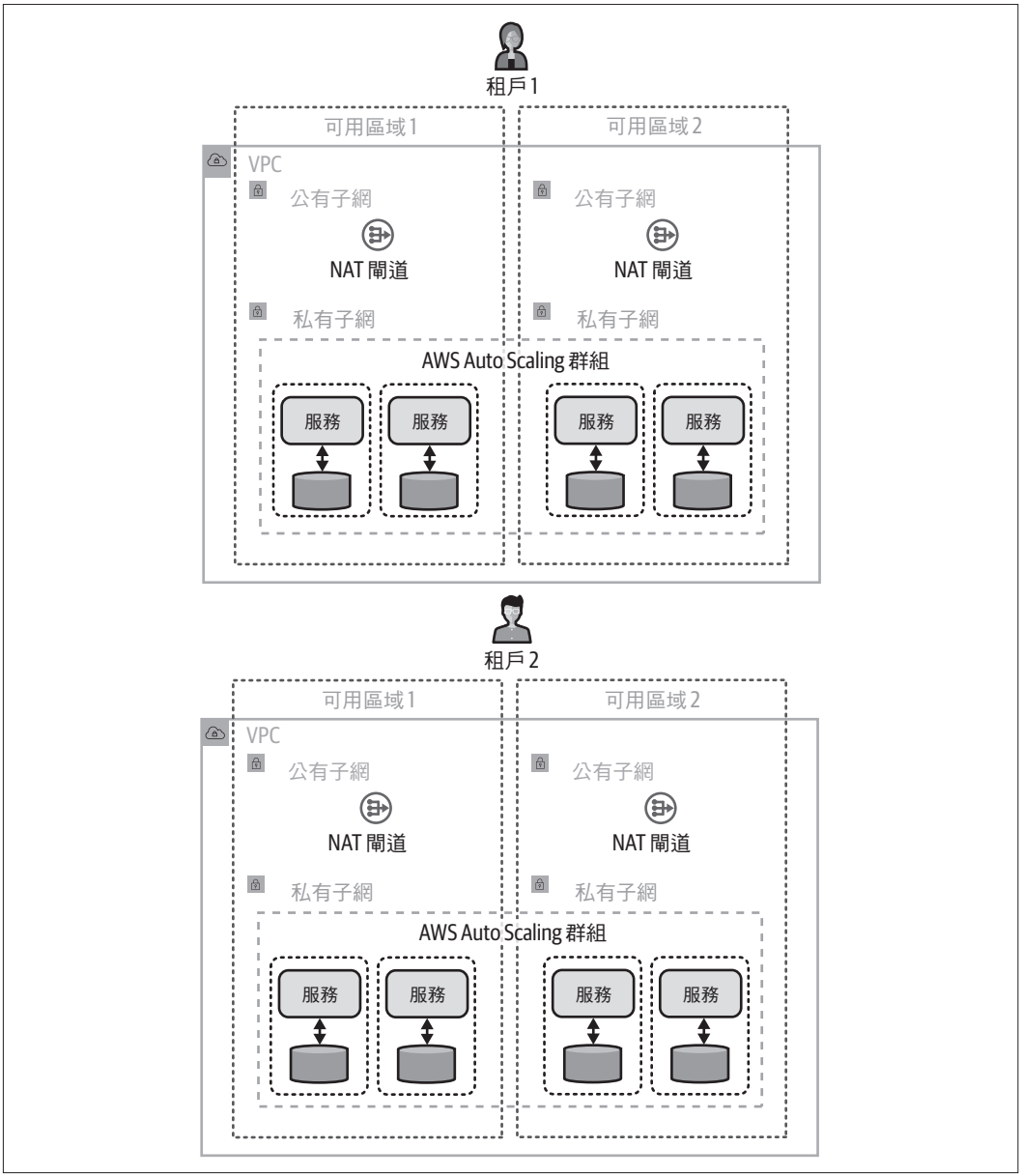

圖 3-6　VPC 租戶獨立的全堆疊式架構模型

混合全堆疊部署

到目前為止，全堆疊式獨立和全堆疊式共享部署架構對我來說，就是解決全堆疊問題的兩種獨立方案，將這兩種模型視為應對一組需求對立的方法，並認為它們互不相容是可以理解的觀點。然而，如果退一步將市場和業務現實納入考量，就會發現某些組織可能認為同時支援這兩種模型有其價值。

圖 3-10 是一個混合全堆疊部署架構的範例。這裡可以看到並列呈現先前討論過的全堆疊式獨立和共享部署模型核心概念。

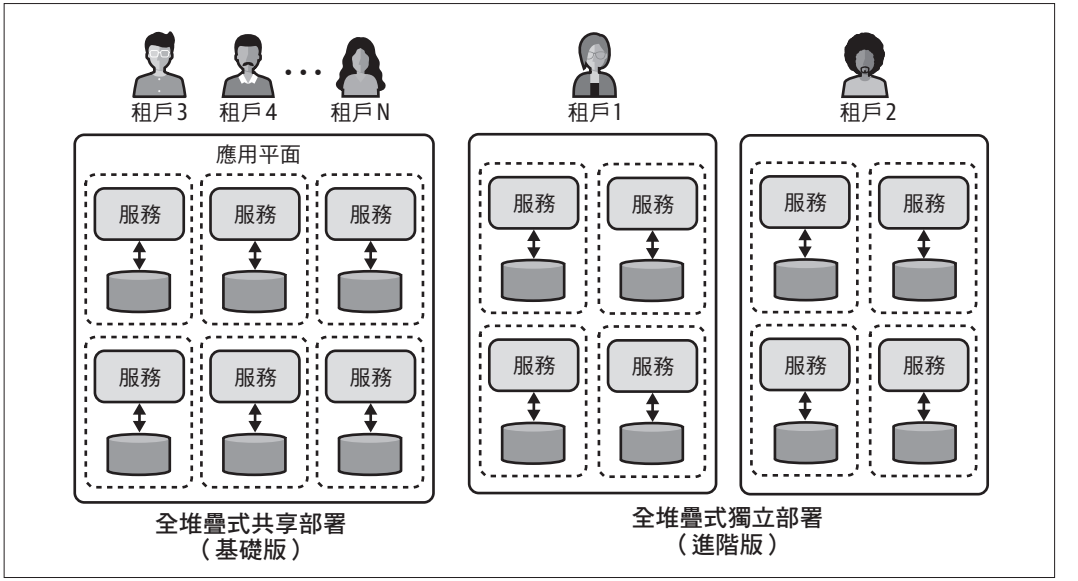

圖 3-10　混合全堆疊部署架構

所以，為什麼要同時採用兩種模型？什麼原因之下要採取這種混合方法？試想一下，您已經建立 SaaS 業務，並在一開始為所有客戶提供全堆疊式共享架構，即圖左側。然而，在業務發展過程中，您遇到不信任共享模型的客戶，他們可能擔心吵雜鄰居問題或特定的合規要求，這時候，您不應該對所有持有異議的客戶妥協，因為這可能會損害 SaaS 業務的核心價值主張；相反的，您應該致力於幫助客戶理解您為滿足其需求而採取的安全措施、隔離策略和其他技術方案，這是銷售 SaaS 解決方案過程中不可或缺的一部分。然而，在某些特殊情況下，您可能會考慮為特定客戶提供專屬全堆疊式獨立環境，可能是因為重要的戰略機遇，或者是某些客戶願意支付溢價，從而使提供全堆疊式獨立選項在經濟上變得可行。

第七章

建構多租戶服務

目前為止的重點，主要還是集中在建立多租戶架構的所有基礎元素上，諸如深入研究控制平面，並找出建立一組核心服務的方法，使我們能夠將租戶概念引入 SaaS 環境。我們探討了租戶導入、建立其身分、進行身分驗證的方法，以及最重要的，這一切最後將租戶上下文注入到應用程式服務中的方式。這應該讓您充分認識到控制平面在 SaaS 環境中扮演的角色，突顯投資於開發無縫整合策略的重要性，以便將基礎租戶架構有效地融入多租戶環境中。

現在，可以開始將注意力轉移到應用程式平面，在這裡思考將多租戶應用於設計和實現，使應用程式更為生動的服務。本章會開始探討多租戶工作負載的細微差別，對設計和分解服務方式的影響。隔離、吵雜鄰居及資料分區等，都代表需要納入服務設計考量的新參數，您會發現多租戶為經典服務設計討論增添新的複雜性，迫使您對服務的規模、部署和資源占用採取新方法。

租戶導入也會直接影響實現服務的方式。我們將仔細研究多租戶融入服務程式碼之處和方法，強調可用於防止租戶增加服務整體複雜性，和 / 或資源占用的不同策略。我將探討幾個服務實現的範例，並概述可用於將多租戶結構推入輔助程式和程式庫的工具和策略，以簡化整體開發人員體驗。

更廣泛的目標是讓您全面性了解開始建構多租戶服務時應考慮的各種因素。從一開始就將此作為優先事項，會對 SaaS 解決方案的效能、複雜度和可維護性產生重大影響。

在本章中，您會發現我在描述 SaaS 應用程式的組件時，使用更為通用的術語「服務」。我刻意避免將這些例子對應到任何特定的服務實現策略，您可能認為這個概念與微服務相關，然而，我不想預設您的解決方案已採用微服務架構。

設計多租戶服務

在探討建構多租戶服務的方式之前，我們需要提升視角，審視您在識別系統中不同服務時需要考慮的規模、形態和一般分解策略。在多租戶模型中，服務的邊界以及分配負載和責任到這些服務的方法，增加複雜性和前瞻性思考的維度。

傳統軟體環境中的服務

為了進一步理解這種動態，先來看看一個傳統應用程式，它的資源占用是單獨安裝、部署和管理的，圖 7-1 是簡化的例子，說明服務在這些傳統、安裝式軟體環境中部署的方式。

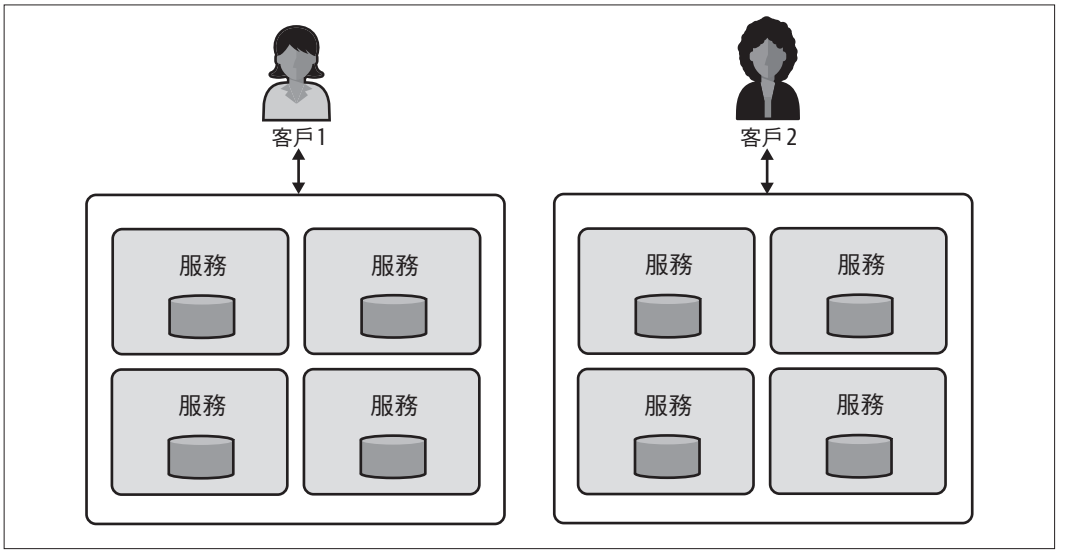

圖 7-1　傳統軟體環境中的服務

您會看到服務完全專屬於個別客戶。在為這些環境設計服務時，重點主要是找到一個優質的服務組合，以滿足單一客戶的擴展性、效能和容錯需求。誠然，客戶使用您系統的方式可能有些差異，但一般重點是創建一個僅限於單一客戶行為和布建的體驗。

這種較為聚焦的方法使得確定服務邊界相對簡單。大部分重點會放在單一責任設計原則上，試圖確保以一種方式分解服務，使每個服務都有明確、定義良好的範圍和功能角色。這個理念是讓每個服務都有一個明確定義的職責。

共享多租戶環境中的服務

現在，來看看全堆疊共享多租戶環境。圖 7-2 就是 SaaS 架構的例子，該架構支援多個租戶在共享模型中共用其基礎設施資源的需求。

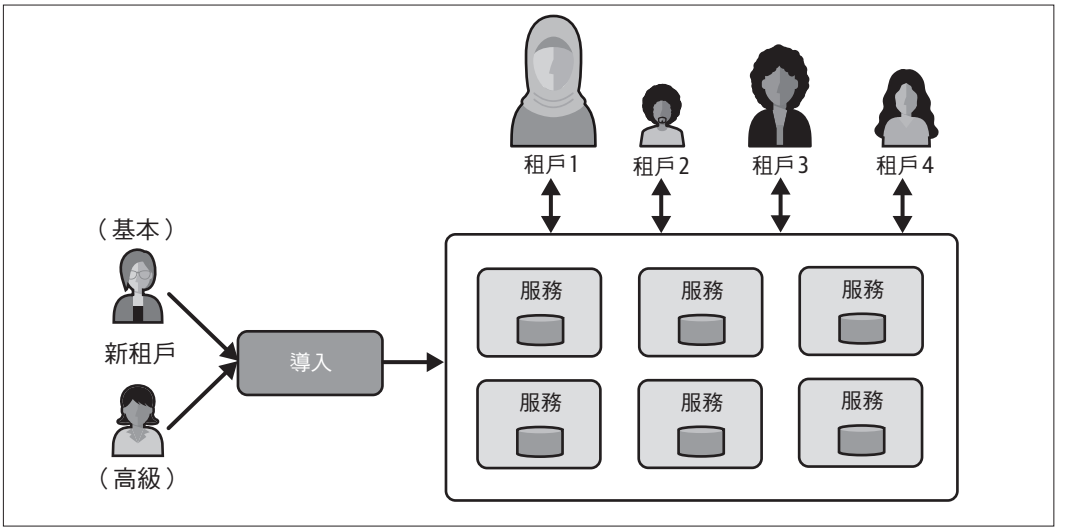

圖 7-2　共享多租戶環境中的服務

表面上看，這裡似乎只是使用這些服務的租戶數量發生了變化。然而，這些租戶同時作為共享資源使用這些服務的事實，對於處理每個服務的規模、分解和資源占用方式有重大影響。

您會首先注意到，我在圖 7-2 頂部特意引入不同規模的租戶。這樣做是為了說明租戶對系統施加負載的方式可能存在巨大差異。一個租戶可能會使系統的某部分達到飽和，另一個租戶可能會使用您解決方案的全部功能，但對環境施加的負載卻很小。這些組合可能變化多端。

左側展示新租戶的導入過程，這是為了傳達可能隨時有新租戶引入您的環境概念，這些新租戶的工作負載和特徵幾乎無法預測。我還強調這些新加入的租戶可能屬於不同層級，有著不同體驗和效能期望。

現在退一步思考我們手上握有的牌。這個環境中的服務由所有租戶共享，必須以某種方式預料到每個租戶角色的所有擴展、效能和資源消耗需求，需要高度關注，確保這些租戶不會造成吵雜鄰居的現象，即一個租戶影響另一個租戶的體驗。這些服務還需要根據

可能相當難以捉摸的一系列參數動態擴展，今天使用的擴展策略，可能無法與您的服務明天或下一個小時需要的擴展保持一致。SLA、分層布建、合規性和其他考慮因素也可能在這個等式中。

這本質上是共享基礎設施的優勢，與支援不斷變化的客戶使用模式現實產生衝突之處。對某些人來說，這導致資源的過度布建，以應對不斷變化的需求和負載特徵，恰好與 SaaS 商業模式相關的效率和規模經濟目標相悖。

在某些方面，這都是擁有多租戶共享架構的一部分，即使服務布建過度，共享這些資源的整體價值可能仍遠高於擁有專用的每租戶基礎設施。然而，在提供更多工具和策略，以應對租戶工作負載和特徵的不斷變化需求方面，您的服務設計可以發揮重要作用。一般來說，設計服務的方法主要聚焦在為自己提供更多調節機制，以應對租戶對環境施加的各種動態變化。

擴展既有最佳實踐

識別多租戶服務的過程，仍將遵循許多用於確定候選服務的公認方法和策略。關鍵在於，應用這些概念時，還需要將額外的多租戶設計考量因素，納入塑造服務設計的要素清單中，將基礎最佳實踐與一系列多租戶考量相融合，如圖 7-3 所示。

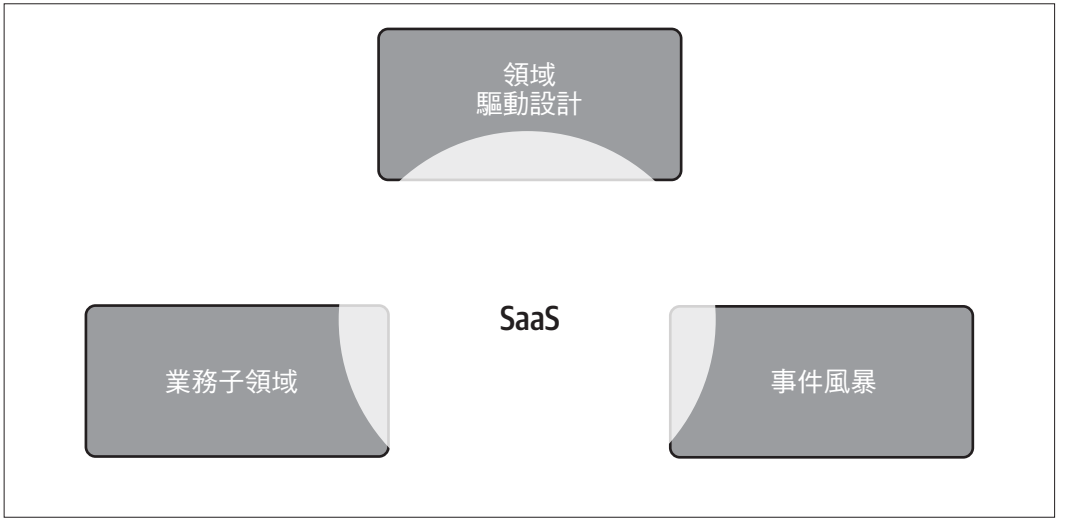

圖 7-3　整合服務設計方法論

第十三章

SaaS 遷移策略

轉型至 SaaS 的旅程並非總是從零起步。事實上，許多組織已擁有成熟的解決方案，期望將其升級為 SaaS 服務模式。這種轉型可能來自多重驅動因素，對某些組織而言，採用 SaaS 可能純粹為了解決成本、營運和規模擴充的效率挑戰。其他組織則可能面臨新興 SaaS 競爭者帶來的市場壓力。在某些情況下，這可能著眼於利用 SaaS 的規模經濟優勢來擴展業務，並開發新市場區塊的策略布局。客戶需求同樣可能是關鍵推手，促使企業將其解決方案轉型為 SaaS 模式。

雖然轉型至 SaaS 的效益顯而易見，但制定最佳轉型策略卻充滿挑戰。擁有穩定的產品、客戶基礎和營收時，這種轉型必然會引發諸多思考，勢必得在延續與創新之間取得平衡，找到一條能夠推動變革，又不會干擾現有業務的發展軌跡。對於必須回應季度財報和營收預期的上市公司而言，這種轉型更顯艱鉅，產業特性、客戶屬性以及法規遵循要求，都可能構成約束，使 SaaS 轉型變得更為複雜。這正是為何沒有所謂的一體適用 SaaS 轉型方案，SaaS 遷移的精髓在於找到最能符合您業務現況的成長路徑。

本章將審視您在選擇遷移路徑時需要評估的關鍵要素、模式和策略。我們將從探討遷移過程中的整體動態開始，分析團隊在權衡即期壓力、市場因素、營運挑戰、成本效益、團隊實力以及其他構成遷移難題的眾多變數時，所面臨的各種拉扯。

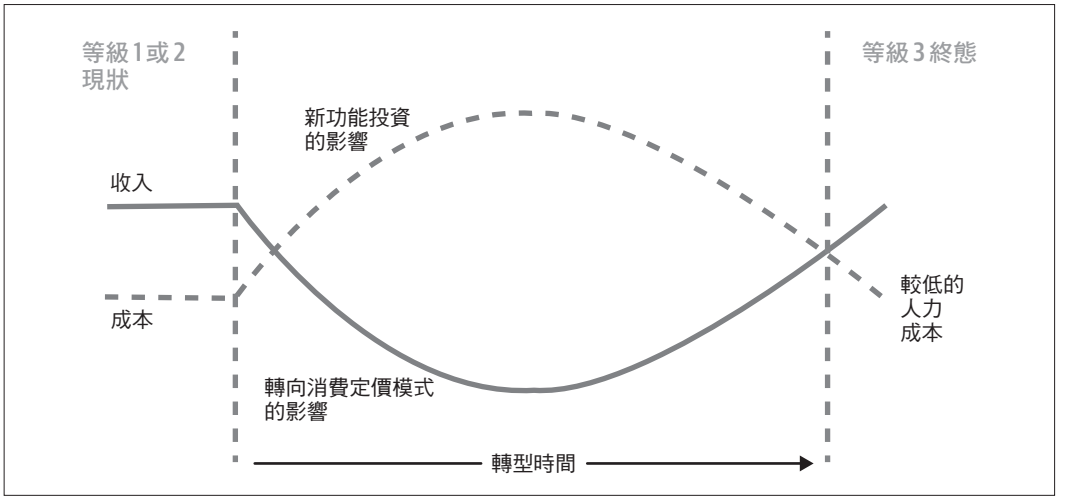

圖 13-3　遷移魚骨模型

這張圖表因其獨特形態而有魚骨圖之稱，同時呈現營收和成本走勢，展現企業在轉型至 SaaS 服務模式過程中的財務變化。圖中上方的虛線描繪成本變動軌跡，在遷移的過程中，團隊需要建構全新的系統框架，以實現營運效能、業務靈活性和成本最佳化等策略目標。對某些組織而言，這項轉型投資可能產生額外的階段性支出，也許源自擴充人力資源、引進新型工具、採用創新技術等多個面向。

魚形圖底部的實線反映營收走勢。在轉換至新的定價和收入模式過程中，業務營收可能會經歷下滑期，這可能源於轉向使用量計費模式或其他定價策略的過渡效應，導致短期內營收減少。這裡的核心概念是，建構支援 SaaS 服務所需的完整基礎架構，必然會產生額外的前期投資。

這種情況自然會讓許多正在轉型的企業感到憂慮，他們通常希望將這條「魚」的體型控制得越小越好。但好消息出現在圖表右側，建立起能支撐高度靈活多租戶環境的各項效率機制後，您終將看到成本曲線下降。同時，隨著客戶群逐步轉移到新的定價模式，您將能夠善用 SaaS 的效率和靈活性來實現規模經濟效益；理想情況下，您還能拓展新客戶群和市場領域，加速業務成長。

在這個關鍵轉折點上，您會看到營收開始上揚，而成本持續下降，彰顯這次轉型的實質價值。這正是 SaaS 模式發揮最大效益的階段，使組織能夠充分運用 SaaS 優勢，來最大化利潤並實現永續成長。

儘管這些新服務在建置時即具備多租戶功能，但在遷移過程中它們僅涵蓋部分系統功能，這表示您的控制平面只能取得系統部分的資料和營運資訊。既有應用平面對控制平面完全無感知，且原始設計時也未考慮租戶概念，這導致遷移期間，團隊需要從架構的兩個不同層面建構營運視圖，不僅增加營運團隊的負擔，也使系統狀態監控變得更為複雜。

這可能促使團隊投資在既有應用平面中建立新的控制平面整合機制，即使明知最終將汰換掉這些程式碼。您需要在投資成本和短期效益間找到平衡點，確實需要為既有應用平面增加額外的監控機制，理想情況是能向控制平面提供充分的營運資訊，同時避免造成重大的架構偏離。這很大程度取決於在不造成明顯影響的情況下，您的技術堆疊要在哪些環節導入這些監控機制。

模式比較

我已詳細探討每種遷移策略的獨特優勢與挑戰，也試圖闡述每種策略所涉及的業務和技術層面的考量。為了更清晰地呈現這些方法的取捨權衡，圖 13-11 提供這些策略優缺點的高階綜覽。

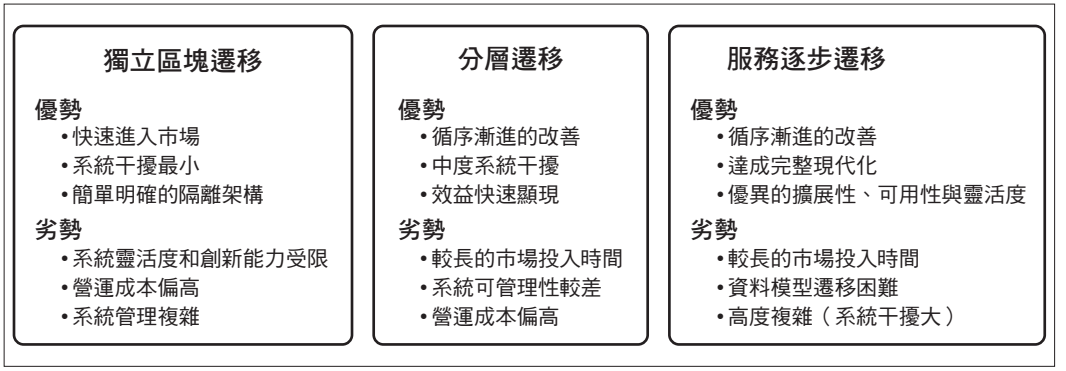

圖 13-11　遷移策略優缺點分析

獨立區塊遷移策略的核心優勢在於其執行速度和操作便利性，這種低干擾性的特點使系統能夠在毋需深入修改應用程式程式碼的情況下，快速上線運行；然而，這種便利性是以較高的營運複雜度、較低的靈活性、效率和較高的營運成本作為代價。它會限制您在 SaaS 環境中透過資源共享實現規模經濟效益的可能性，不過，若您的最終目標本就是完全隔離的架構模式，這些限制可能就不是關鍵考量。

分層遷移策略則提供一個平衡方案，讓您能在避免大規模重構或重寫的前提下，達成特定領域的效率優化，優點在於適度的干擾性，並能在初期階段就展現實質效益；但當然，任何重構工作都會對產品上市時程產生一定影響。此外，雖然能獲得效率提升，但這些改善仍有其限制，在這個策略中，仍需處理成本管理和系統維運的挑戰。而這些共享層的設計也可能在您的環境中增加單點故障的風險。

最後，服務逐步遷移（漸進替換）模式更著重於加速實現系統現代化，儘管這仍是一個漸進式的策略，但轉換至新微服務所需的工作和投資會相對現有系統造成一定程度的影響，不僅會延長產品上市時程，也會增加整體遷移過程的複雜度。值得慶幸的是，這仍然是一個循序漸進的模式，因此一旦基礎架構就位，就能夠按部就班地將您的環境演進為全面現代化的系統。

分階段實施策略

前面章節所描述的這些模式並非互斥關係，也不代表涵蓋所有可能的遷移路徑。特別要提醒的是，這些模式可以作為分階段策略的組成部分，讓您能夠從一種模式，平穩地過渡到另一種模式。圖 13-12 說明在分階段策略中整合運用這些模式的方法。

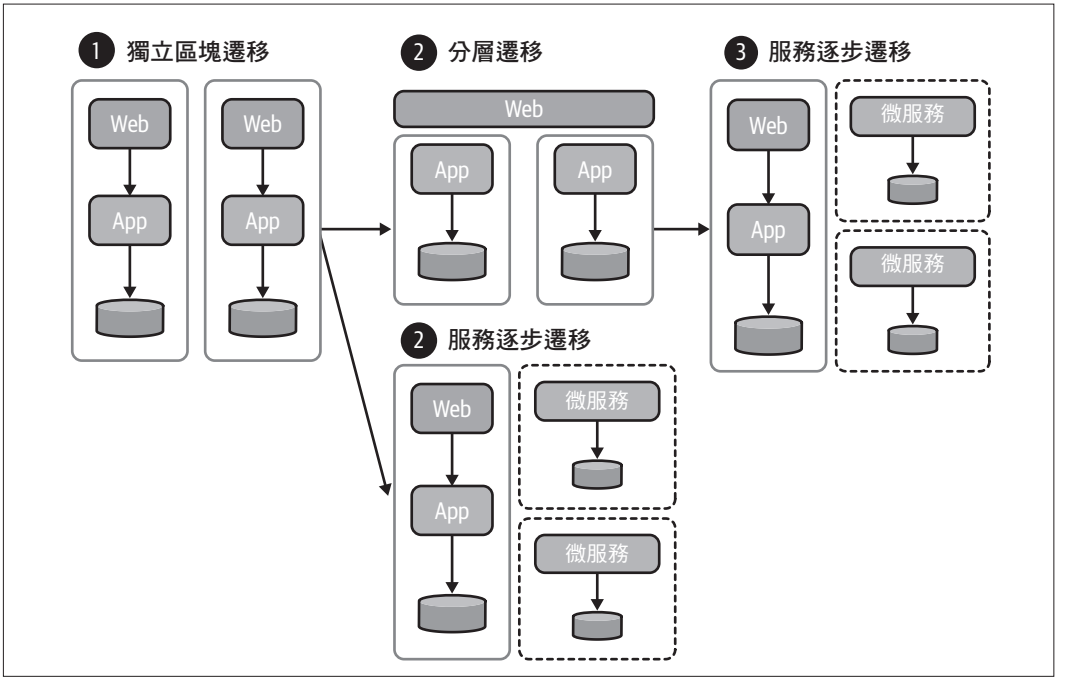

圖 13-12　遷移策略的整合應用

第十六章

生成式 AI 與多租戶架構

整個軟體開發領域，都在探索將生成式人工智慧（GenAI）融入他們產品中的方法和時機；GenAI 開創全新的機會領域，促使團隊評估將 GenAI 架構導入他們系統中的辦法與位置。當我們審視 GenAI 的潛力時，可以預見它將對來自各種領域和應用場景的系統產生深遠影響。就我們的目標而言，我希望找出 SaaS 供應商可以將 GenAI 架構與多租戶特性結合的領域，以打造具有差異化的創新體驗。基於這個考量，本章聚焦於描述特定的 GenAI 架構策略，使 SaaS 供應商能夠在其 GenAI 模型中導入租戶情境能力。這個額外的情境將使 SaaS 供應商能夠建立單一共享的多租戶 GenAI 架構，在推理時運用租戶情境，為個別租戶需求產生客製化的回應。這也開啟全新的多租戶考量面向，隔離機制、資源競用、成本結構和定價策略，都需要用創新方法，在多租戶情境中實現這些原則。

然而，在開始之前，需要先建立基礎，檢視建構具備 GenAI 能力的多租戶解決方案所涉及的核心概念。這裡的目標是在考慮將多租戶實務整合到您的整體 GenAI 策略之前，透過闡述基本架構元素，為整體 GenAI 和 SaaS 生態系統提供更清晰的藍圖。

在奠定基礎後，本章開始探討您可以布建，以支援各租戶 GenAI 功能的特定 GenAI 架構和機制。這個理念是要思考擁有一個所有租戶共享的大型語言模型（LLM）的意涵，同時保持支援圍繞該 LLM 進行租戶專屬布建的能力，以實現獨特的租戶或領域體驗。我們將探討的兩個核心架構是檢索增強生成（RAG）和微調技術，著重闡述將這些機制應用到多租戶架構時的關鍵差異。

在整合這些新的 GenAI 功能時，還必須思考將核心多租戶原則應用於這些架構的方法，本章的這個部分將探討將租戶隔離、資源競用和導入等傳統多租戶概念，應用到這些新機制中的辦法。如何隔離 GenAI 請求？如何防止租戶造成系統過載？這些都是在環境中導入 GenAI 時需要思考的新挑戰。最後會探討這些 GenAI 策略對您的定價模式、成本分攤、分層架構和流量控制策略框架的影響，以結束本章。

這裡更廣泛的目標是啟動多租戶 GenAI 的討論，識別 GenAI 和多租戶架構交會的潛在領域。重點在於，隨著 GenAI 的興起並逐步融入更多 SaaS 解決方案中，需要持續評估它可能對 SaaS 架構的整體藍圖影響方式和位置。同時值得注意的是，這是一個快速發展的領域，會一直調整近期指導方針。

核心概念

在深入探討多租戶 GenAI 的具體細節之前，建立基礎架構以進一步理解技術全貌相當重要，可以探討在這些基本概念之上，建構多租戶架構和原則層的意涵。圖 16-1 是 GenAI 體驗中一些核心組件的高度簡化視圖。

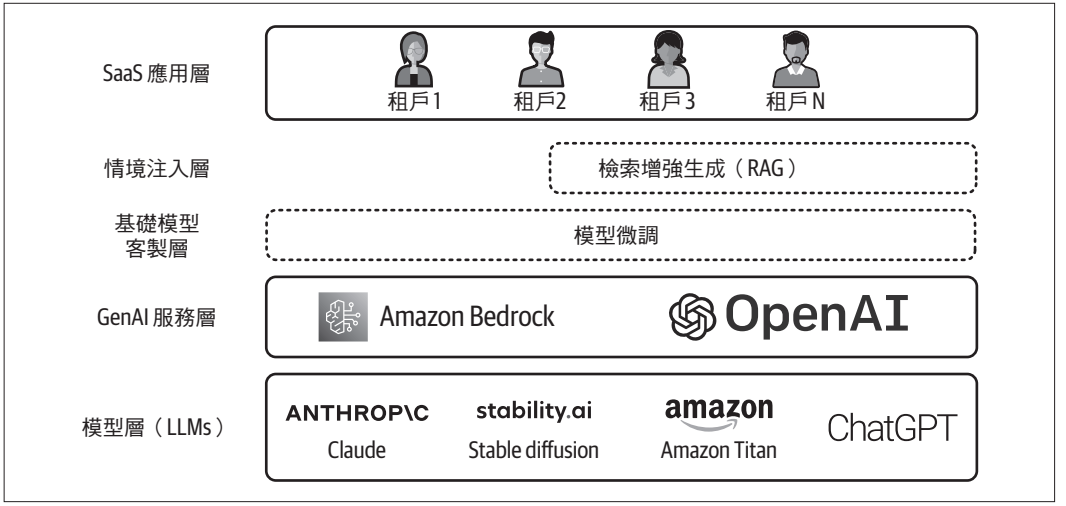

圖 16-1　GenAI 基礎架構元素

我試圖用這個圖表建構一個 GenAI 領域中關鍵元素的層級視圖，聚焦在檢視多租戶策略時將發揮關鍵作用的元素。