第二版的改動

在《The DevOps Handbook》全新修訂版中，作者對正文進行更新與編修，將全新的研究、學習和經驗納入第二版內容，加深我們對 DevOps 的理解以及它在產業的應用。此外，我們很高興邀請到知名學者 Nicole Forsgren 博士共同創作，以新的研究及相關指標豐富本書內容。

持｜續｜學｜習

我們將把從第一版上市以來的這段期間所獲得的新的見解與資源，放在全書中的「持續學習」部分，就像你在這裡看到的一樣。內容將涵蓋新的佐證資料和額外的資源、工具和技術，協助你展開 DevOps 之旅。

我們還在書中增加了更多的案例研究，藉以說明 DevOps 在各行各業的傳播程度，特別是它是如何超越 IT 部門，進入管理高層的視野。此外，在每個案例研究的結尾，我們都加上了一兩個關鍵收穫，紀錄最重要的（但絕不僅限於此）的學習收穫。最後，我們更新了每一章節的結論，提供新的資源來輔助你繼續展開學習。

DevOps 和軟體時代的下一步

如果問過去的五年教會了我們什麼，那一定就是技術無與倫比的重要性，以及當 IT 部門和業務部門願意坦誠相對時，能使企業取得多麼非凡的成就——正如 DevOps 文化所帶來的正向改變一樣。

也許沒有什麼能夠比 2020 年 COVID-19 疫情於全球爆發後，為了應對疫情而做出的快速變化更能說明這一點了。透過 DevOps 實踐，各家組織爭相靈活調動技術，在迅疾而空前的變化之中，為內部和外部客戶提供服務。這些複雜的大型

組織，往往以其無法快速軸轉和適應變化而惡名昭彰，突然間，它們別無選擇，只能紛紛擁抱這些變化。

美國航空公司（American Airlines）在當時運用他們正在推動的 DevOps 轉型規劃，迅速獲得成功，你可以在第一章和第五章見到相關討論。

你也可以在第二章讀到 Chris Strear 博士分享他使用約束理論（Theory of Constraints）來最佳化醫院流程的經驗。

2020 年，世界上最大的互助金融機構 Nationwide Building Society 得以在幾週之內迅速對客戶需求做出反應（而不是拖到好幾年之後），這全都要歸功於他們正在進行的 DevOps 轉型。你可以在第八章中讀到更多關於他們的經驗。

話雖如此，技術如此關鍵，且是成功轉型到「未來工作方式」的一大要素，但這一切必須由企業領導階層帶頭行動。今日，組織經營的瓶頸不僅僅是技術上的實踐（儘管它們依舊存在）；最大的挑戰和首要之務是讓企業領導階層一同響應。變革必須由企業和技術共同創造，而本書所提出的理論可以引領這種變革。

企業不能再故步自封，繼續非黑即白的二元思考方式：「自上而下」或「僅有技術」。我們必須實現真正的協同合作。這項工作百分之九十涉及到讓對的人參與進來、一同響應，並保持一致共識。一切就從這裡開始，以此為動力，在未來持續努力。

——IT Revolution
奧勒岡州 波特蘭
2021 年 6 月

第二版　序

自《The DevOps Handbook》第一版問世已經五年了，很多事情發生變化，同時也有許多東西保持不變。一些工具和技術不再流行，甚至不復存在，但這不應該影響到任何讀者。儘管在技術層面發生了一些變化，但本書所介紹的基本原則仍然和過去一樣重要。

事實上，今日企業對 DevOps 的需求甚至更勝以往，因為組織需要快速、安全、可靠地向客戶和用戶提供價值。為此，他們需要改造內部流程，並利用技術來提供價值 —— 使用 DevOps 實踐。這對所有組織來說都是千真萬確的真理，無論它的產業類型、無論它位於世界上哪一個地方。

在過去的幾年裡，我帶領每年《State of DevOps Reports》的研究工作（最初與 Puppet 合作，後來與 DORA 和 Google 合作），共同作者包括 Jez Humble 和 Gene Kim。這些研究證實，本書中描述的許多實踐都能帶來改進，比如：軟體發布的速度和穩定性、減輕部署工作的痛苦、減少工程師的倦怠感，以及對組織績效的貢獻，包括盈利能力、生產力、客戶滿意度、工作績效及營運效率。

在第二版的《The DevOps Handbook》中，我們以最新的研究和最佳實踐，更新主文內容，並納入了全新案例研究，分享更多關於轉型的真實故事。感謝你加入我們這個持續改善的旅程。

——Nicole Forsgren 博士
微軟研究院（Microsoft Research）合夥人
2021

導論

當開發（Dev）和營運（Ops）成為 DevOps

想像一下，如果某個世界上的產品所有者、開發人員、QA，IT 營運人員和資訊安全人員一起協同合作，這不止是為了幫助彼此，更是齊心為了讓企業組織取得商業上的成功而努力。有了這個共同努力的目標，各部門人員一同實現讓工作計畫快速投入生產的快捷流程（例如：每天產出數十、數百、甚至數以千計的程式碼部署），同時達成一流的穩定性、可靠性、可用性和安全性。

在這個世界上，跨職能團隊嚴格檢驗各種假說，找出哪一個功能最能令使用者滿意，繼而達成企業目標。他們在乎的不止是推出各種使用者功能，還要積極地確保工作流程在整體價值流中維持順暢無阻且密集頻繁的流動，極力排除混亂與中斷，避免造成 IT 營運部門或任何內外部客戶的困擾。

同一時間，QA、IT 營運及資安人員始終致力於減少團隊分歧，建立一個使開發人員更具生產力的工作系統，取得更佳工作成果。藉由把 QA、IT 營運及資安部門的專業能力注入交付團隊、自動化的自助服務工具和平台之中，每一個團隊都能在日常工作中靈活運用這些專業能力，而不需要依賴其他團隊。

這讓企業組織得以建立一個安全的工作系統，小型團隊可以快速並獨立地開發、測試和部署程式碼，快速、安全、可靠地將價值傳遞給客戶。這也讓開發人員的生產力達到最大化，同時促進內部學習、提高員工滿意度，並且在市場中佔有一席之地。

這些正是 DevOps 所締造的非凡成果。對大多數人來說，上文所描述的世界並非真實情況。我們更有可能在一個破爛的系統中工作，只能產生糟糕透頂的結果，而且完全無法展現我們的真正潛能。在我們的世界中，開發人員和 IT 營運人員永遠站在對立面，測試工作和資安活動只會發生在專案收尾的時候，想要解決問題卻總是為時已晚。而且幾乎所有關鍵活動都要求過多手動作業和過多交接，導致我們總是在等待。這不僅嚴重推遲交付時間，還會影響工作品質，特別是生產部署的工作，其產生的問題往往會製造混亂，最後對客戶和業務造成負面影響。

最終，我們無法實現當初設定的目標，整個企業組織對於 IT 人員的績效不甚滿意，導致預算被砍，員工士氣低落不振，無力改變這樣的工作流程和結果*。有沒有解決之道？改變工作方式勢在必行：DevOps 為我們指出一條明路。

為了充分體會 DevOps 革命的各種潛能，我們先來回顧一下 1980 年代的精實革命。實施精實原則和做法的製造業者，明顯地提高工廠生產力、降低從生產端到客戶所需的時間，提高產品品質及客戶滿意度，在商業市場中拔得頭籌。

在精實革命之前，製造工廠的平均接收訂單前置時間（order lead time）為 6 週，不到七成的訂單能夠準時出貨。到了 2005 年，隨著精實生產原則被廣泛實施，平均產品前置時間大幅縮短為不到 3 週，而且超過 95% 的訂單可以準時出貨。[1] 沒有採用精實生產原則的企業組織喪失市場競爭力，甚至有很多企業倒閉破產。

同樣地，科技產品和服務的交付門檻也提高了。過去幾十年來還算過得去的產品與服務，以現在的眼光看來早已遠遠不夠格。在過去四十年中，策略性商業化功能的開發和部署工作所需要的時間與成本都逐年大幅下降。在 1970 和 1980 年代，許多功能需要花上一到五年開發部署，而且通常耗資數千萬美元。

到了 21 世紀，科技不斷進步，再加上企業組織響應採用敏捷式開發原則，僅需幾週或幾個月的時間就能完成新功能開發，開發時間成本大幅度降低，但是部署到實際生產環境依然需要數週或數月，且時常伴隨著災難性後果。

---

* 這是典型 IT 企業組織中相當常見的問題之一。

到了 2010 年，人們開始接觸 DevOps 的概念，加上硬體、軟體和全面商品化的雲端服務，產品功能（甚至是整個新創企業）可以在幾週之內迅速到位，並在幾小時或幾分鐘內快速部署到生產環境。對這些組織來說，部署作業終於變成低風險的例行工作。這些組織可以針對商業創意進行實驗，找出哪些點子能為客戶和整個企業組織創造最大的商業價值，然後進一步開發成各種功能，快速且安全地部署到生產環境。

表 0.1：更快速、低廉且低風險的軟體交付發展趨勢

	1970 ～ 1980 年代	1990 年代	2000 年代至今
世代	大型電腦	客戶端／服務端	商品化與雲端服務
代表性技術	COBOL、DB2 on MVS 等	C++、Oracle、Solaris 等	Java、MySQL、Red Hat、Ruby on Rails、PHP 等
週期	1 ～ 5 年	3 ～ 12 個月	2 ～ 12 週
成本	數百萬至數億美元	數十萬至數千萬美元	數萬至數百萬美元
風險層級	整個公司	生產線或某部門	產品功能
失敗成本	公司破產、出售、大量裁員	虧損、更換首席資訊長	微不可計

（資料來源：2013 年 11 月，Adrian Cockcroft 於美國加州 FlowCon 的演講內容：「Velocity and Volume (or Speed Wins)」）

到了今日，採用 DevOps 實踐原則的企業組織通常每一天會進行數以百計，甚至是數以千計的程式碼部署。在需要儘速上市和不斷實驗才能取得競爭優勢的時代，無法複製這些成果的企業組織，註定會在市場競爭中輸給更加靈活的對手，而且可能因此一蹶不振，就像過去那些不曾採用精實開發原則的製造業者一樣。

不論身處哪一個產業，我們吸引客戶並傳遞價值的方式都必須仰賴科技價值流。奇異公司首席執行長 Jeffrey Immelt 一語中的：「任何產業、任何公司，如果沒有讓軟體成為業務核心的一部分，那麼他們注定自食其果。」[2] 又如微軟技術院士 Jeffery Snover 所言：「過去的經濟年代，企業靠著移動原子來創造價值。現在的公司改靠位元（bit）來創造商業價值。」[3]

這個課題的嚴重性不容小覷，不論哪個行業、不論什麼規模、營利或非營利，它都影響了所有組織。管理和執行技術工作的方式比起以往的任何時候，還要能夠預測出企業組織是否可以在市場上取得勝利、以及其能否存續。在很多情況下，現在的我們必須採用的原則與做法，已經迥異於過去幾十年來曾成功實踐的原則與做法（參見附錄 1）。

現在，我們已經認識到問題的急迫性，而 DevOps 可以提供解決之道。讓我們再花上一些時間，更詳細地探討這個問題的症狀：為什麼這問題會發生，以及為什麼在沒有強力干預的情況下，問題仍會隨著時間推移逐漸惡化。

難題：組織中的某項內容應當被改善（否則你也不會讀這本書）

許多組織沒有辦法在幾分鐘內或幾小時內將某個變更部署到生產環境，必須花上數週或數月時間。這些組織也無力每天部署數以百計或數以千計的變更到生產環境中。相反的，他們甚至很難每月或每季定期進行部署。對他們而言，生產部署絕非例行公事，這可能會帶來中斷危機，而且需要調度「英雄」花費大量時間進行災情搶救。

在這個講求讓產品快速進入市場、高級的服務水準、不斷進行試驗才能獲得競爭優勢的時代，上述這些組織很容易落入競爭劣勢。主要原因就在於其無力解決在技術組織內長期存在的核心矛盾。

長期的核心矛盾

幾乎所有的技術組織內，開發部門與 IT 營運部門往往存在一種與生俱來的矛盾，這容易產生惡性循環，造成許多負面影響，例如推遲新產品或新功能實際上市的時間（Time-to-Market）、產品或服務品質低落、服務中斷的機率提升，還有最惡劣的結果 —— 堆積如山的技術債。

「技術債」（Technical Debt）一詞由 Ward Cunningham 首先提出。與金融債務的概念相仿，技術債被定義為：「我們作出的決策，其所導致的問題是如何

隨著時間推移變得越發難以解決，使得未來可用的選項不斷減少。」儘管人們始終理性斟酌以進行決策，技術債的產生仍然無法避免。

造成技術債的一項因素正是致使開發與 IT 營運相互競爭的目標。技術組織負責許多任務，其中包括以下兩項必須同時追求的目標：

- 回應快速變化的市場競爭。
- 向客戶提供穩定、可靠且安全的服務。

通常，開發人員負責迅速回應市場變化，盡可能快速地將功能與變更部署到生產環境。IT 營運人員則負責向客戶提供穩定、可靠且安全的 IT 服務，讓任何人都難以或無法導入可能危及生產的變更。這樣的職責設置，導致開發部門和 IT 營運部門有著截然不同的目標和工作誘因。

製造業管理運動發起人之一的 Eliyahu M. Goldratt 博士，將這類職責設置形容為「長期的核心矛盾」—— 不同組織角色的評量標準和工作誘因相互牽制，阻礙了組織整體發展的目標。[4] *

這種矛盾會產生強大的惡性循環，導致技術組織對內對外都無法達成預期的商業表現。這些長期的矛盾衝突常常讓技術人員陷入困境，導致低劣的軟體及服務品質、客戶評價下滑，最終讓隨機應變調度「英雄」進行災害搶救這件事，成為了每日工作的一部份，這一切對於產品管理、開發、QA、IT 營運或資安部門都稀鬆平常（參見附錄 2）。

分為三幕的惡性循環

IT 產業的惡性循環由三幕組成，想必多數科技從業人員都不陌生。第一幕發生在 IT 營運部門，工作目標著眼於應用程式和基礎設施的正常運行，確保組織能向客戶傳遞價值。日常工作出現的許多問題，都是因為應用程式和基礎設施的架構複雜、紀錄不清、異常脆弱。這就是與技術債和變通辦法共處的日常，人們永

* 在製造業中，也存在類似的長期核心矛盾：必須同時保證準時交貨和成本控制的需求。附錄 2 會說明如何解決這個矛盾。

遠在承諾：「只要多給我們一點時間，保證能解決混亂。」然而，這個時間點卻永遠不會到來。

令人擔憂的是，最不堪一擊的環節正是目前支撐起整個營收系統或重要專案的關鍵角色。換句話說，最容易發生故障的系統，正是最為重要且迫切需要變更的一環。當這些變更失敗時，將會損害我們最重視的企業承諾，例如：產品供應、營收目標、客戶資料安全性、正確的財務報告等等。

當有人需要站出來，為無法履行的企業承諾負責時，惡性循環的第二幕就開始了。這個人可能是一名產品經理，承諾令客戶眼睛為之一亮的大膽功能，或者是一位商業主管，承諾更高的營業收入。他們忘記了目前技術能力範圍、或者忽略某些因素使他們與早前的承諾失之交臂，然後，他們選擇讓技術組織兌現這個新的承諾。

因此，開發部門被賦予另一項緊急專案，不可避免地需要解決新的技術挑戰，趕工抄捷徑以便滿足承諾的發布日期，進一步加重了技術債。當然，別忘了還有「只要給我們更多時間，保證修復任何問題」的承諾。

第三幕的舞台就這樣建構完成，這也是最後的一幕，在這裡，所有事情逐漸變得更加困難。每個人都變得更忙、需要更多時間完成工作，溝通協調變得更慢，任務清單變得更長。每個人的工作變得密不可分，這讓每一個小小的行動都有可能導致巨大的失敗，我們益發害怕做出改變。工作需要更多的溝通、協調和批准；團隊必須等上更久才能完成相關工作；工作品質越來越差。驅動組織的車輪轉得越來越慢，需要花上更多力氣才能繼續運轉（參見附錄 3）。

雖然這個惡性循環在當下難以想見，但是「當局者迷，旁觀者清」。退一步來看，我們可以發現程式碼的生產部署作業需要花上更久時間，從數分鐘內到數小時、從數天到數週才能完成。更糟的是，部署結果可能充滿問題，導致更大的負面影響，造成服務中斷或使用者體驗不佳，讓營運部門迫切需要調度更多「英雄」投入災害搶救，進而剝奪了償還技術債的可能性。

結果，產品交付的週期越來越長，能夠投入的專案越來越少，而且做法越來越保守。意見回饋對於每個人工作的評價是：越來越慢、越來越負面——尤其是來自客戶的意見。無論我們做了什麼嘗試，事情似乎只往更糟的方向走 —— 無法快

速回應不斷變化的競爭格局，也無力為客戶提供穩定可靠的服務。最後，我們只能向市場投降。

這種惡性循環層出不窮，一次次的經驗讓我們終於體悟：一旦 IT 失敗，整個組織都會遭殃。正如 Steven J. Spear 在著作《The High-Velocity Edge》中所述：無論傷害「像慢性疾病一樣緩緩蔓延」或是「像一場烈火熊熊燃燒」，對於組織的破壞力都非同小可。[5]

為什麼這類惡性循環隨處可見？

過去十年來，本書的作者們不斷目睹這種破壞力驚人的惡性循環發生在無數各形各色的組織中。我們比以往任何時候，都更清楚為什麼會發生這種惡性循環，以及為什麼需要 DevOps 來緩解這種情況。首先，如前文所述，任何一間技術組織都有兩個完全相反的目標。其次，任何一間公司，骨子裡都是科技公司，無論它們是否體認這一事實。

資深軟體主管兼 DevOps 最早的紀錄者之一 Christopher Little 曾說過：「每一間公司都是科技公司，不管它們認為自己身處哪種產業。銀行其實是一間擁有銀行執照的 IT 公司。」[6] * 為了更具有說服力，請仔細想想，是否絕大多數資本專案計畫都必須仰賴 IT 技術？俗話說：「幾乎不可能做出一個不涉及任何 IT 變更的商業決策。」

在商業和金融產業中，專案計畫之所以至關重要，因為它們正是組織內部改革的主要機制。專案計畫通常由管理高層負責批准、提撥預算，並為成效負責。因此，無論最後成功或失敗，專案計畫是實現組織目標和願望的機制。†

專案計畫通常以資本支出（例如：工廠、設備和大型專案等預期需要數年才能回收成本的支出，將予以資本化）的形式獲得資助，其中有 50% 的專案內容與科技相關。即使是科技相關支出最低的「低科技」產業中同樣如此，諸如能源、金

---

* 2013 年，HSBC 銀行雇用的軟體工程師比 Google 多。[7]

† 關於軟體究竟應該以「專案計畫」或「產品」的形式獲得資助，讓我們先留個懸念，留待後文討論。

屬、資源開採、汽車和建築業。[8] 換句話說，企業領袖比想像中更仰賴 IT 科技的有效管理來實現他們所設立的目標。*

代價：人性與經濟層面

久而久之，當人們被困在這種惡性循環中，尤其是處於開發下游環節的人們，他們常常會覺得陷在一種註定失敗的體系當中，而且無力改變任何結果。伴隨這種無力感而來的是倦怠，夾雜了疲憊、憤恨，甚至是無助與絕望的感受。

許多心理學家直言，建立令人感到無能為力的體系是對人類同胞最具毀滅性的事情之一。我們剝奪了其他人掌控自己結果的能力，甚至創造出一種文化 —— 因為害怕受到懲罰、害怕失敗，或者危及生計，人們不再敢勇於做正確的事情。這可能產生「習得性無助」，讓人們變得不願意或無法正確地避免同樣問題再次發生。

對員工而言，這意味著高工時、週末加班及生活品質的下降，這不只是影響員工本身，也影響到員工的家庭和朋友。所以，優質人才的會流失絕非偶然（除了那些因為責任感或義務感而覺得無法離開的人們）。

除了現有工作方式所帶來的苦難之外，我們曾有機會能夠創造出的價值，其機會成本是非常驚人的——我們認為每年損失大約 2.6 兆美元的價值創造，等同於本書撰寫時世界第六大經濟體：法國，其一年的經濟產出。

看看這項統計：國際數據資訊（IDC）和顧能公司（Gartner）預估，在 2011 年，約 5%的全球國內生產總額（3.1 兆美元）用於資訊科技（硬體、服務與電信）。[10] 假如 3.1 兆美元的 50%用在營運成本和維護現有系統，其中這 50%的三分之一被用在緊急突發和非預期的工作或重工（rework），那麼就有 520 億美元被白白浪費掉。

---

* 舉例來說，Vernon Richardson 博士及同事發表了此一驚人發現。他們研究了 184 家上市公司的財務報表，並將這些公司分為三類：（A）在資訊科技方面存在重大缺陷的公司、（B）不存在與資訊科技相關缺陷的公司，以及（C）不存在任何缺陷的「完美企業」。研究發現，A 類企業的 CEO 替換率比 C 類企業高出 8 倍，而 CFO 替換率也高出 4 倍。顯然，資訊科技比我們所認為的還要重要。[9]

如果採用 DevOps 可以幫助我們利用最佳化的管理及營運方式，將這種浪費減半，並將人力資源重新部署到產生 5 倍（這數字並非空口白話）價值的工作上，則每年可以創造 2.6 兆美元的價值。

DevOps 的使命：有一個更好的方式

上述章節闡述了長期的核心矛盾而產生的現況問題及後續影響，從無力達成組織目標，到對其他人類同胞造成的傷害等等。DevOps 能夠解決種種問題的驚人成效，讓我們能夠同時提高組織績效，實現各個職能技術角色（如開發、QA、IT 營運和資安部門等）的目標，並且改善人類現況。

這種令人振奮且罕見的可能性，解釋了為什麼 DevOps 在極短時間內就得到如此多人興奮和熱情的響應，包括科技領袖、工程師以及大多數軟體生態系統的一份子。

以 DevOps 打破惡性循環

理想情況下，小型開發團隊獨立地進行功能實作，在類生產環境中驗證可行性，並快速、安全且可靠地將程式碼部署到實際環境。程式碼部署變成例行且可預期的常規作業。程式碼的部署作業不再是從週五半夜開始，花上整個週末才能完成，而是發生在正常工作日，每一位成員都在辦公室時就能如期完成，而客戶甚至不會注意到，除非是推出新功能或錯誤修正。而且，在週間完成程式碼部署也意味著，IT 營運團隊終於能夠第一次在正常營業時間內進行工作，就像其他人一樣。

透過在工作流程的每一個階段建立快速的回饋迴路（feedback loop），所有人都可以立即見證行動所帶來的效果。無論何時將變更提交到版本控制系統，都能在類生產環境中進行快速的自動化測試，確保程式碼和環境依照設計初衷運行，並始終處於安全且可部署的狀態。

自動化測試可以幫助開發人員快速發現錯誤（通常在幾分鐘內），迅速進行錯誤修復並且促成真正的「從錯誤中學習」—— 在六個月後的整合測試時才發現程式碼出錯，對於學習毫無助益，因為這時對於程式碼的記憶和錯誤發生的因果關

係早已被拋到腦後。正確的做法是在發現問題時立刻著手修復，而不是不斷累積技術債，當全域（global）目標比區域（local）目標更為重要時，則動員整個組織一同投入。

在程式碼和生產環境中無所不在的生產遙測，可以快速偵測問題並進行校正，確保一切如常進行，客戶可以從我們所創造的軟體中獲得價值。

在這種情況下，所有人都能感受高效的生產力 —— 這種工作體系允許一個個小團隊進行安全工作，這些團隊透過自助服務平台，善用營運和資訊安全的集體經驗，在工作架構上與其他團隊的工作脫鉤。所有人不再需要等了又等，只為了處理大量被延遲的急件工作，每個團隊能夠以小批次的形式獨立工作，迅速且頻繁地為客戶提供新的價值。

即使是備受矚目的重大產品和功能發布，也能透過「暗度發布」（Dark Launching）♥ 而變成例行常規。早在正式的發布日期之前，我們就將新功能所需要的程式碼投入生產環境（真實環境），除了內部員工和小型真實使用者群之外，沒有人會發現變化。這種發布方式讓我們可以測試與持續改善該功能，直到達到預期的商業目標。

我們不再需要日以繼夜地花上長達數週的時間準備功能上線。現在只需要變更功能的切換或配置，新功能就會正常運作。這樣小小的變更，就能讓新功能出現在更大的客戶群體中。萬一出現問題，它還能自動回滾至前一版本。因此，版本發布重新回到我們掌握之中，它可以被預期、被回溯、而且負擔較小。

這不止代表功能發布更為平穩 —— 當各種可能發生的問題更小、代價更少而且更易於修復時，它們更容易被及早發現與修復。每一次的修復活動所產生的心得與收獲，可以幫助我們遏止問題再度發生，並在未來更迅速地偵測及導正類似問題。

除此之外，每個人都在不斷地學習，共同打造一種由假設驅動的工作文化。在這種工作文化中，科學方法被用來確保沒有任何事情是理所當然的 —— 必須以嚴謹的實驗態度進行開發與流程改進。

♥「暗度發布」（Dark Launch）是 DevOps 文化的獨創詞彙，意思為以不被客戶注意的方式發布新功能或修復錯誤。

因為所有人的時間都該被重視，我們不會白白花費數年開發客戶不需要的功能、部署毫無作用的程式碼，或是在根本不是問題癥結的地方糾結不已。因為我們在乎目標的達成，所以建立了負責履行目標任務的長期團隊。不再是在每個版本發布之後將開發人員再次打散、重新分配到專案團隊，對開發人員工作的回饋意見視而不見。相反地，我們保障專案團隊的完整性，讓原班人馬繼續著手迭代和改善，善用這些工作帶來的收獲，更好地實現計畫目標。這對協助外部客戶解決問題的產品團隊，以及幫助其他團隊提升工作效率、安全性及可靠性的內部平台團隊來說，情況也是如此。

我們沒有戒慎恐懼的文化，而是擁有一種高度信任、協同合作的工作文化，人們因為勇於冒險而得到獎勵。沐浴在這種工作文化的人們將無所畏懼地談論問題，而不會對問題三緘其口或視而不見 —— 畢竟，我們必須要正視問題，才能解決問題。

而且，每個人都能夠完全掌握自己的工作品質，在日常工作中建立自動化測試流程，利用同儕評閱（peer review）來確保問題在影響客戶之前就能獲得解決。這一連串流程可以降低風險，不需要來自遙遠高層的批准，我們就能快速、可靠且安全地傳遞價值 —— 甚至向抱持懷疑態度的監察單位證明，我們具備一個有效的內部控管機制。

當問題發生時，我們會採行一種「不怪罪任何人的事後驗證」（blameless post-mortems），目的不在於找出罪魁禍首，而是為了釐清問題背後的原因研擬預防對策。這種方式提升了員工的學習風氣，我們可以舉辦內部技術會議互相砥礪，磨練技能，讓每個人教學相長。

由於我們相當重視品質，為了預測系統的失敗，我們甚至會刻意將錯誤注入生產環境中進行實驗。在生產環境中進行計劃性演練，模擬大規模故障、隨機殺死程序或運算伺服器、故意造成網路延遲以及其他惡意行為，好讓程式碼更具韌性，能夠從錯誤或失誤中快速復原。因此，我們可以造就更卓越的組織韌性、促進組織學習和改善。

在這個世界上，每個人對自己的工作擁有掌控權，無論在科技組織中的角色為何，他們都相信自己的工作很重要，能夠為組織目標做出有意義的貢獻，這件事可以從低壓力的工作環境和所在組織的商業成功加以證明。而所謂證明正是，該企業組織確實在市場上取得非凡勝利。

DevOps 的商業價值

我們手上握有證明 DevOps 具有商業價值的決定性證據。從 2013 ～ 2016 年，身為 Puppet Labs 的《State of DevOps Reports》的一份子，本書作者 Nicole Forsgren、Jez Humble 和 Gene Kim 蒐集了來自 25,000 多名科技專業人士的資料，試著了解在採用 DevOps 之後，企業組織在每一階段的健全程度和工作習慣。*

這份研究資料帶來的第一份驚喜是採用 DevOps 工作方式的高效能組織，在以下領域的績效表現遠遠勝過其他企業組織：[11]

- 生產量指標
  - 程式碼與變更的部署次數（更頻繁 30 倍）。
  - 程式碼與變更的部署的發布時長（更快 200 倍）。
- 可靠性指標
  - 生產部署（變更成功率更高出 60 倍）。
  - 恢復服務所花費時間的中位數（更快 168 倍）。
- 組織績效指標
  - 生產力、市場佔有率、目標利潤（有 2 倍可能性超越）。
  - 市值成長（三年內成長 50%以上）。

換句話說，高效能組織變得更敏捷、更可靠，研究實證顯示 DevOps 能幫助我們打破長期的核心矛盾。高效能組織部署程式碼的次數多了 30 倍，從「提交程式碼」到「成功在生產環境中運行」所需的時間快了 200 倍 —— 高效能組織的交付週期以分鐘或小時計算，而低效能組織的交付週期卻是動輒數週、數月，甚至以季度計算。

此外，高效能組織達成甚至超越目標利潤、市值成長和生產目標的可能性多了兩倍。而且，在會發放股票給員工的組織中，我們發現高效能組織在三年內市值成長率會超過 50%。這些組織內員工對工作的滿意度更高、職業倦怠率更低，

* 《State of DevOps Reports》每一年都會發表相關研究結果。此外，2013-2018 年度報告的關鍵發現也被收錄於《ACCELERATE：精益軟體與 DevOps 背後的科學》一書。

而且更樂意向親友推薦他們所在的公司是一間優質工作場域，其可能性多了 2.2 倍。† 高效能組織的資訊安全維護成果也更為卓越，將安全目標整合到開發和營運流程的每一階段，處理安全問題的時間可以降低 50%。

DevOps 規模化開發人員的生產力

當我們增加開發人員的數量時，通常會因為工作上的溝通、整合和測試，導致每一位開發人員的生產力明顯地降低。

Freferick Brook 的經典著作《人月神話：軟體專案管理之道》（The Mythical Man-Month）中特別強調了這個狀況。他解釋：當專案計畫的進度落後時，增加更多開發人力不僅會降低開發人員的個人工作效率，更會削弱整體生產力。[13]

另一方面，DevOps 告訴我們，當具備正確架構、正確的技術實踐和正確的文化規範時，小型開發團隊能夠快速、安全、獨立地進行開發、整合、測試和部署變更到生產環境中。

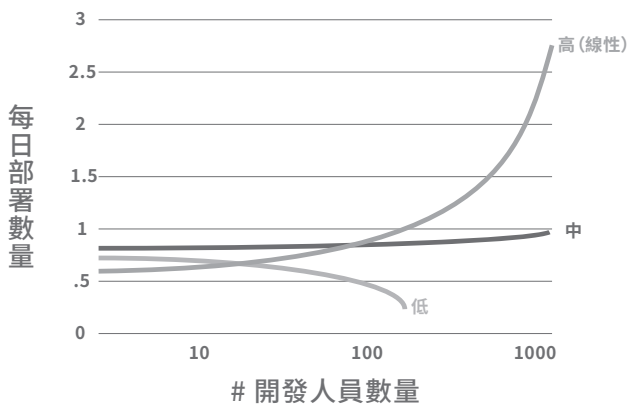

圖 0.1　每日部署數量 vs. 開發人員數量

僅計入每日部署至少 1 次的組織。
（資料來源：Puppet Labs, 2015 *State of DevOps Report*）

† 依照員工淨推薦人值（eNPS）衡量。這是一項顯著發現，因為研究顯示「擁有高度員工參與度的公司，其收入成長是低員工參與度的公司的 2.5 倍。從 1997 ~ 2011 年間，工作環境信任度較高的公司，其（公開交易）股票表現優於市場指數 3 倍。[12]

正如前任 Google 工程總監，現任 eBay 工程 VP 的 Randy Shoup 觀察到的，採用 DevOps 的大型組織「擁有數千名開發人員，但組織架構和實踐方法讓小型團隊仍然具有難以置信的高度生產力，就像剛剛起步的新創企業一樣。」[14]

《2015 State of DevOps Report》不僅檢驗「每日部署量」，也一併將「每開發人員每日部署量」列為觀察要點。該研究報告假設：隨著團隊規模擴大，高效能組織有能力將部署數量規模化。[15]

的確，這個假設可以從資料中得到驗證。圖 0.1 顯示，在低效能組織中，每位開發人員的每日部署量隨著團隊規模擴大而下降，在中效能組織中維持不變，而對於高效能組織來說，數據則以線性成長。換句話說，採用 DevOps 的企業組織有辦法在增加開發人員數量的同時，讓部署數量線性成長，比如：Google、Amazon 和 Netflix。*

解決方案的普遍適用性

精實生產運動中最具影響力的書籍之一是 Eliyahu M. Goldratt 博士撰寫於 1984 年的《目標：持續改進的過程》。它的影響力遍及全世界，深深影響了整個世代的專業工廠經理。這是一本關於一位工廠經理的小說，他必須在 90 天內解決成本和產品交貨等經營問題，否則他的工廠將面臨倒閉。

在職業生涯後期，Goldratt 博士提到了關於《目標》一書的讀者來信。這些信上通常寫道：「你顯然隱身於我們的工廠之中，因為你正確無誤地描述了我（身為工廠經理）的生活……」更重要的是，這些來信顯示人們有能力在自己的工作環境中，如法炮製書上所描述的績效突破。

《鳳凰專案》由 Gene Kim、Kevin Behr 和 George Spafford 於 2013 年撰寫，是一本以《目標》為原型的小說。這本書的主角是一位 IT 經理，他面臨在技術組織中普遍存在的所有典型問題：一份嚴重超出預算、進度大幅落後的專案，必須盡快進入市場才能維持公司生計。他經歷了多災多難的部署作業，必須解決可用性、安全性和合規性問題等等。

---

* Amazon 是一個更加極端的例子。在 2011 年，Amazon 每日進行將近 7,000 次部署。到了 2015 年，每天部署數量來到 130,000 次。

最終，這位 IT 經理和他的團隊採用了 DevOps 原則和做法，克服重重困難，幫助組織在市場中贏得勝利。除此之外，這本小說還顯示了 DevOps 文化如何在過程中吸引更多人一同參與，創造出壓力更低、滿意度更高的團隊工作環境。

與《目標》一書一樣，大量證據顯示《鳳凰專案》所描述的問題和解決方案具有無可否認的普遍性。看看這些在 Amazon 評論的句子：「我可以和《鳳凰專案》的許多角色產生共鳴……在我的職業生涯中也遇到多數工作困境與難題。」、「如果你曾經做過任何 IT、DevOps 或資安相關工作，你絕對會對這本書心有戚戚焉」、或者「《鳳凰專案》書中出現的所有角色，沒有一個角色無法對應到自己身上或真實生活周遭的人物……更不用說這些角色要面臨和克服的種種問題。」

《The DevOps Handbook》：不可或缺的指導手冊

在本書的餘下內容，我們將會介紹如何如法炮製《鳳凰專案》中的轉型活動，並提供許多案例研究，了解其他組織如何使用 DevOps 原則和實踐做法來複製這些結果。

《The DevOps Handbook》將為你提供成功啟動 DevOps 計畫並實現預期結果所需的理論基礎、原則與實踐方法。這本指導手冊的完成，基於數十年來的管理學理論、關於高效能科技組織的研究、幫助組織轉型的工作、以及驗證 DevOps 工作方式之有效性的調查研究，再加上訪談無數位與主題相關的專家名士，以及在 DevOps Enterprise Summit 上提出的近百份案例研究之分析。

本書分為六個部分，以「三步工作法」哲學闡述 DevOps 理論和原則。「三步工作法」是《鳳凰專案》基礎理論的實作哲學。《The DevOps Handbook》適合所有位於技術價值流（通常包括產品管理、開發、QA、IT 營運和資訊安全）的從業人員，以及發起大多數 IT 計畫的業務及行銷主管。

不需要深入了解對 DevOps、敏捷式開發、ITIL、精實生產或流程改善，你就能輕鬆閱讀這本書。這些主題將在必要時於書中介紹並詳細說明。

本書目的是針對每一個領域的關鍵概念建立知識架構，既可以作為入門讀物，還能幫助從業者學習如何在整個 IT 價值流中，以共通的語言與人進行合作，制定共同工作目標。

對於越來越仰賴技術組織來實現企業目標的商業領袖和利益相關者來說，本書也是不可錯過的優質讀物。

此外，本書當然也適合那些所在組織可能沒有遇到本書描述的所有問題的讀者（例如：漫長的部署前置時間或痛苦磨人的部署作業）。即使是這類幸運的讀者，也能藉由掌握 DevOps 原則而受益匪淺，尤其是與共享目標、意見回饋和持續學習相關的原則。

在第一部分中，我們簡單介紹了 DevOps 的歷史由來，並敘述由橫跨數十年的相關知識體系而積累的基礎理論和關鍵主題。然後，我們介紹「三步工作法」的大方向原則：工作流、回饋循環、持續學習與實驗。

第二部分描述該從何處下手、該如何開始，並提出了諸如價值流、組織設計原則和模式、組織採用模式等概念與案例研究。

第三部分講述如何為部署流水線建立堅實的基礎來加速「工作流」：實現快速有效的自動化測試、持續整合、持續交付，追求以低風險的方式進行發佈。

第四部分著重討論如何加速及強化「回饋循環」，透過有效的生產遙測來偵測並解決問題，更準確地預測問題並實現目標，啟動回饋機制，讓開發人員和營運人員可以安全地部署變更，將 A/B 測試整合到日常工作當中，建立評閱和協調機制來提昇工作品質。

第五部分則描述如何藉由建立公正的文化，擴大「持續學習」的影響力，將區域發現轉化為全域改善目標，同時適當分配工作時間，鼓勵組織學習探索，進行改善活動。

最後，第六部分描述將安全性和合規性整合到日常工作的適切做法。諸如，將預防性安全控制整合到共享原始碼庫和服務中、將安全性整合到部署流水線中、強化遙測功能以便更快偵測錯誤和復原、維護部署流水線，以及達成變更管理的目標。

將這些實踐作做法錄下來，我們希望能激發更多組織採用 DevOps 工作方式，提高 DevOps 計劃的成功機率，鼓勵更多人群起響應，提升邁向 DevOps 轉型的先決動能。