海峽兩岸信息產業和技術標準論壇
技術報告彙編

開發維運一體化兩岸共通標準研究報告
DevOps Cross-strait Common Standard Research Report

中國電子工業標準化技術協會　中國技術標準化研究院
華聚產業共同標準推動基金會　台灣雲端物聯網產業協會
共同公布

目 錄

前言

雲計算作為資訊技術領域的一種重大創新應用模式，是戰略性新興產業的重要組成部分，為推動兩岸雲計算標準融合與共通，促進兩岸雲計算產業合作與發展的理念，自2013 年 10 月「第十屆海峽兩岸資訊產業和技術標準論壇」開設雲計算分論壇以來，在兩岸雲計算專家共同努力下，雙方在合作機制、標準、產業、開源社群等方面進行了廣泛溝通，成功牽引和帶動兩岸雲計算產業及標準務實合作。

在 2017 年第十四屆活動中，許多與會學者專家和產業代表一致認為，雲計算發展至今已扭轉整個世界、產業、消費習慣甚至生活形態，而背後支撐雲計算相關軟體的重要性已不可同日而語，不僅默默支援業務，更成為拓展業務的關鍵要素，也被賦予成重塑企業軟體發展流程和組織文化的神聖使命，而 DevOps 的出現，改變了企業開發與交付軟體的方式，打破了開發和營運團隊的文化隔閡與壁壘，被視為當今企業轉型的有力工具，包括 Amazon、Facebook、Netflix 等國際企業均因採用 DevOps 流程大幅提升市場競爭力。

為推動推動兩岸在 DevOps 方面的相互瞭解與合作機會，海峽兩岸雲計算分論壇組成海峽兩岸雲計算工作組，組織雙方企業和專家，在今年首次針對這個議題共同編著了《開發維運一體化（DevOps）兩岸共通標準研究報告》，邀集了兩岸知名且具代表性的企業案例分享，從理論、實務與標準等方面加以探討，希望提供兩岸產業界參考，希望透過此工作，對接海峽兩岸產業合作需求，推動兩岸合作實質落地。

第一章 DevOps 簡介

軟體工程發展史上，工程師們為了解決各時代不同軟體開發環境和效率、品質與速度問題，不斷推出相關軟體開發標準和規範，譬如早期 ISO9001 進化到 CMMI，或從瀑布式開發演變到今日的敏捷（Agile），其目的皆與軟體開發程式能與時俱進，滿足當代產業脈動需求。近年，網際網路和雲端服務大規模普及，徹底顛覆了由來以往的競爭格局，客戶需求變化多端，商業模式日新月異，以行動 APP 為核心市場成為兵家必爭之地，在此產業發展背景之下，企業軟體開發速度和品質受到空前重視，而特別能滿足此類產業需求的 DevOps 方法便應運而生。

市場對於 DevOps 的發展和應用雖不陌生，但各方定義目前卻仍未江山一定，原因是現階段尚未出現正式的 DevOps 國際標準（ISO 與 IEEE 皆草案制定中）。本研究報告綜整目前全球 IT 企業如 Amazon、HP、IBM、微軟等以及 Garner 等知名研究機構對 DevOps 的定義加以融合為：「DevOps 象徵 IT 文化的轉變，著眼於採用敏捷、精實的系統化作法，達成快速的 IT 服務交付，其本質是一種分工，強調在人群與文化層面，試圖促進開發和營運團隊的合作，亦即透過開發、測試、維運等角色職責的分工，來實現工程效率最大化，進而滿足業務需求。」

1.1 創新思維與革新

傳統企業分工模式無法反應新業務工作需求是 DevOps 崛起的原因之一，DevOps 提出了許多創新概念和實施方法，消除了這些因為重複性工作和資源不均導致的效率低落問題，DevOps 分工和傳統模式的區別如圖 1-1 所示。

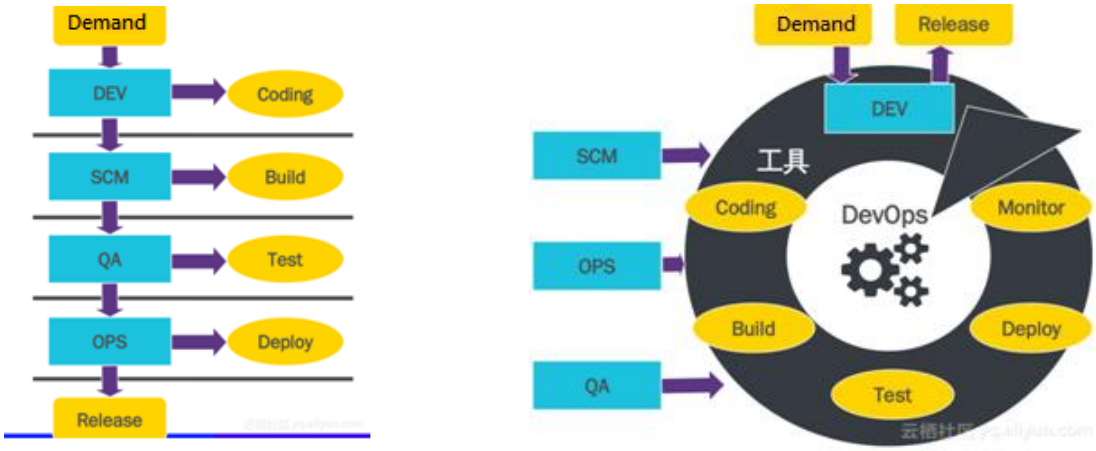


圖 1-1 DevOps 分工和傳統模式的區別

圖片來源: http://www.sohu.com/a/146111672_403354

舉例來說，過去軟體研發通常將開發、IT 維運與 QA 品質保障，設為各自獨自的部門，開發部門的驅動力是頻繁交付新功能，因此開發和佈署無需考慮 IT 支援或 QA 跨部門支援，而維運部門則關注在 IT 服務的可靠性和運作效率，各部門目標彼此衝突、時而相擊，久而久之，開發和維運部門產生一道鴻溝，拖累整個企業交付業務的速度，可能衝突情境如下：

- 開發人員往往沒有考慮程式對維運造成的可能影響，在交付程式之前，自然也不會邀請維運人員參與架構決策或程式評審；
- 開發人員修改配置或環境後，並未及時告知維運人員，導致新的程式無法執行；
- 維運人員對應用程式缺乏瞭解，不易正確選擇執行環境和發佈流程；
- 維運人員希望儘量避免修改功能，降低異動可能引發的宕機風險；
- 決策階層不瞭解團隊工作狀況，業務部門亦無法掌握需求被處理的進度；
- 開發團隊工具不同，不同團隊溝通不易。

上述場景並不罕見，因為在多數企業裡，應用程式發佈是一項涉及多團隊、風險高、壓力大的任務，各團隊立場很容易針鋒相對。但 DevOps 提供協同工作的流程和方法，搭配自動化工具，來打破不同部門之間的壁壘，打通以前曾是瓶頸的每個環節，大幅減

少甚至消除這些障礙，主要差異在於：

- 先以價值產出為目標導向，再向下展開流程和方法；
- 以應用程式為中心來理解基礎設施；
- 定義簡潔明瞭的流程；
- 更小、更頻繁的變更；
- 讓開發人員有更多權限能控制生產環境；
- 盡可能自動化；
- 促成開發與維運協作。

有些專家認為 DevOps 是敏捷（Agile）和精實（Lean）開發概念的延伸，主要在打破過往封閉迴路，從需求分析、系統設計、程式開發、安裝測試、系統維運的每一個獨立的階段，要求開發人員、維運人員等盡可能以自動化方式執行任務，例如由工具進行自動化測試、自動化佈署，減少手動及傳遞或等待的時間，避免人為錯誤，改善軟體交付品質，另外，也將自動化相關資料提供給所有參與的人，根據量化的資料加以滾動式改進。DevOps 以頻繁更新降低系統風險如圖 1-2 所示。

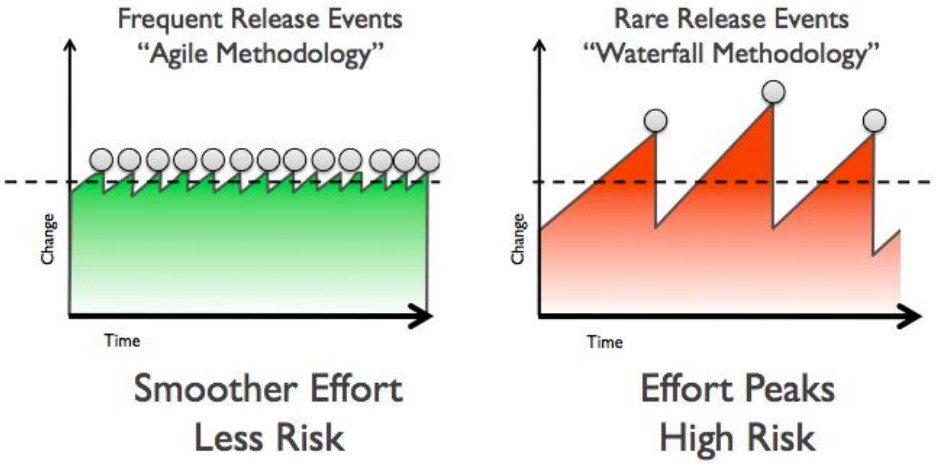


圖 1-2 DevOps 以頻繁更新降低系統風險

圖片來源:維基百科 https://upload.wikimedia.org/wikipedia/commons/1/1c/Agile-vs-iterative-flow.jpg

從一些導入案例來看，實施 DevOps 確實能對企業生產起巨大作用，以開發佈署速度為例，傳統開發週期常用大規模、不頻繁的發佈，所以常常以「季」或「年」為單位發佈，改用 DevOps 或迭代式開發後，大幅縮短成為以「天」或「周」為單位週期，1 年發佈 18 個版本，甚至 1 天發佈 18 次都不令人意外。DevOps 用持續頻繁的發佈，每次發佈變化自然相對變少，每次佈署也就不會對生產系統造成巨大影響，因此應用程式會以平滑的速率逐漸增生，也讓企業不用擔心系統崩潰和服務失效的風險。

1.2 未來發展前景

目前許多企業已經開始關注 DevOps，不論是從交付和佈署流水線的自動化起步投入小規模試點，或從底層基礎架構的容器化開始探索這塊領域，都看中它在扭轉傳統軟體開發時過於片段與無法快速回饋需求改變的缺陷，不過，從整體來看全球除了大型聯網業者之外，其他還是處於比較前期的嘗試階段，缺乏大規模、系統性的導入。

總結而言，在傳統軟體開發和基礎設施程式管理的組織，其內部開發、維運和測試經常互不相通，但透過導入 DevOps 科技化方法、自動化工具等，就可以更快、更有效率的開發和改進，讓組織能對客戶提供更快、更好的服務。因此，即使有些專家認為雖然它不是一個具體的技術，但仍對它的未來發展寄予厚望，並認為將進一步結合新工具和新技術後迅速在市場普及。

第二章 DevOps 發展情況

DevOps 興起並非一朝一夕，事實上，在歷經近 10 年的進化改進後，DevOps 的含意已不再只是單純字面意思的開發維運一體化，而具有組織文化變革的意涵，成為貫穿產品與軟體研發生命週期，包括縱向打通需求、設計、開發、編譯、建構、測試、佈署、維運，橫向打通架構、開發、測試、品管、維運、營運等概念。

2.1 萌芽與源起

一般認為 DevOps 之所以興起，在於進入雲端時代，企業核心商業行為與網路密不可分，大量的應用服務將載體遷移到雲端，使過去網路底層架構、中介軟體和環境設定工作承載變得不再吃重，取而代之的挑戰是如何滿足快速市場變化與需求，推出對應的獲利服務，對這些如同企業命脈的應用服務，勢必要加速開發、不斷更新、持續交付，速度等同商機，所以網路大腕級企業和領域專家的軟體工程師們，都相繼提出各自的解決方案比拚。

時間回溯 2008 年，全球資訊科技產業出現一些重大訊息，像是：金融海嘯引發經濟危機、微軟擬以 446 億美元收購雅虎遭拒、Google 推出手機專用的 Android 平台、全球最大的垃圾郵件發送商 McColo 遭斷線懲處、以及歐巴馬運用網路科技當選美國總統，讓人們體認網際網路對當代局勢影響的重要性。

場景轉到加拿大多倫多敏捷會議（Agile Conference），Patrick DeBois 和 Andrew Shafer 兩位專家對傳統軟體發展流程感到挫折，故深入討論敏捷架構的發展性，業界人士咸認這場會議讓 DevOps 概念萌芽。次年，兩位 Flickr 工程師 John Allspaw 和 Paul Hammond 在美國加州 O’reilly Velocity 大會發表「1 天佈署 10 次」引發熱烈迴響，而此一演說啟動了 Patick DeBios 在同年 10 月於比利時根特市創辦全世界第一場 DevOpsDays 活動，當時為了要讓消息能在 twitter 上快速傳播，許多人將 DevOpsDays 縮寫簡化成「DevOps」，

DevOps 一詞就此正式誕生。

自此之後，DevOps 經常變成各大 IT 論壇和演講焦點議題，迅速在世界各地蔓延，至今全球各地 DevOpsDays 已舉辦超過 60 場，若再加上其他與各種形式相關討論或分享更為可觀，意味有愈來愈多的人對這個詞所包含的理念與實踐，有非常深刻的共鳴。

經過數年的產業倡議，DevOps 發展日趨成熟，在 2010 年美國山景城 DevOpsDays 活動中，Damon Edwards 提出以「CAMS」來詮釋 DevOps，即文化（Culture）、自動化（Automation）、度量（Measurement/Metrics）和分享（Sharing），之後，又有 Jez Humble 把原本用於豐田生產方式的精益（Lean）管理原則加以轉變並融合其中，變成「CALMS」，其精神更能抓住 DevOps 的深意，即除了技術之外，還有管理與組織文化的議題，也就是人的問題，概述如下：

- 文化：指組織文化應勇於變革，促進協同工作與溝通；
- 自動化：指盡可能降低價值鏈中可能存在的人為干擾環節；
- 精益：指及時製造（開發），消除一切浪費，利用快速推出逐步改善的方式強化產品的彈性；
- 度量：指透過量測和監控取得資料，並透過數據改善循環週期；
- 分享：指開放與他人分享成功或失敗經驗，以不斷學習。

2.2 淬鍊與演進

DevOps 工具對其快速普及扮演了很重要的推手，不論是商務軟體或開源軟體，企業能夠選擇使用的工具，早期只有單純建構、佈署、維運階段的個別方案，到今天細化成可以分別支撐建構、持續整合、持續交付、配置管理、日誌紀錄、監控、協同運作、測試等不同流程的工具包，整合出幾乎等同全方位解決方案，使用這些工具的成功案例，又被當成新進企業的工作指引，協助企業挑選和搭配符合自己公司自動化程式無縫接軌

的落地方案。

早期 DevOps 工具如 Saltstack，曾經嘗試把部分 DevOps 概念具象出一些基本的功能和介面，但真正投入實際運作過程中，會消耗大量資源，而且需要自行加工，例如自己再撰寫一些程式，才能達到期望的自動化功能，而且有時需要安裝在用戶端，增加了佈署的難度，這些都讓早期 DevOps 發展沒有那麼順利，直到最近，許多工具在工具大廠和開源社群的集體創作之下，有了比較突破性的發展，也讓 DevOps 的接受度快速打開，以下列討論度較高的兩種技術工具為例：

- 容器（Container）

過去在軟體發展流程中，對於執行軟體的作業系統和網路環境相當依賴，而容器出現後，它運用虛擬化「應用程式及其相對應的環境」技術，從根本上解決了軟體對環境依賴的問題，在整個應用程式生命週期工作流程中，提供隔離、可攜性、彈性、延展性和控制能力，以及為開發與作業提供隔離。

尤其 Docker 出現後，這種具有輕量化優勢的容器技術，帶來高可用性和易用性，讓 DevOps 的普及向前躍進。根據 VMfive 的調查，2013 年以前，只能使用 Chef 或 Puppet 等架構厚重、使用不便的容器技術，所以企業導入不易，直到 Docker、Kubernetes 等輕量化容器出現後，再加上工具大廠適時推出各種可雲端託管的容器管理服務，讓 2017 年全球採用容器技術的企業大爆發，一舉超過 70%，也吸引愈來愈多企業願意採納 DevOps。

- 微服務（Microservices）

傳統應用程式開發架構多半是單體式應用程式或三層架構，而不管哪一種，都越來越難應付大規模佈署、服務不中斷，龐大的架構與無窮無盡的程式碼成為開發上的沉重負擔。微服務架構出現後，它以單一責任與功能的小功能區塊為基礎，彼此透過 API 通訊來建構出大型的應用程式，這種方式由於職責單一、程式碼少，不僅提高開發速度，

亦可達到獨立佈署、調校與測試的目的，讓大型服務能透過快速水準擴充易於應付突發性流量暴衝的問題，這也是 DevOps 發展的重要助力之一。DevOps 相關工具如圖 2-1 所示。

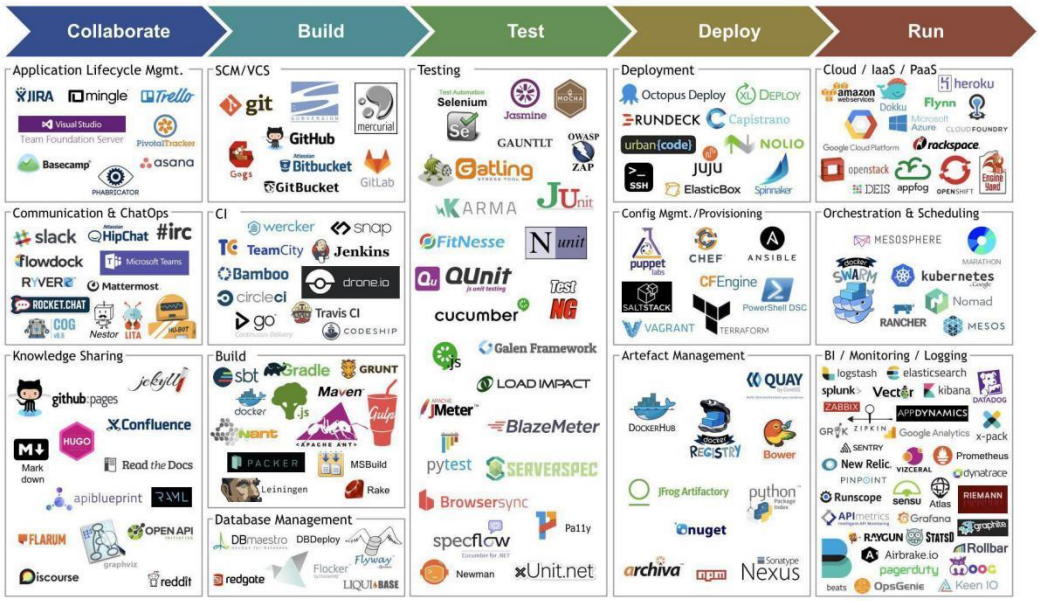


圖 2-1 DevOps 相關工具

資料來源: Continuous delivery tool landscape · James Bowman

http://www.jamesbowman.me/post/continuous-delivery-tool-landscape/

愈來愈多支撐 DevOps 工具問世，讓企業對 DevOps 的觀念和技術愈來愈容易採用和熟悉，協助企業成功重整原本在軟體發展遭遇的破碎流程和工具雜散問題，甚至把開發/維運等傳統企業戰略位階較低的工作，提升到與企業策略同級，與戰術目標緊密結合。

2.3 適用類型漸增

適用對象方面，過去有些專家認為 DevOps 比較適合大型網路服務業者，譬如 Amazon、Google、Netflix、阿里巴巴這種巨型商業公司，論點在於此類企業擁有大量優秀的工程師，也有充沛的資源可以解決導入 DevOps 時遭遇的各種困難，所以可以專注改善內部流程和優化。但根據由 Puppet、DORA、與雲霽科技合作發表的「2017 年 DevOps 現況調查報告」發現，DevOps 適用於所有類型的組織，說明其更快、更好地進行研發與佈署

軟體，實現組織價值。而另一份 Interop ITX 於 2017 年對 Devops 狀況調查結果則顯示，企業對於實施 DevOps 後得到的效果，第一名是佈署的應用表現和品質，有 70%以上認為確實有提升或顯著提升，而其他除錯和維護應用時間也有縮短，增加軟體/服務的佈署頻率亦有提升，部門之間的協同合作等都有超過 50%的人表示肯定。

在 Gartner 一份針對 2017 年之後應用程式開發（Application Development）的研究報告資料指出，今後從事應用程式開發的企業中，將有 35%從原本使用的 Scrum 改變用敏捷/精益為基礎的軟體發展方法;而至 2020 年,預估將有 50%的企業會採用 DevOps 方案，並以開源工具進行持續檢測，讓 IT 組織能在穩定、近似於實際生產環境的情境中執行定期測試；另外將有 50%的企業採用先進的分析技術，提升應用程式的品質和交付速度。Lean-Agile IT 最低限度的建構組成如圖 2-2 所示。

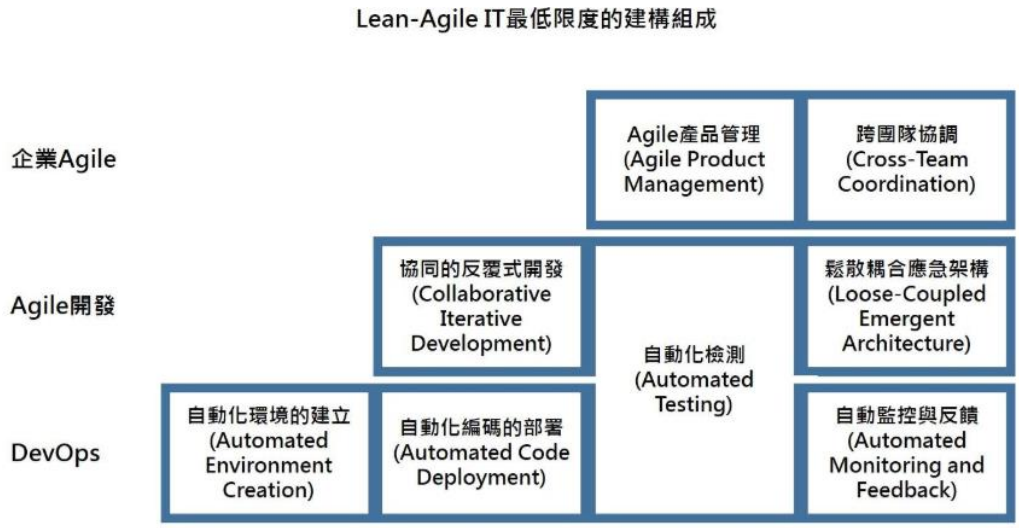


圖 2-2 Lean-Agile IT 最低限度的建構組成

資料來源:Gartner/科技發展觀測平台整理 https://outlook.stpi.narl.org.tw/index/focusnews/detail/307

事實上，DevOps 接納度之所以快速提升，原因之一為實證效益，有愈來愈多工具使用和導入之後，對於公司的效率或其他目標有顯著的提升，因此累積愈來愈多好評和口碑，就會吸引愈來愈多企業投入。

DevOps 未來除了在既有技術基礎優化或創新之外，將轉向創新分析技術，即結合

大數據或 AI 人工智慧等技術，從企業龐大的巨量資料中探勘挖掘，產出有價值的情報，超越現今單純提供戰情資料回報功能而已，協助企業增加競爭力，這也象徵 DevOps 未來的應用範圍有機會再進一步擴大。

2.4 組織文化與管理能力

經過多年發展，DevOps 概念、技術、工具、架構和方法已逐漸成熟，例如導入方法上可從時間、品質兩大目標著手改進，只要先抓住這兩個最高位階的戰術目標後，再細化後續展開的各項任務，就容易合理化相對應的各項工作，而從許多市場研究報告和產業分享案例中也能發現，企業在導入 DevOps 過程中，若先著眼於最終目標再細化各流程步驟時是比較容易成功的，幾乎不會有技術阻礙，反而對於管理能力的要求成為最艱難的部分，亦即技術層次最容易實踐，流程其次，人的問題才是最難的最後一里。

完整的 DevOps 是一個龐大的體系，和企業的組織架構、技術、流程、文化息息相關，並且涉及組織改造，而只要改造就有受到影響和波及的物件，因此任何革新都不能忽略組織的反彈和人性的抗拒。

部分專家認為 DevOps 在某種程度上只是一組相互鏈結的技術實踐，所以企業採納有不同適應，雖然遵循相同的規則，但最終會拼貼出屬於自己企業文化和流程的藍圖。雖然拼圖不盡相同，但在產業實務發展上，仍有許多組織偏好和習慣能有一個管理框架去實現這些不論是組織文化或流程方面的變革，因此本研究報告在第三章邀請海峽兩岸在 DevOps 實踐有成功經驗的企業，提供導入歷程或方法，希望這些案例能讓有意採納 DevOps 的企業更有信心或縮短學習曲線。另外，本研究報告也嘗試從國內外案例與調研資料中嘗試提煉出一些發展特點及可行的標準框架，來提供更多思維面向，讓 DevOps 實踐能夠更為容易。

第三章 DevOps 應用場景

3.1 研華—WISE-PaaS/EnSaaS 物聯網雲平台 DevOps 實踐

3.1.1 案例簡介

（一）案例背景

在物聯網的帶動下，製造業正迎來新一輪變革浪潮，雲計算、大數據、人工智慧等新技術正在加速與工業領域的全方位融合。縱觀全球市場，工業 4.0 趨勢所向，各國的企業都在製造領域中尋找新的經濟成長契機。自從 2015 年大陸公佈《中國製造 2025》計畫以來，大陸的工業轉型正迎來大突破、大升級。《中國製造 2025》的核心關鍵是為了打造智慧化、網路化生產系統的「智慧工廠」。物聯網、大數據、人工智慧和實體經濟深融合，在工業的應用下已迎來蓬勃發展時期。

作為最早工業電腦行業的廠商之一，研華意識到物聯網的發展趨勢將給世界帶來巨大的變化，致力於充當推手的角色來推動物聯網產業的發展。研華為建構工業物聯網之完整價值鏈，自 2017 年起與資策會協同開發 IoT PaaS 物聯網雲平台，並導入 DevOps 的開發流程，以加速從端到雲（Edge, PaaS, SaaS）之串接與維運服務。雙方將透過研華在 Edge 端既有的硬體系統優勢、共同開發之 WISE-PaaS 2.0 資料分析平台，以及開放各垂直領域第三方單位之 SaaS 服務於該平台上進行開發，以布建完整工業設備聯網之維運雲端服務平台。

（二）案例特點

WISE-PaaS 工業物聯網雲平台，以下簡稱為「WISE-PaaS 雲平台」或「雲平台」，是一個整合的物聯網服務平台，旨在從邊緣到雲端提供可操作的洞察力。讓使用者能夠輕鬆安全地連接、管理和吸收大規模的物聯網數據，即時處理和分析/視覺化數據。憑藉全套開發工具，WISE-PaaS 簡化了物聯網解決方案佈署，使資源可集中在關注的專業領

域。WISE-PaaS 2.0 平台架構如圖 3-1 所示。

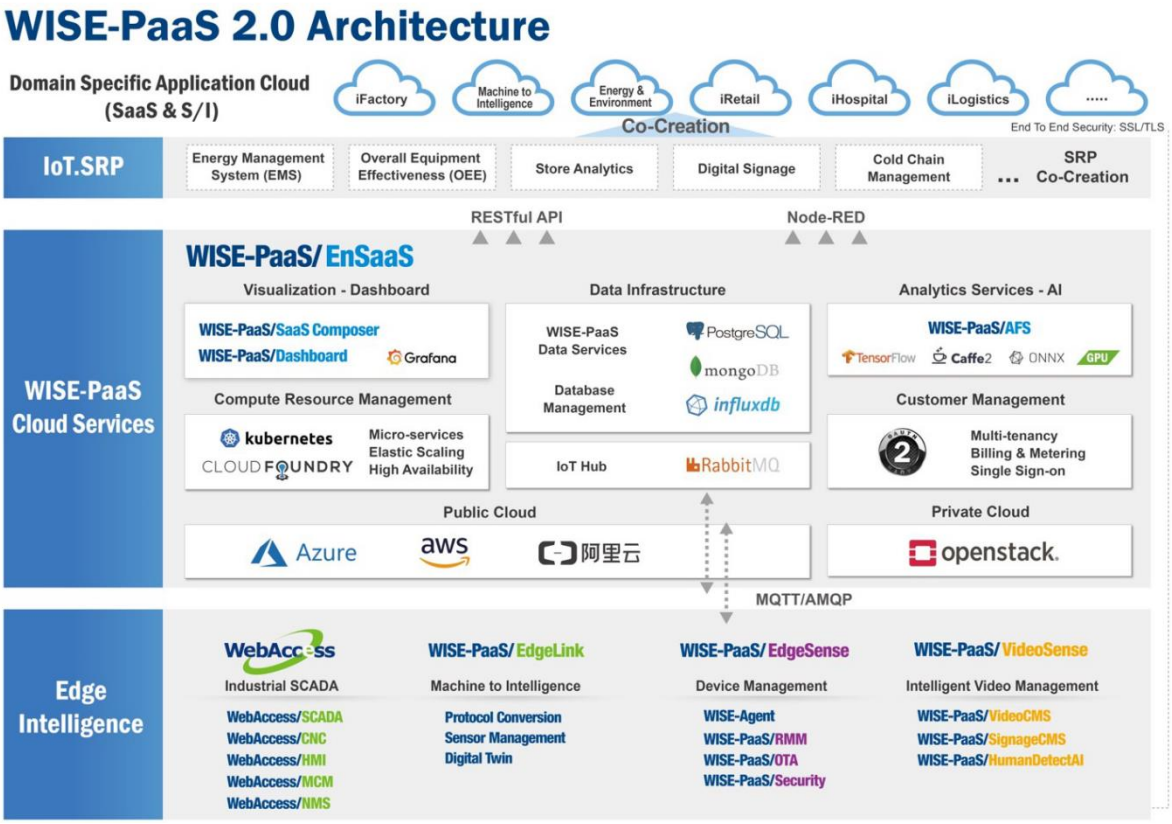
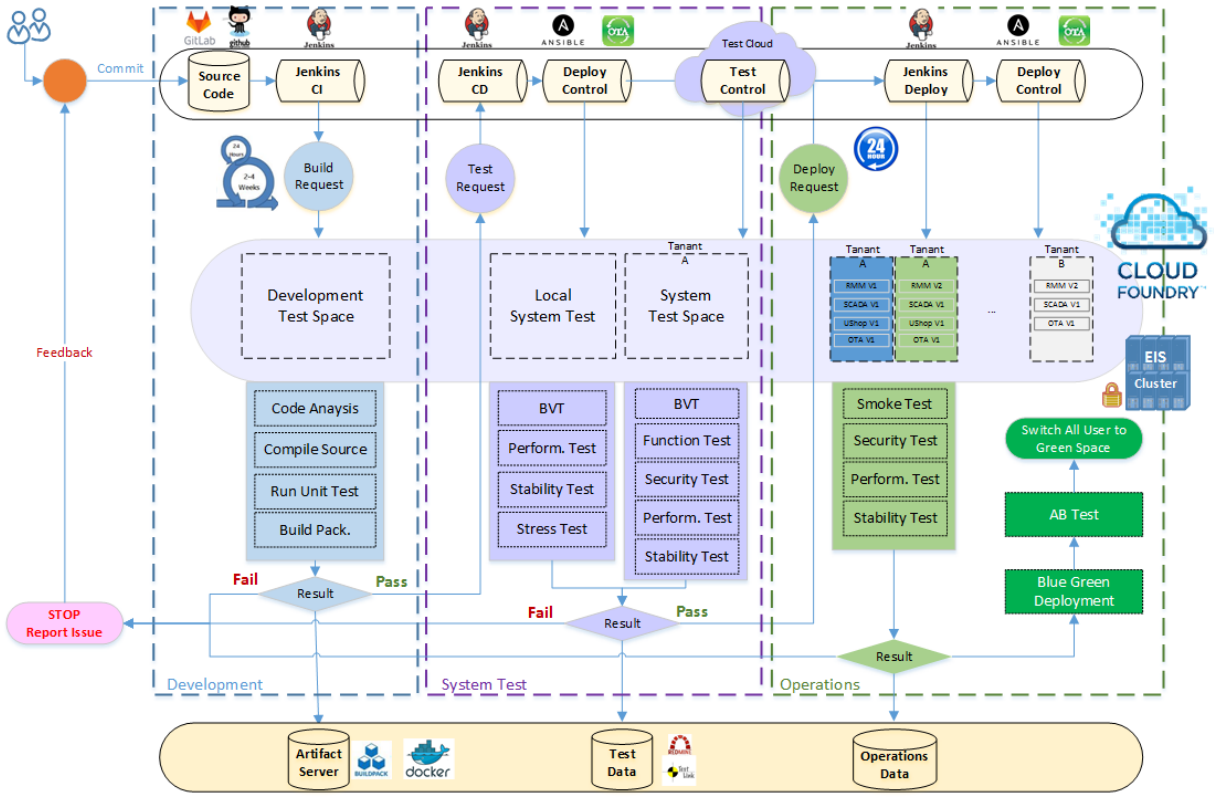


圖 3-1 WISE-PaaS 2.0 平台架構圖

3.1.2 需求分析

軟體交付需要經過建構、測試、佈署等複雜過程，如果主要依賴人工去完成這個流程需要花費很多時間，延誤產品的上線發佈。

為了實現軟體的快速交付，越來越多的企業開始遵循 DevOps 軟體交付理念和方法，DevOps 集文化、實踐和工具於一身，以開發團隊和維運團隊的密切合作為核心，透過工作實踐將交付過程打造成一條包含開發、建構、測試、發佈、佈署、維運等步驟的標準化流程，並用各種工具將其自動化，最終實現產品的快速、高品質交付，並提供 7*24 小時不間斷服務，如圖 3-2 所示，上圖為研華 WISE-PaaS 架構流程與工具，下圖為打造的交付作業標準化流程。



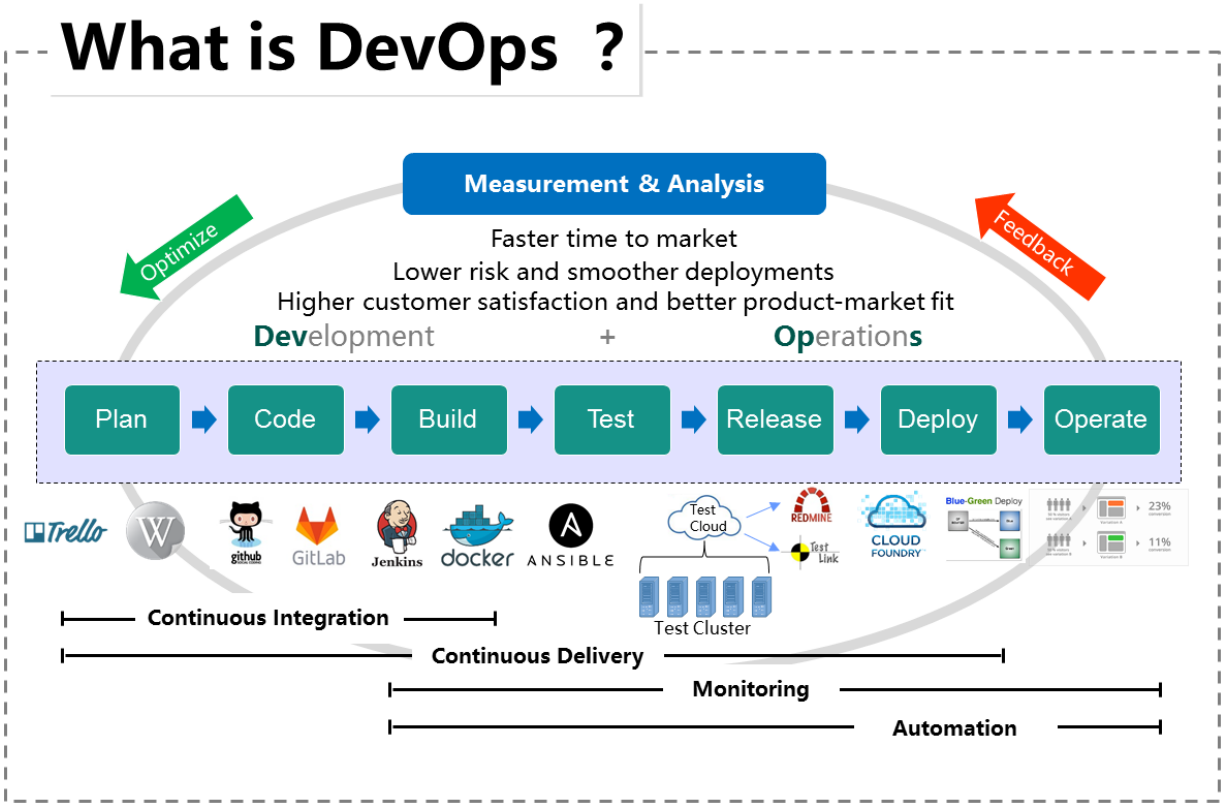


圖 3-2 DevOps 打造交付標準化流程

3.1.3 解決方案

（一）總體技術架構

WISE-PaaS 平台解決方案如圖 3-3 所示。解決方案的最上層提供 CodePipeline 服務，它是正在開發的一款具有持續整合/持續交付能力，並能相容 Jenkins 的 SaaS 化產品。透過使用 CodePipeline，可以使客戶方便的在雲端實現從源碼到應用的持續整合和交付，方便客戶快速的對產品進行功能迭代和推進。

整個解決方案的核心是 Jenkins，Jenkins 提供了軟體發展的持續整合服務，它透過 Master/Agent 架構來實現分散式建構，將不同的任務下發到多台機器（Jenkins Node）執行，提高處理性能。

解決方案的最下層透過 Kubernetes 來管理 Jenkins 的節點，當有建構任務時會自動創建一個 Docker Container 來完成建構任務，當任務結束後 Container 會自動銷毀，資源動態使用動態銷毀，避免資源浪費，無需擔心源碼或建構物外洩。

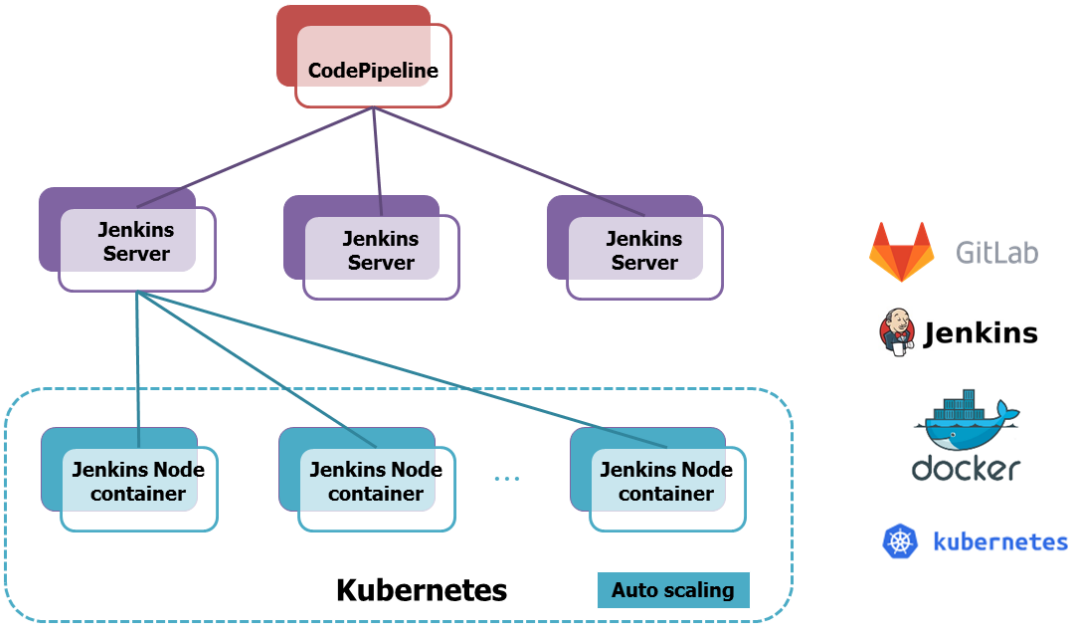


圖 3-3 WISE-PaaS 平台解決方案

（二）具體技術方案

WISE-PaaS 平台技術方案可從兩方面闡述，包括持續交付流程以及 Jenkins Pipeline，如圖 3-4 所示。

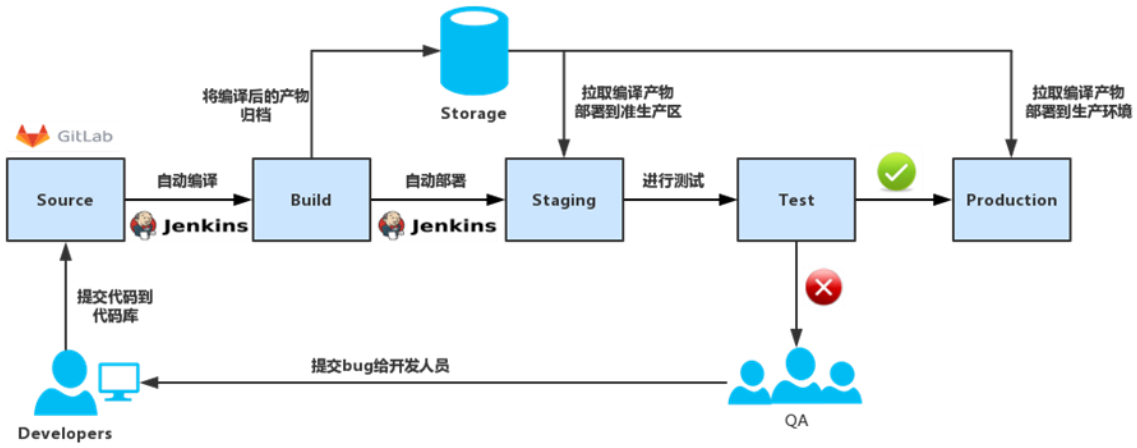


圖 3-4 WISE-PaaS 持續交付流程

1) WISE-PaaS SRP（Solution Ready Package）持續交付流程

- 開發人員提交源碼到源碼倉庫；
- 倉庫打 tag 後透過 Git Webhook 觸發 Jenkins 上面自動編譯的 Pipeline；
- 編譯後將產物儲存到 storage，例如 blob；
- 觸發 Jenkins 上面自動佈署的 Pipeline 從 storage 拉取編譯產物佈署到準生產區；
- QA 在準生產區進行自動化及人工測試，包括功能測試、性能測試、壓力測試和穩定性測試。
- 測試通過後觸發 Jenkins 上面自動佈署的 Pipeline 將編譯產物佈署到生產區。

2) Jenkins Pipeline

Pipeline 是 Jenkins 的一系列外掛程式的組合，透過這些外掛程式可以將持續交付管道化流程在一個 Jenkinsfile 中實現，將複雜的交付流程轉化為 code，即「Pipeline as Code」，並且可以將 Jenkinsfile 放入專案的源碼管理中，像管理其它源碼一樣來管理 pipeline 的 code。jenkins 中 pipeline 腳本是基於 groovy 編寫的，可實現靈活、可擴展的持續發佈（CD）

工作流。

WISE-PaaS 平台 App 的自動建構和自動佈署都是透過編寫 Pipeline 腳本來實現的。在 Jenkins 上面創建建構和佈署 Job，並將建構和佈署的步驟在 Pipeline 腳本中實現，然後配置各種運行參數，便可實現建構和佈署的自動化。Job 被觸發後，Jenkins Server 會將 Pipeline 腳本發送到腳本中指定的 Node 上去執行，最終完成建構和佈署任務，如圖 3-5 所示。

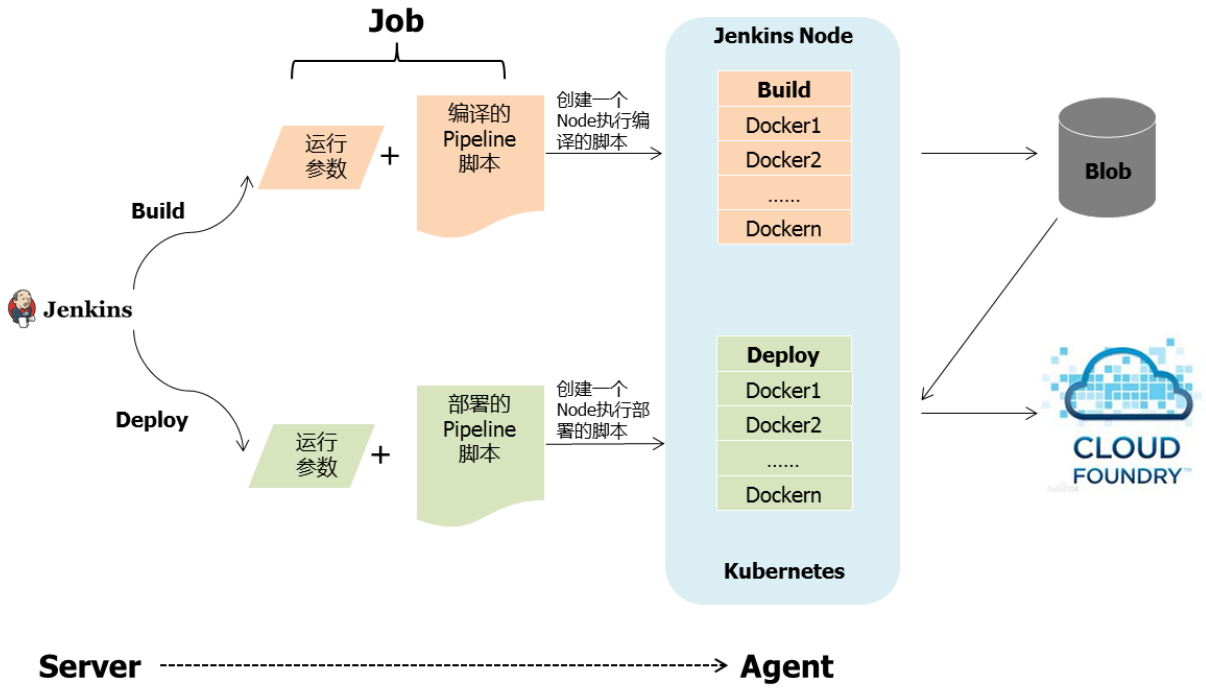


圖 3-5 WISE-PaaS 平台實現自動建構和自動佈署

3) Kubernetes+Jenkins

傳統的 Jenkins Master/Agent 方式可以說明使用者實現分散式建構，提高處理性能，但是在使用時還是會存在很多缺點，例如：

- 當 Master 節點發生故障時，便無法再進行任何建構任務；
- 為了完成不同語言的編譯打包等任務，會創建很多 Jenkins Node，但是這些 Node 的環境又很難複製，導致管理和維護都很困難；

➢ 資源配置不均衡，有些 Node 使用率比較高，會出現 job 排隊的情況，但有些使用率比較低的 Node 卻很多時候又處於空閒，導致資源的浪費。

為了解決以上種種問題，需要尋找一種更可靠更高效的方式來完成 CI/CD 流程，使用 Kubernetes 搭建 Jenkins 集群的架構便解決了這些問題，如圖 3-6 所示。

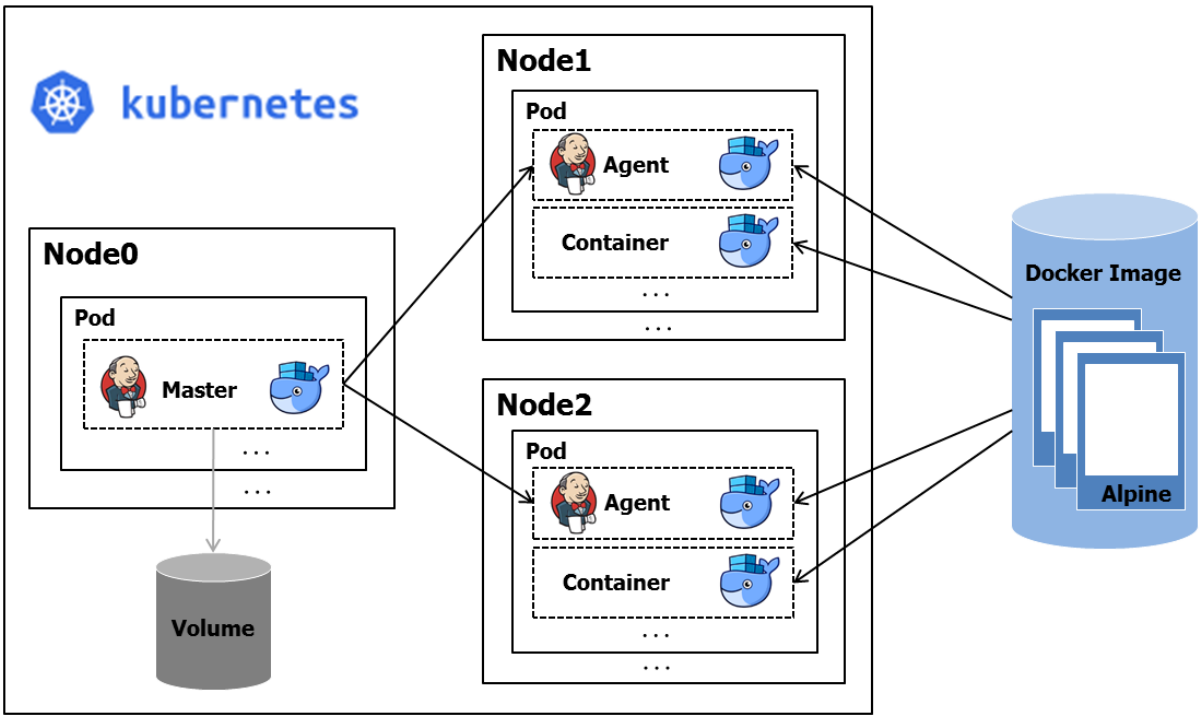


圖 3-6 以 Kubernetes 搭建 Jenkins 集群完成高效 CI/CD 流程

在這種架構中，Jenkins Master 和 Jenkins Agent 以 Docker Container 形式運行在 Kubernetes 集群的 Node 上，創建一個持久化的 Volume 用來儲存 Jenkins 服務的資料，當 Master 出現故障時，可以保證資料不會丟失。創建 Jenkins Agent 使用的 Docker Image 保存在 Docker 儲存服務中（例如 Docker Hub），便於管理和複用。Jenkins Agent 會根據需要拉取 Docker Image 動態創建和銷毀，不會一直佔用資源。

具體實現過程：Jenkins 提供了 Kubernetes 的外掛程式，安裝外掛程式並配置好 Kubernetes 連接資訊後，就可以在 Pipeline 腳本中調用 Kubernetes 的介面自動啟動一個 Pod 和多個 Docker Container，其中一個 Container 作為 Jenkins Agent 註冊到 Master 上面，

建構任務可以指定在其他 Container 上面完成，當所有操作都完成後，Jenkins Agent 會被註銷，並且釋放掉所有 Docker Container 的資源。

4) 藍綠佈署

傳統模式下，如果要更新應用，基本上無可避免有停機時間。在 WISE-PaaS 上，透過藍綠佈署實現不間斷服務的更新。藍綠佈署是軟體佈署模式的一個術語，藍色是現在正在運行的當前版本，綠色是更新的版本，先佈署綠色版本，然後對綠色版本執行煙霧測試。通過測試後，才將應用流量逐步切換到綠色版本，然後監控綠色版本，一旦異常，立刻回滾到藍色版本。整個過程高效迅速，可保證零宕機升級，服務不間斷，如圖 3-7 所示。

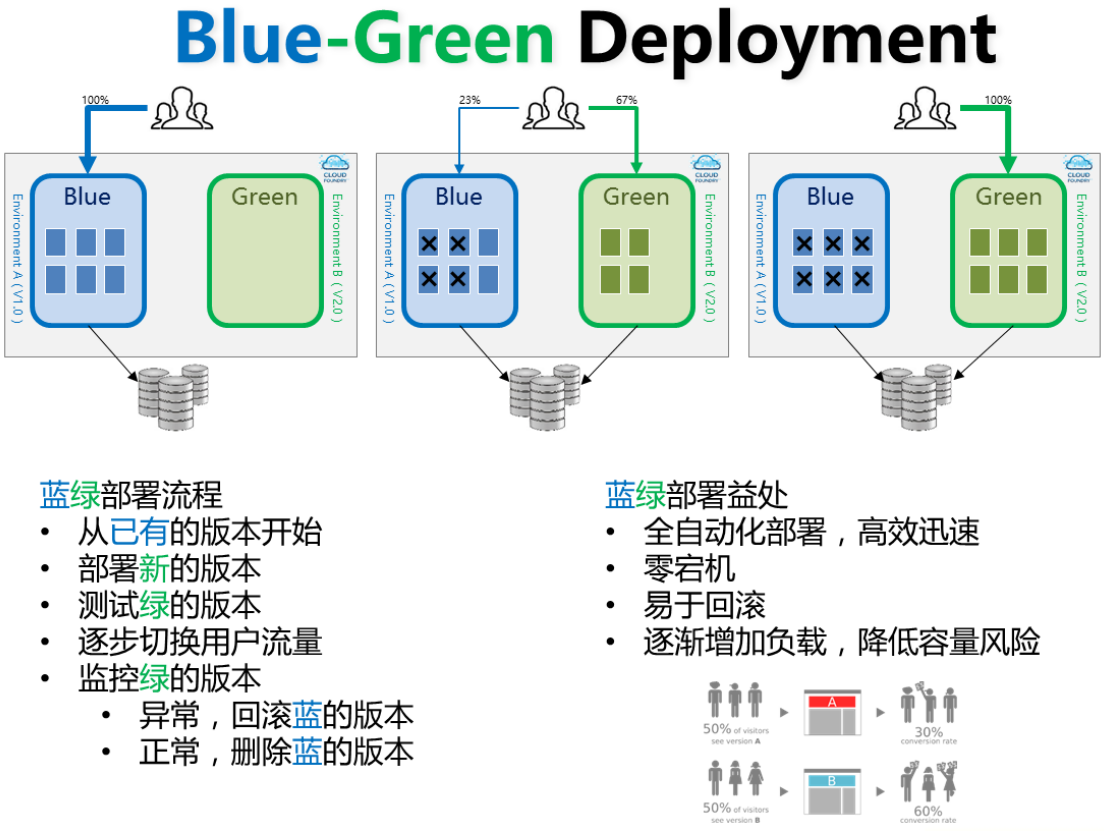


圖 3-7 透過藍綠佈署實現不間斷服務更新

WISE-PaaS App 能夠實現藍綠佈署，主要是因為平台支援 App 透過 Scale 實例個數來分擔流量，比如藍的版本有 6 個實例，目前使用者訪問的所有流量都集中在藍的版本

上。需要升級時，先佈署只有一個實例的綠的版本，測試通過後，再將其 Scale 到 4 個實例，並將使用者使用的 Route Map 到綠的版本上，並且將藍的版本 Scale 為 2 個實例，這個時候，綠的版本正式被用戶所用，並且分擔了 67%的流量，然後再繼續將綠的版本 Scale 到 6 個實例，並且將藍的版本 Stop，最終所有流量成功導入到綠的版本，升級過程中不會出現宕機的狀況。

5) 煙霧測試與 A/B 測試

QA 為每個 APP 都編寫了自動化的 SmokeTest 腳本，每次佈署之後都會自動運行 SmokeTest 腳本，保證 APP 佈署以及基本功能的正確性，如圖 3-8 所示。

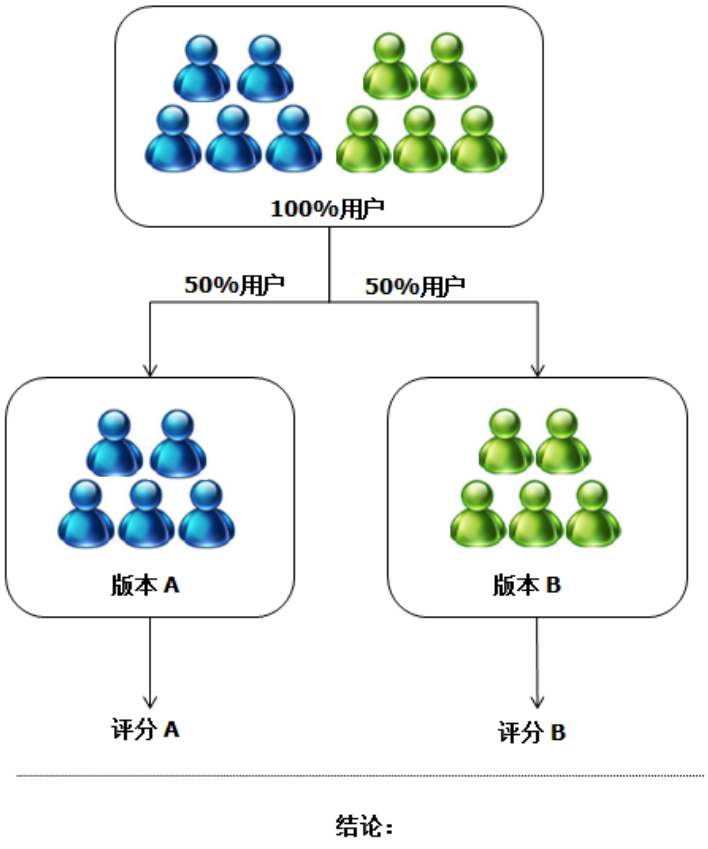


圖 3-8 煙霧測試與 A/B 測試確保功能正確性

WISE-PaaS 平台的 APP 還支援 A/B 測試。在不斷引進持續交付的流程後，產品更新越來越快，不同的方案設計有可能會帶來不同的用戶使用效果，A/B 測試提供了一個有價值的方式來評估新特性對客戶行為的影響。具體來說，就是為同一個目標制定 2 個

方案（例如 2 個頁面），讓一部分使用者使用 A 方案，另一部分用戶使用 B 方案，對 2 種方案進行用戶轉化率、點擊量等指標的統計，以判斷 2 種方案的優劣並進行決策。

WISE-PaaS App 支持 A/B Test，主要是因為平台支持同一個 Route Map 到不同的 App，這樣就可以讓用戶透過同一個 url 訪問到不同版本的 APP。實際測試時，同時佈署 2 個版本的 App，保持相同的實例個數，並且 Map 相同的 url，就可以將使用者流量分別導向 2 個 App，然後透過統計用戶對不同版本的 App 的使用情況，最終確定選擇哪個方案更好。

6) CodePipeline 服務

CodePipeline 是在 Kubernetes +Jenkins 的基礎上開發的一個 SaaS 化的產品，提供視覺化的動作頁面，協助用戶簡單快捷地實現持續整合與持續交付的流程，服務結構設計如圖 3-9 所示，而 CodePipeline 服務提供的視覺化動作頁面如圖 3-10 所示。

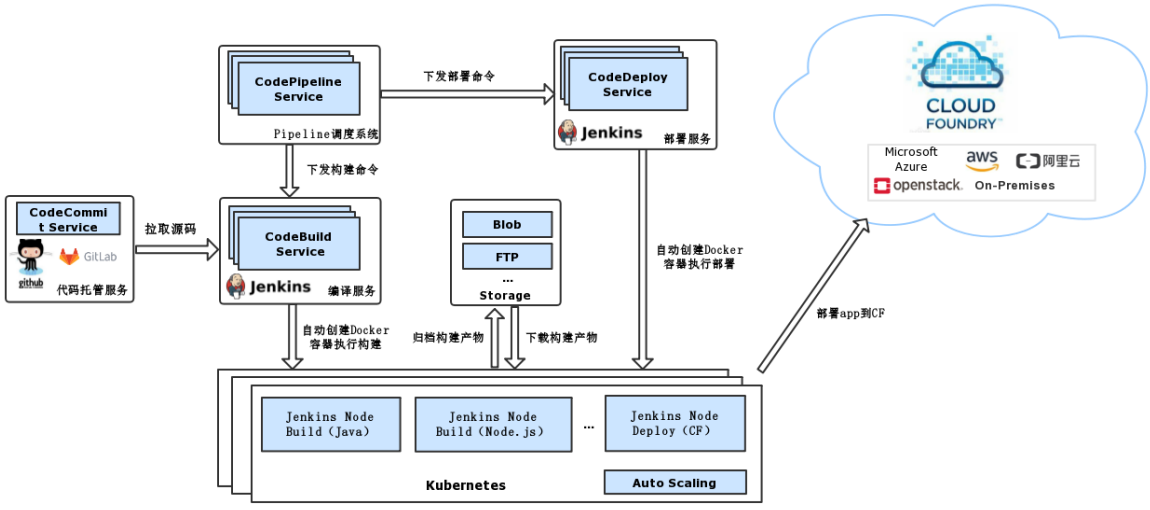


圖 3-9 CodePipeline 服務結構設計

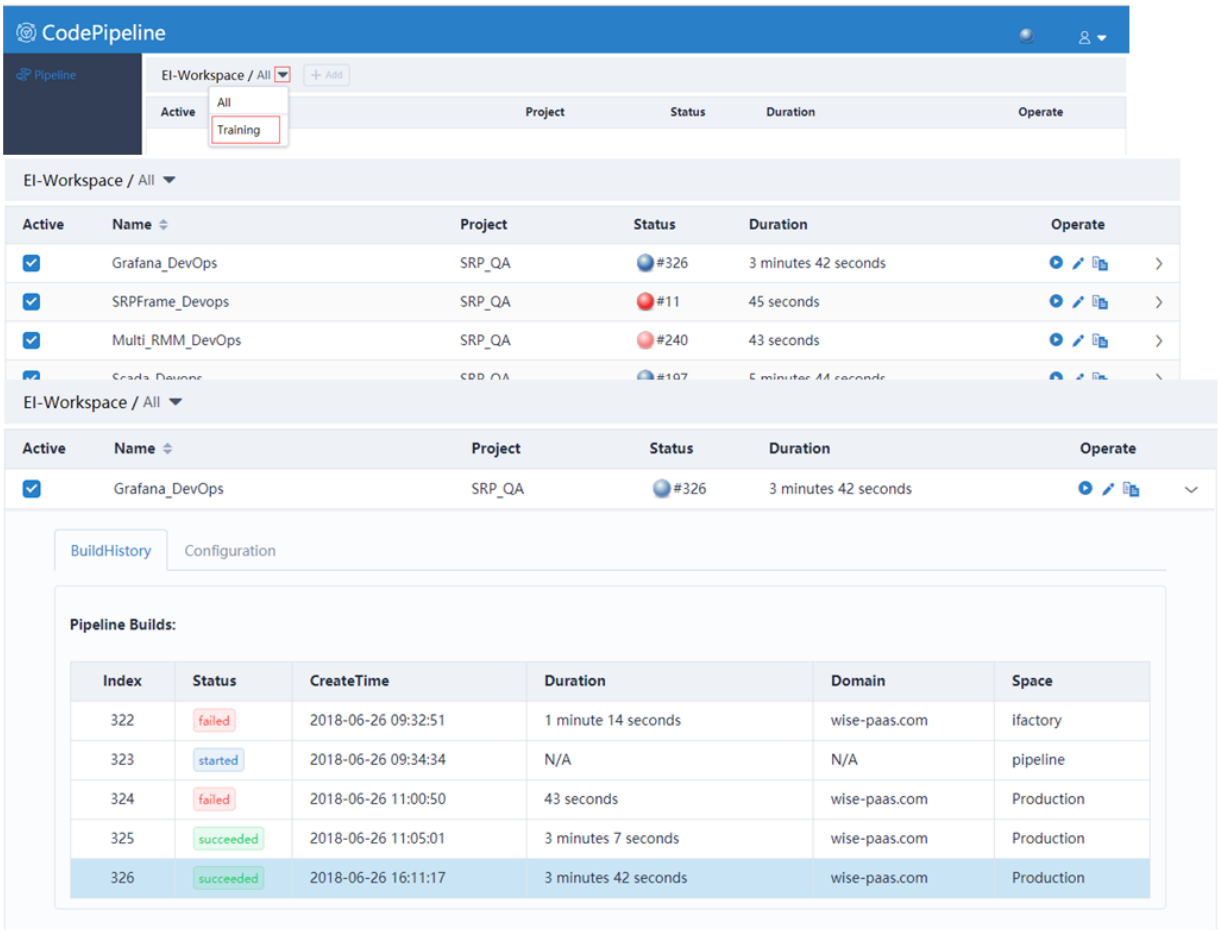


圖 3-10 CodePipeline 服務提供的視覺化動作頁面

（三）解決方案特點

使用 Kubernetes +Jenkins 作為基礎架構，提高了服務可用性，並且實現資源按需使用，避免浪費；上層又封裝了 CodePipeline 服務，讓使用者打開服務即可使用，不需要自己搭建編譯和佈署等複雜的環境，也無需用戶維運：

- 服務高可用：當 Jenkins Master 出現故障時，Kubernetes 會自動創建一個新的 Jenkins Master 容器，並且將持久化的 Volume 分配給新創建的容器，保證資料不丟失，從而達到集群服務高可用性。
- 資源按需使用、動態生成、動態銷毀：Jenkins Agent 只有在 Job 運行時才會被創建，Job 運行結束後會自動銷毀以提高資源使用率，而且還可以防止源碼或者建構物外洩。
- 服務打開即用，無需額外的配置：打開 CodePipeline 服務連結，即可在上面透過

配置完成 CI/CD 的過程，不需要自己搭建編譯和佈署等複雜的環境。

- 使用者無需關心整個交付系統的維運，由平台統一管理。

3.1.4 總結

（一）經濟/社會效益

目前研華內部人員已經開始使用 CodePipeline 服務來完成部分 SRP 的持續交付過程，加速了軟體產品的交付流程，降低佈署過程出問題的風險，提高了客戶滿意度和產品的市場競爭力。

- 自動化軟體發佈流程
- 提高開發人員的工作效率
- 更快發現並解決缺陷
- 更快交付更新

（二）用戶評價回饋

目前 CodePipeline 還有很多需要完善的地方，使用者在使用過程中也發現了一些 bug，目前正在修正中。系統本身也有很多功能需要完善，比如需要設計更完善的使用者操作許可權，整合 WISE-PaaS 雲平台的 SSO 系統，在使用方面也希望能增加一些更簡單的視覺化配置，讓使用者不用寫源碼就能快速完成編譯和佈署的工作。

（三）小結

在快速發展的雲時代，軟體產品層出不窮，為搶奪先機，第一時間將產品上市，快速發佈和保證品質成為產品取勝至關重要的因素，而 DevOps 就是順應時代的潮流產生的一種軟體交付理念和方法，遵循 DevOps 的交付理念，可幫助開發人員第一時間發現軟體中的缺陷，保證產品快速和高品質上線，並提供不間斷的服務，提高客戶的滿意度和產品的市場競爭力。所以 DevOps 已經成為軟體驅動型企業在雲時代取得成功的關鍵。

3.2 Gogolook—全球千萬使用者等級 App 的 DevOps 架構

3.2.1 案例簡介

（一）案例背景

Gogolook（走著瞧股份有限公司）成立於 2012 春天，成員來自臺灣、香港、韓國、巴西等地的軟體創新人才，希望透過快速且不間斷地服務創新以及使用者為中心的設計理念，建立全球行動裝置上最值得信賴的人際溝通網路。

Whoscall 是目前最具代表性的產品，擁有全球超過十億筆電話資料庫資訊，提供 Android 及 iOS 行動裝置使用者最即時的來電辨識服務。Whoscall 於 2012 年獲當時 Google 執行長 Eric Schmidt 公開表揚，並獲數字時代、華人行動應用大賞、Google 年度創新獎等獎項肯定，曾被全球知名科技媒體 TechCrunch、TechinAsia 專題報導，也與臺灣警政機構、韓國金融監督局（FSS）等單位簽訂合作備忘錄。

Whoscall 全球用戶超過六千五百萬，主力市場為臺灣、巴西、香港、日本、韓國等地，可警示惡意與推銷來電，資料庫中也提供豐富的商家營業信息，並已成功的協助數千萬的商家透過來電辨識功能取得行動用戶的信賴。目前亦致力於運用資料庫系統發展人工智慧，藉由長期使用者接收詐騙電話的資料分析模擬詐騙集團的行為，進而預防犯罪。

隨著使用者規模擴大，服務內容增加，對於基礎設施穩定性、研發敏捷性、產品安全性的考驗亦不斷上升。為了適應下一階段的服務挑戰，公司開始進行一連串的 DevOps 大改造。

（二）案例特點

Whoscall 服務的 DevOps 改造案例是依據 DevOps 指導原則逐步進行，流程與工具改造並重，此經驗對其他軟體公司應該很容易借鑒實施。以《鳳凰專案》（The Phoenix

Project）「三步工作法」觀點闡述案例特點如下：

- 流動原則：以自動化工具為輔，全面貫通提交源碼、測試、組態配置、佈署等流程。在安全網的基礎上，放心提交源碼進行實驗，加速研發與應變腳步。
- 回饋原則：透過許多層次，透明化監測佈署的任何服務的實際運作狀態，以得到即時且精密的回饋，進而產生對於服務品質及商業目標的洞見。
- 持續學習與實驗原則：系統化改善端到端服務流程，對於服務流程及規格，有宏觀及微觀的掌握。對研發的工程品質，能夠起規範化的積極作用，也有助於加速團隊人才訓練，培育更多即時戰力。

3.2.2 需求分析

Whoscall 核心服務涵蓋行動裝置與雲端系統，為了進行 DevOps 改造，需要技術面與流程面密切配合，故先界定整個改造計畫的總體目標如下：

- Infrastructure as Code：在可能的情況下，應盡力追求配置透明化，利於優化配置、查找問題、經驗傳承，才能進一步演練 chaos engineering 等高級措施；
- 快速佈署：從提交源碼，經由一系列測試、組態配置、佈署程式，需要迅速且穩定的佈署流水線，才能有效支持快速實驗的需求；
- 兼顧安全與彈性：每次佈署都需要通過一系列的編譯期與執行期的檢測基準，確保合規要求；同時也兼顧彈性，能在執行期進行迅速的動態調整；
- 服務監控：針對內部系統及外部依賴系統，需多層次監控以便快速反應、優化服務；
- 端到端服務流程系統化：過去在研發驅動力主導之下，累積了許多技術債，現在已逐漸反噬到研發的步調。因此，從服務頭尾兩端開始系統化檢討整頓服務流程環節，以利於查找問題、累積組織流程資產；

- 透過上述基礎來敏捷地拓展更多服務，發掘更多商業價值。在改造 DevOps 的同時，也持續進行技術選型評估。由於 Gogolook 是小而精的公司，為了集中研發能量，降低導入及學習遷移阻力，降低招聘新人時教育訓練的成本，其技術選型原則有下：

- 避免 lock in，這點非常重要，尤其在技術快速發展、版圖不斷挪移的軟體產業，不被特定工具或廠商 lock in，維持技術選型的自主性，才能敏捷前進；

- 以前瞻眼光選擇活躍的開源軟體，並積極參與線上及線下的社群活動，交流實務經驗；

- 選擇泛用性高、擴充性高的開源軟體，避免技術棧過度碎片化，增加維運基礎設施的困難度；

- 審慎選擇 IaaS/PaaS/SaaS 的服務。善用這些服務，可降低維運基礎設施的精力；但其推陳出新速度，可能沒有對應的開源軟體來得快，這部份需要再加以權衡；

- 若沒有適當的現有組件，則自建，但仍對替代方案保持開放態度。

3.2.3 解決方案

（一）總體技術架構

根據前述技術選型原則，與 DevOps 相關的技術選型如下：

- CI/CD 流水線：Jenkins；
- 行動端平台：Android 及 iOS；
- 行動端 A/B 測試機制：早期自建，近期逐漸改用 Firebase；
- 行動端效能分析：Android vitals、Crashlytics；
- 後端公有雲：Amazon Web Services（AWS）；
- 後端組態管理：Ansible 及 Docker/Kubernetes 生態系；
- 後端效能及日誌分析：New Relic、Sentry、EFK、Prometheus、Grafana；

- 流程與 API 規格介面：Swagger、Cucumber。

（二）具體技術方案

1) Infrastructure as Code（IaC）

- Infrastructure as Code（IaC）是現代從業人員必備的紀律，在可能的情況下，應盡力追求配置透明化，利於優化配置、查找問題、經驗傳承，也才能夠進一步演練 chaos engineering 等高級措施。另外，Ansible 是個泛用性極高的組態管理及遠端執行工具，故選擇以 Ansible 作為 IaC 機制，理由如下：

- 輕量化：IaC 不應有過多的額外開銷（譬如：額外的認證機制、額外的背景程式）。而 Ansible 不需要額外常駐的背景監聽程式，也不需要額外的帳號認證機制，完全沿用 Linux 既有的機制，可降低管理的複雜度；

- 隨插即用：對機器及環境僅有最少的假設。一般來說，只要 Linux 主機有 ssh 及 Python 元件，就能被 Ansible 管理，因此，甚至連 Alpine 這類極輕量的 Linux 系統都能納入列管範圍；

- 易學：架構單純，YAML 配置語法比傳統指令碼語言更簡單、更易複用；

- 易擴充：Ansible 架構單純，可以用正規的 module 形式擴充功能，亦可用陽春的腳本 + stdout/stderr + json 手法擴充，彈性非常大；

- 社群活躍：這個理由尤其在被 RedHat 併購之後，其永續發展更是無虞。

- 後端系統是以 Linux 為主，多半佈署於 AWS，因此以羽量級的 Ansible 為核心的組態配置工具，再搭配其他技術特有的組態配置設施：

- Linux 主機：自行編寫並維護 Ansible roles；

- AWS 系列：用 Ansible 驅動 CloudFormation 等配置機制。

- 新一代的後端技術，多半都直接內建 IaC 機制，如 Docker 生態系，甚至各種

serverless 系列；採用這些技術時，直接套用各自的 IaC 機制即可：

- Docker：用 Dockerfile 及 docker-compose.yml 配置；
- Kubernetes：一堆 YAML 檔案配置。

目前已針對這些組態配置有小規模進行 chaos engineering 的試驗，預計未來會更進一步利用 Netflix 釋出的 Chaos Monkey 系列進行大規模的演練，以確定整個 IaC 配置的完整性。

2) 快速佈署

從提交源碼開始，經由一系列測試、組態配置、佈署程式，都需要迅速且穩定的佈署流水線（deployment pipeline），才能有效支持快速實驗的需求，因此佈署流水線以 Jenkins 為主軸，貫串相關的前端與後端流程：

- Android 源碼→Git→Jenkins→Gradle 腳本；
- iOS 源碼→Git→Jenkins→Xcode 腳本；
- 後端源碼→Git→Jenkins→Ansible 腳本。

重要的提交狀態也都會送到 Slack 及對應的監控儀錶板，讓進度透明化。

在關於行動端軟體的部分，利用 Android 新的 Dynamic Delivery 機制，將行動裝置軟體打包成 App Bundle 格式，各種行動裝置使用者能夠下載最適尺寸的 APK 套裝軟體，也簡化組態配置流程。

後端系統主要採微服務（microservices）架構，將後端服務劃分成更小的獨立佈署單位，每次提交源碼時，只要不涉及重大的架構變動，正常情況下都可不停機持續佈署。

利用前面提到的 Ansible 進行後端服務的打包及佈署步驟。主因 Ansible 是一泛用性極高的組態管理及遠端執行工具，可用一致性的 YAML 語法調用多種不同的技術，減少需要掌握的技術棧：

- 可針對編譯式或直譯式的源碼進行打包、測試、佈署等步驟;
- 可針對本機端、裸機、雲端主機進行組態設定及持續佈署;
- 可針對傳統軟體、Docker 化軟體、serverless 軟體進行持續佈署。

在 AWS 上利用 Ansible 或 Docker 開啟新主機或容器相當簡單，善用這種優勢將後端架構設計成 immutable infrastructure，進一步降低持續佈署的複雜度，也提升後端系統的穩定性。

3) 兼顧安全與彈性

由於與多個公部門有合作關係，為確保品質，每次佈署都會通過一系列的編譯期與執行期的檢測基準;其中自建的檢驗工具委由符合 ISO/IEC 17025 規定的公正第三方定期進行資安檢測。除了上述測試與合規程序外，也須兼顧彈性，確保能在執行期進行迅速的動態調整，系統流程才算完整。

關於行動端軟體的部分則藉由 Google Play Console 的分段回滾（staged rollout）機制、Apple App Store 的分段發佈（phased release）機制，在快速佈署之餘，亦能兼顧風險控制。另外也利用 Firebase Remote Config 及自建的 A/B testing 技術，讓不具軟體研發能力的實驗設計者，也能在不重新佈署行動端軟體的前提下，動態進行多變數的實驗，且同時適用於 Android 及 iOS 兩大平台。

關於後端系統，由於 AWS 提供許多可程式化的動態機制，所以再利 Ansible 或 Docker 進行藍綠佈署（blue/green deployment）或金絲雀佈署（canary deployment）就相對簡單。隨著 AWS 正式支持 Kubernetes（名為 EKS），更可直接在儀錶板上進行這方面的操控，降低維運複雜度，提升透明度。

4) 服務監控

內部系統及外部依賴系統需要多層次的監控，以便快速反應、優化服務。故針對行

動端軟體的部分，除了自建的日誌與性能監控機制外，也利用 Crashlytics 服務來監控軟體閃退事件。最近更利用 Android 新的 Vitals 機制，監控行動軟體的種種品質數據：啟動時間、閃退、顯示速度、網路熱點掃描、背景執行行為等等，作為優化產品的依據。後端系統則動用較多層次的監控機制。

- 硬體基礎性能：大部分以 AWS 的 CloudWatch 服務來監控雲端主機群的硬體基礎性能資料。如需更細緻或更廣泛的性能資料，則透過 Prometheus 生態系提供的一系列 exporter；
- 日誌：一般情況下以 AWS 的 CloudWatch 服務來搜集與彙整 layer 7 的日誌，但如需更即時、更細緻、更有彈性的日誌處理，則透過 EFK（Elasticsearch + Fluentd + Kibana）技術棧來達成；
- 軟體棧性能：以 New Relic 服務掌握軟體在作業系統層次、執行引擎層次、用戶端層次等的分層效能資訊；
- 執行期錯誤：採 Sentry 服務全盤掌握應用程式動態拋出的執行期錯誤；
- 外部健康狀況：採 Pingdom 服務從全球使用者視角理解己方或第三方廠商 API 服務的回應時間與健康狀況；
- 上述的後端系統監控機制多針對特定環節，缺少綜合性、全域性、多層次的視角，因此另外再透過 Prometheus + Grafana 系統，取得以下更多效果：
- 自訂儀錶板：透過 Grafana 針對各服務、各流程所關心的視角，設計儀錶板及統計資訊呈現方式；
- 自訂時間序列運算式：透過 Prometheus 的 PromQL 設計合適的時間序列運算式，精準呈現出關鍵的服務運作指標及趨勢曲線。

5) 端到端服務流程系統化

因過去研發驅動累積的技術債反噬到研發步調，故從服務頭尾兩端系統化地整頓服務流程環節提升整體品質，利於查找問題，累積組織流程資產，逐步進行改革如下：

- 介面思考

多人協作團隊甚至與第三方合作互聯，都非常需要制定良好的技術介面，才能促成順暢合作。為了加速前端後端的並行開發，在研發團隊小規模導入 Swagger 及 RAML，隨著 OpenAPI Specification (OAS) 大勢底定，相關工具逐漸豐富，開始更大規模實施，也推廣到品質監管等職能身上，讓感興趣的人可透過技術介面，理解並實驗相關的後端服務，進而降低教育訓練的成本。這是很重要的基礎，它可以改善後端系統的介面透明度，進一步優化服務流程。

- 流程思考

隨著使用者規模擴大，服務內容增加，流程也漸趨紊亂，因此，開始以系統化角度，重整過去大大小小的服務流程與資料流程，採取以下措施提升服務流程的透明度：

盤點：首先組成特別小組，針對既存的所有服務，進行第一次廣泛的總盤點，先不求深，但力求廣泛無遺漏。

規格表述：由跨職能小組針對高優先度的專案，以類似 BPMN 業務流程模型和標記法（Business Process Management Notation）的形式，深度分析服務的業務與資料邏輯，除做精準表述外，也藉此機會找出可優化流程。

數據埋點：有了透明的流程表述之後，就能進一步在關鍵處埋點，以利於追蹤每一個內部環節的即時狀況。

儀錶板：有以上基礎便可針對各服務、各流程所關心的視角，設計各自的儀錶板，呈現各自的進度狀況，甚至預測未來的走勢，及早預警。

- 規格思考

可執行的軟體規格一直是軟體圈的夢想。以 DevOps 角度而言，MDA（model-driven architecture）之類的軟體塑模作法不見得是最優先的目標，但 SBE（Specification By Example）則有機會以最小的投入產生最大的串連效益：

代表性的測試案例：即使團隊不想轉變成 TDD（Test-Driven Development）或 BDD（Behavior-Driven Development）的研發風格，SBE 仍然可以從端對端測試案例的角度切入，不大幅改變工作習慣，即可從多方面對整個 DevOps 有所貢獻 。

測試自動化：搭配適當的工具可將 SBE 搭上自動測試、回歸測試的流程。

規格：Cucumber 倡議的 GWT（Given-When-Then）語法是一種結構化的規格表述形式，可應用在端到端的黑箱層次，也可應用在 API 服務或高階模組的灰箱、白箱層次。

將上述特性結合起來，SBE 規格就能成為活檔（living document），不僅捕捉業務邏輯，也透過測試自動化機制確保 SBE 檔如實反映最新現況。上述三項措施若實施到位整合就會形成多層次的系統化 DevOps 視角。以端到端的業務服務流程視角來說，BPMN 這類形式可展現出巨觀的流程透明度，SBE 展現出微觀的透明度，且兩者都以非技術人員能夠理解的方式呈現，有助於形成團隊內的共同語言。

以內部技術介面視角而言，Swagger 提供精準的技術語彙，讓稍具技術介面觀念的人，能夠自行掌握介面資訊，並自行做一些操作實驗。

以內部品質管制的視角來說，SBE 搭配測試自動化機制，能降低端到端黑箱或模組白箱的反覆測試人力，進而釋出精力進行更具開創性的探索性測試等品質管制方法。

以內部知識管理的視角來說，上述三項措施，都在累積組織流程資產，減少盲目試誤與爭論的精力。

3.2.4 總結

1) 維運效益

本案例以「三步工作法」相關措施增進的維運效益大致如下：

- 流動原則：Infrastructure as Code 及兼顧安全與彈性的快速佈署程式，積極面在提供催化實驗與緊急應變的溫床；而消極面在提供資訊系統的安全網，避免常見的人為錯誤；
- 回饋原則：逐漸改善服務監控機制就能透過多層次，透明化監測佈署的任何服務的實際運作狀態，作為進一步優化系統效能的根據；
- 持續學習與實驗原則：端到端服務流程的透明度愈高，愈能從流程面發掘更多優化的可能，增進整體服務的效率及可靠度。

2) 研發效益

本案例措施增進的研發效益如下：

- 流動原則：當逐漸改善 Infrastructure as Code 機制之後，研發者免於瑣碎的組態設定問題，得以集中精力在研發任務上，當實施兼顧安全與彈性的快速佈署程式後，研發者可在安全網的基礎上，放心提交源碼進行實驗，加速研發與應變腳步；
- 回饋原則：當逐漸改善服務監控機制，就能透過多層次透明化監測佈署的任何服務的實際運作狀態，研發者可得到即時且精密的回饋，進而產生對於商業目標的洞見；
- 持續學習與實驗原則：當逐漸系統化改善端到端服務流程，研發者對於服務流程及規格可有巨觀及微觀的掌握，對於研發工程品質，能夠起規範化的積極作用，也有助於加速團隊內人才的訓練，培育出更多的即時戰力。

3) 小結

隨著使用者規模擴大，服務內容增加，對於基礎設施穩定性、研發敏捷性、產品安全性的考驗亦不斷上升，內部進行了一連串 DevOps 改造，以常見的 CALMS（Culture, Automation, Lean, Measurement, Sharing）模型來說，改造不僅在技術面運用許多自動化

工具，亦著重流程面的改良及文化面的孵育，務求迅速且安全的流動、即時且多層次的回饋、充分支援持續學習與實驗。

本案例從 Whoscall 的服務背景開始探討，分析 DevOps 改造需求，再進一步逐一探討技術選型思路，並提供具體的建議，這樣改造沒有終止，各種議題也仍持續嘗試反思調整，期以此文共同探索更好的 DevOps 之道。

3.3 騰訊—社交網路業務持續維運方案與實踐

3.3.1 案例簡介

（一）案例背景

騰訊社交網路業務持續維運的技術方案，是基於騰訊公司社交網路事業群的 DevOps 技術實踐的維運側技術的實踐案例。實踐經驗均來自於騰訊為了提升維運品質、效率的總結，相關的技術架構與解決方案適用於大規模的維運管理，屬於 DevOps 持續維運與持續回饋的範疇。

1) 騰訊社交網路業務簡介

騰訊公司一直以社交產品著稱，其產品線也是以社交業務為核心，往外拓展與延伸到線民的日常生活中。在眾多的騰訊社交網路產品中，以 QQ、QQ 空間等為代表的社交網路業務，經過十幾年的發展，已經是日活躍用戶數超過 2 億的大規模業務，這些業務既擁有傳統的基礎設施和應用軟體架構，又有新興的雲技術架構，對於 DevOps 的持續維運提出了巨大的挑戰。騰訊公司組織架構如圖 3-11 所示。

圖 3-11 騰訊公司組織架構

2) 持續維運平台簡介

騰訊織雲一體化維運平台，是騰訊公司 DevOps 流水線中，負責持續維運和持續回饋環節的維運解決方案，具體功能覆蓋自動化維運、立體化監控與智慧化維運三個方向，本案例會重點介紹組成這三部分的核心技術與設計思路。騰訊 DevOps 流水線解決方案如圖 3-12 所示。

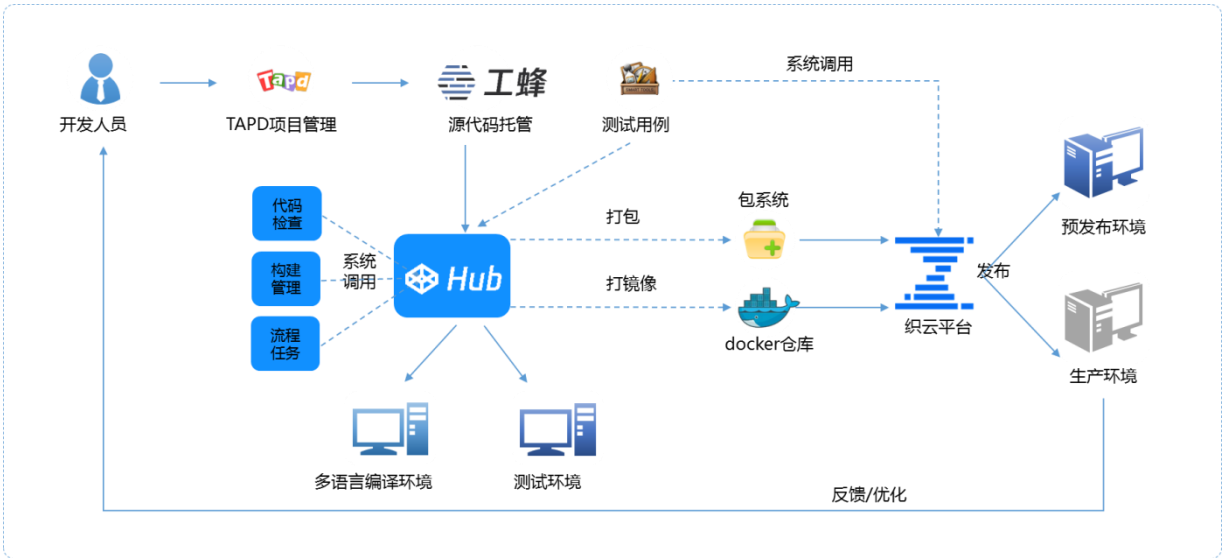


圖 3-12 騰訊 DevOps 流水線解決方案

（二）案例特點

企業要實現持續維運的能力，不僅需要解決來源於產品、開發、測試團隊發起的需求，還要解決來源於維運團隊自己發起的需求，這需要有全域的規劃，從標準化規範、配置管理、工具建設和維運自動化場景等，體系化的建構維運平台的功能。同時，企業建設持續回饋的 DevOps 能力，需要有體系化的監控能力支撐，實現統一監控與告警的維運平台功能。

1) 基礎設施環境管理

騰訊社交網路業務分佈全球，基礎設施由實體設備、虛擬機器、容器化組成，對於持續維運技術而言，需要考慮多 IDC、多雲管理的難題。

2) 複雜維運物件的管理

有效的管理好維運物件，是持續維運和持續回饋過程中必然要應對的。在對社交網路業務的維運過程中，包括網路設備、主機、應用軟體、設定檔、業務、進程等等，都需要有品質與效率的對應管理方案。

3) 智慧化維運的技術架構

騰訊社交網路業務擁有超 20 萬台的基礎設備規模，在大規模維運的場景中，無論是建構自動化維運技術或立體化監控技術，都難以用常規的技術架構來實現。智慧化維運技術的導入，可以更好的幫助維運團隊完成工作。

3.3.2 需求分析

（一）多基礎設施管理

多基礎設施的管理，從維運技術本質分析，是對於基礎設施的管控與監控的需求。維運的基礎能力如何能夠觸達不同的基礎設施環境，如何能夠傳輸命令控制不同基礎設施環境中的維運物件，如何能夠從不同的基礎設施環境中收集監控資料，是實現多雲管理的技術需求。

（二）多維運物件管理

對維運物件的管理，直接會決定維運效率與品質建設的成熟度與深度。全域枚舉出維運工作中可能涉及的維運物件，結合維運工作的需求和場景，建構出管控和監控該維運物件的工具體系，是持續維運所期望達到的目標。

（三）維運效率建設的需求

DevOps 中對維運效率的建設需求，實質上是對標準化維運體系的建設需求。是持續交付原則中，用最低的成本來創建可靠可重複過程，確保維運品質與效率並重的需求。

（四）維運立體化監控的需求

傳統維運的監控思路，可以認為是「知其然，所以然」的技術實現過程，也常常被詬病為「頭疼治頭，腳疼治腳」的方案。在大規模的維運監控需求分析中，立體化的覆蓋業務各個品質點是監控技術的關鍵。並且，在保證監控能力「全、快、準」的同時，要有方案能夠從繁雜的監控資料中提煉或收斂出最核心最有價值的關鍵指標，是持續回饋的核心需求。

（五）一體化維運平台管理需求

靠傳統煙囪式的維運工具和監控資料，將無法實現維運技術的智慧化升級，為此一體化維運平台的需求應運而生，其目的是透過實現維運數據的關聯分析，快速實現故障定位、自癒與止損，走向無人職守的全自動維運階段。

3.3.3 解決方案

本文技術方案來自於騰訊社交網路營運部，是騰訊維運團隊解決自身大規模持續維運與持續回饋的技術實踐。

（一）平台技術架構

織雲是騰訊一體化維運平台，整體功能分成幾大模組：基礎設施、CMDB(Configuration Management Database)、基礎維運、公共元件服務、持續佈署、架構元件、監控體系、流程與服務、AIOPS、通知&日誌中心，在這些功能模組之上，是統一的維運門戶，集中化實現維運的操作、場景化工具、統一視覺化的功能，滿足持續維運與持續回饋的中心化或一體化維運管理的需求。騰訊織雲一體化維運平台如圖 3-13 所示。

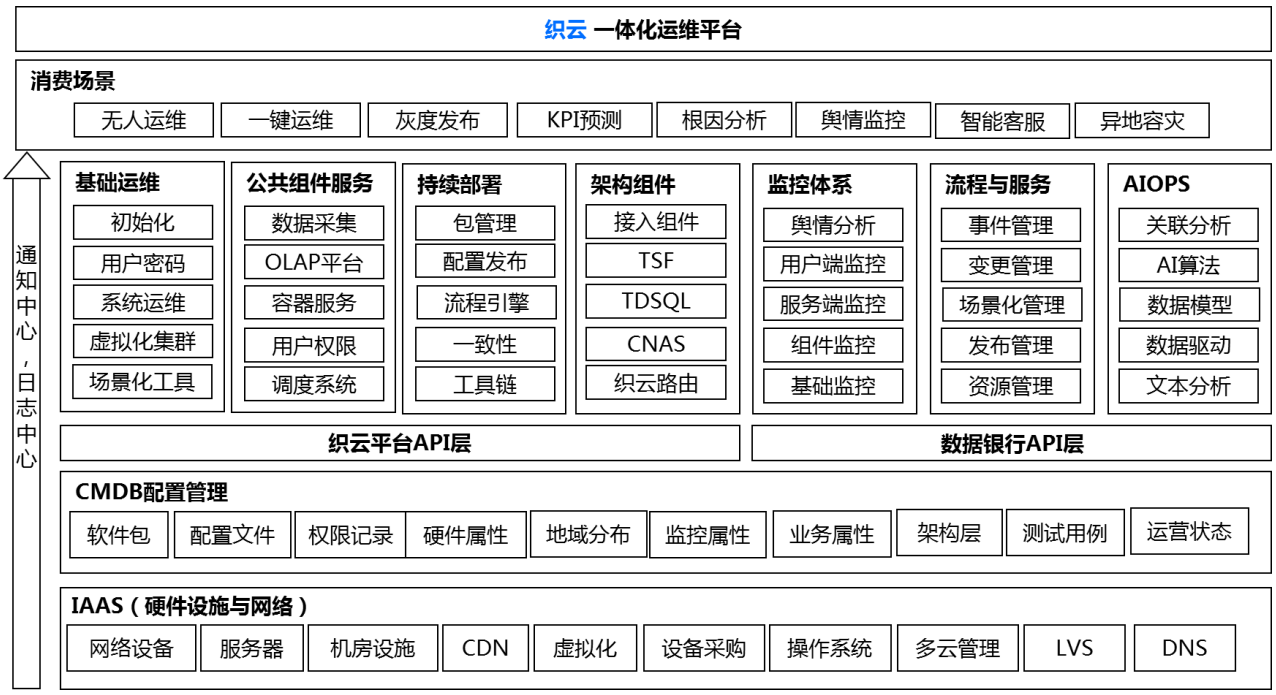


圖 3-13 騰訊織雲一體化維運平台

（二）具體技術方案

為解決持續維運過程中會遇到的問題，以下是核心的幾個一體化維運平台的技術架構設計與技術實踐。

4) CMDB 配置中心

在維運自動化平台的設計理念中，騰訊維運一直提倡「減少維運物件」，並將維運物件進行抽象化、模型化、配置化的錄入 CMDB 中管理，進而讓維運工具有途徑消費 CMDB 中的資料，讓維運自動化流程能夠透過介面維護 CMDB 中，各個維運物件的屬性與狀態，這是建構自動化維運體系的配置基礎。泛 CMDB 設計如圖 3-14 所示。

圖 3-14 泛 CMDB 設計

在織雲面向業務架構的 CMDB 實踐中，將 CMDB 的配置資料主要分為幾類（更多配置需求可擴展）：

- 業務對象：業務樹、架構層等；
- 硬體物件：主機、網路設備等；
- 應用物件：套裝軟體、設定檔、腳本等；
- 自訂對象：變更資訊、密碼庫等；
- 每種維運物件都包含其各自的屬性和狀態配置，如：
- 業務樹：業務層級關係、負責人、重要級別等；
- 主機：IP 位址、主機名稱、作業系統、上聯交換機埠等。
- 套裝軟體：版本、佈署實例、流程、埠、清理策略等
- 變更信息：時間、IP、操作內容、操作人員等

維運物件與維運物件之間需要建立規則或關係，CMDB 中維運物件間的關係描述如圖 3-15 所示。

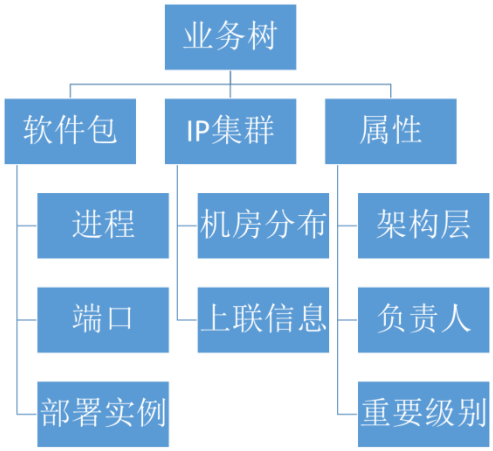


圖 3-15 CMDB 中維運物件間的關係描述

透過上述 CMDB 的設計，使持續維運過程中的工具和介面都能夠保證資料來源的統一集中，基於 CMDB 配置資料的組合可以實現豐富場景的維運效率工具。其中多個雲平台或多個資料中心的管理方案，基於 CMDB 的技術架構如下，透過 CMDB 配置項區分不同雲使用的工具與介面模式，自動的選擇最恰當的方案。多雲管理的技術架構如圖 3-16 所示。

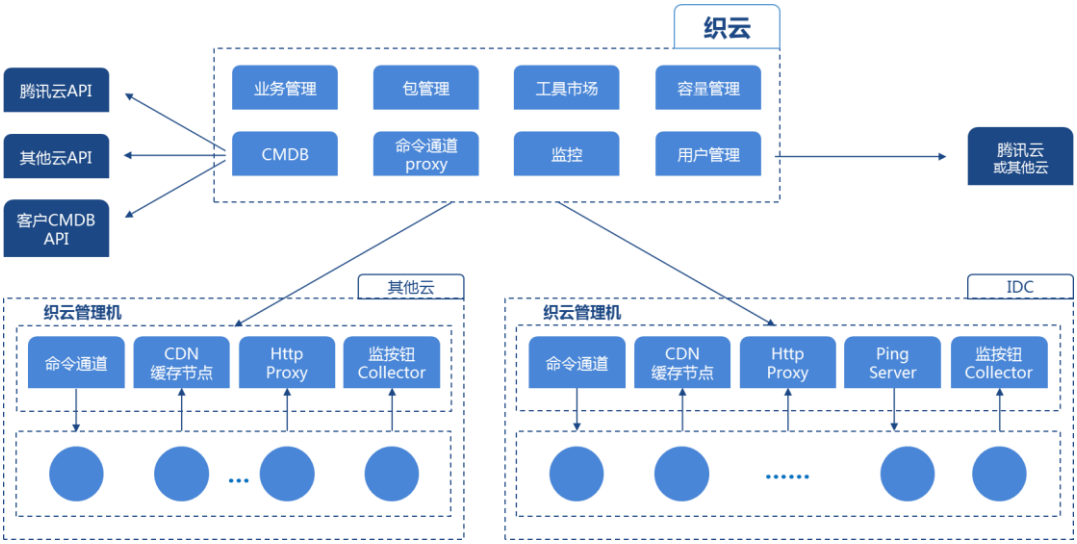


圖 3-16 多雲管理的技術架構

5) 作業平台

在持續維運的工作過程中，存在著大量的重複度高的工作需要被執行，經常會出現需要打開很多個終端執行一個腳本的場景。為降低用戶重複開發工具和傳承維運經驗的問

題，作業平台也就應運而生。從使用者（維運人員）的角度出發，平台設計時將工具作為整個系統的核心進行建設。圍繞著工具做文章，將工具從創建、編輯到執行以及執行記錄的整合納入持續維運的系統能力中。作業平台與命令通道如圖 3-17 所示。

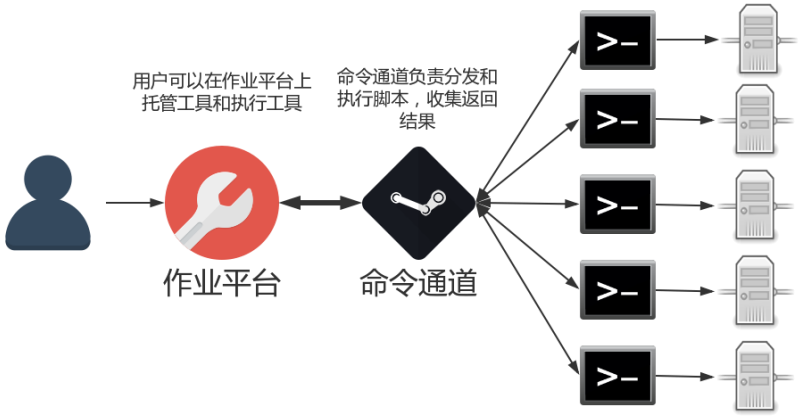


圖 3-17 作業平台與命令通道

在作業平台中，工具作為最小的操作單元，將常見的維運操作從日常的維運場景中抽離出來，獨立為一個最小的原子性操作，賦予相應的描述資訊和基本特徵（比如執行環境、超時時間等），像程式設計語言中的函數一樣，具備最基本的獲取輸入參數以及返回值的能力。作業平台傳參介面如圖 3-18 所示。

圖 3-18 作業平台傳參介面

透過以上對於工具的基本資訊，對使用者一個個的腳本進行包裝，對於使用者來說，

再也不需要過多地瞭解腳本的內容，可以從工具的描述以及輸入參數的描述中瞭解該工具的功能以及參數的作用。

作業平台提供了一個簡單的介面供使用者執行工具使用，執行工具時，只需要傳入工具所需要的輸入參數，並指定工具的執行機器，剩下的工作，包括將工具分發到指定機器上，發起執行，收集執行結果等，都交個作業平台來解決。

工具在自動化維運體系中的定位是一個最小的操作單元，即專注於完成一項特定的任務。而實際的維運場景中，往往需要多個工具組裝起來一起解決一個特定的問題。作業平台中的工具編排如圖 3-19 所示。

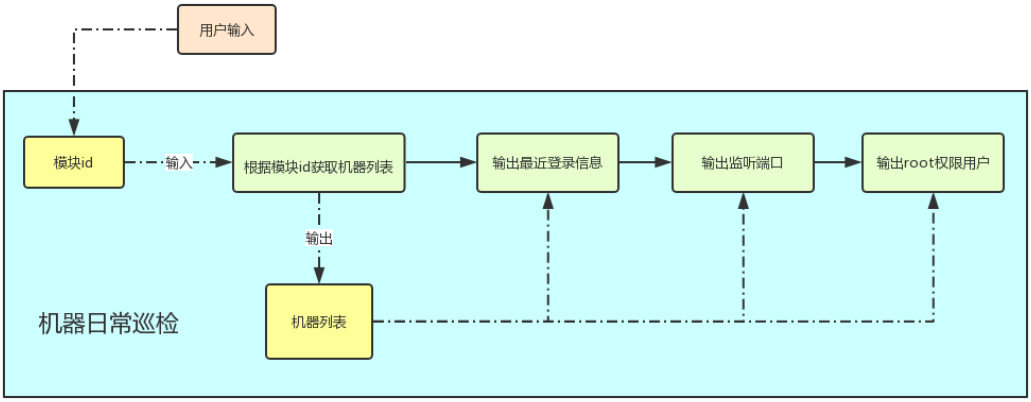


圖 3-19 作業平台中的工具編排

作業中的工具編排的方式來解決工具間的依賴關係，編排的範本有如下資訊：

1. 编排基本資訊：包括编排名稱和描述；

2. 编排的參數：當前编排需要用户輸入的參數清單；

3. 编排步驟：

- 每一個步驟都對應一個工具，步驟與步驟之間串列執行；
- 工具所需要的參數可以從其他工具的執行結果或者編排參數中獲取；
- 可以指定當前步驟執行完成後是否需要使用者確認後再繼續執行。

工具編排截圖如圖 3-20 所示。

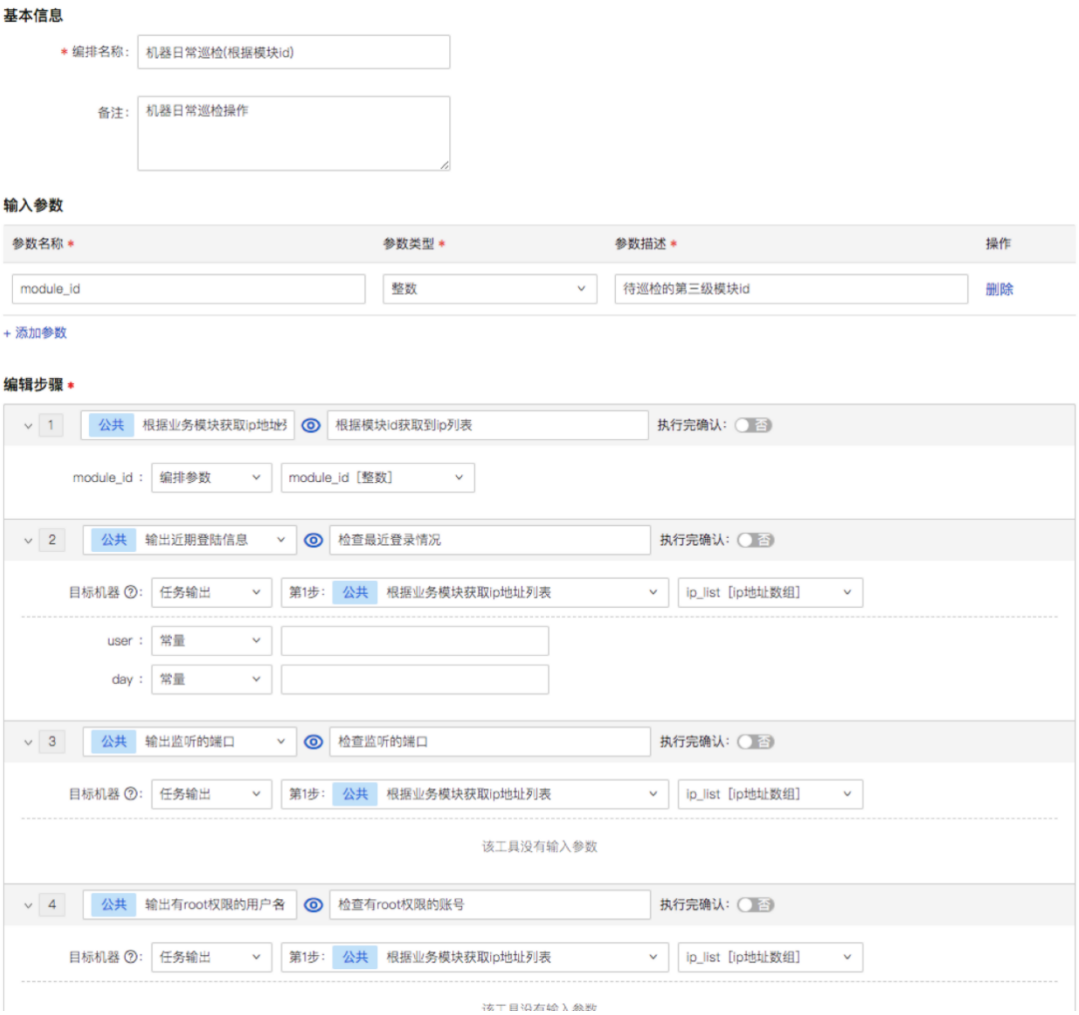


圖 3-20 工具編排截圖

作業平台的擴展能力，將工具編排擴展到維運系統的介面調用編排，包括 CMDB、包系統以及監控等介面的按維運場景組合，這都是圍繞著自動化維運能力去建設的。作業平台作為一個子系統，在持續維運的體系中的定位，是對於各個平台所提供自動化維運能力的整合和擴展。因此，在織雲的體系中，工具不可避免地會大量調用到其他系統提供的介面，實現資料查詢或者自動化操作處理等作業。作業平台在持續維運體系中的位置如圖 3-21 所示。

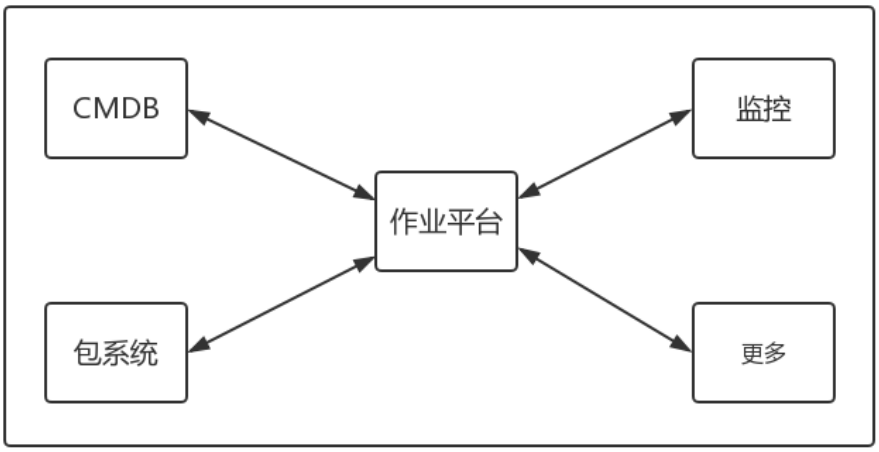


圖 3-21 作業平台在持續維運體系中的位置

6) 維運數據銀行

在建構持續回饋的維運能力時，核心的技術要點是要在立體化監控的能力基礎上，完成監控資料的打通與整合使用。

騰訊維運團隊的立體化監控技術架構的實現，結合了分層的維運管理方案，對不同架構層級的維運物件分別選型不同的監控方案支持，這種「因地制宜」的監控管理方案，符合企業分階段投入與建設的需求，也是大部分企業監控體系的建設過程。按架構層次的立體化監控設計如圖 3-22 所示。

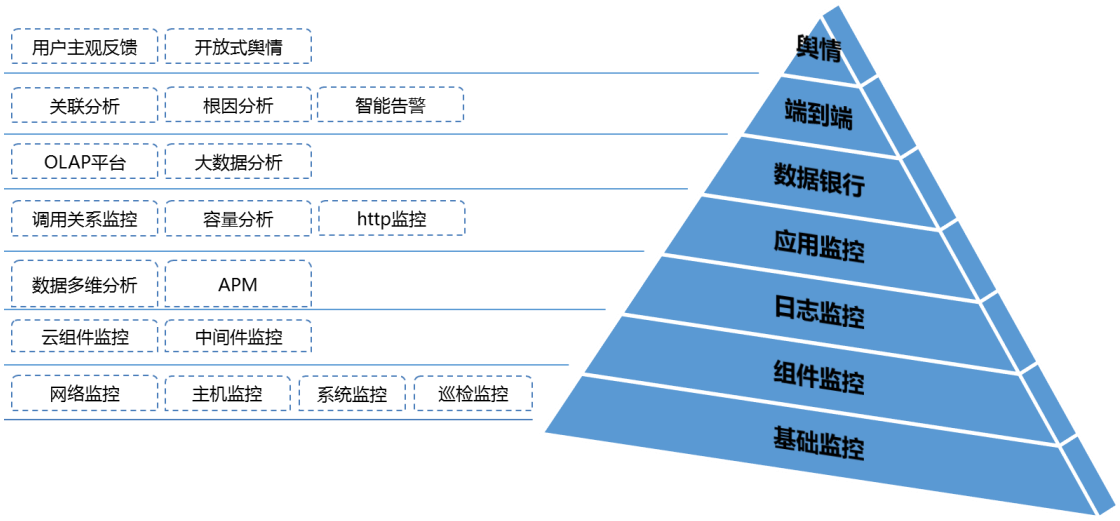


圖 3-22 按架構層次的立體化監控設計

利用架構的分層管理思路，持續回饋體系能夠更好的對監控資料分類，從而能夠說明維運在處理大規模的監控資料和告警時，能夠有更好的對資料進行優先順序排序和收斂。這就是立體化監控體系的好處，是實現統一監控告警平台重要的前置條件。

基於立體化監控，維運還沒法達到在緊急故障發生時，快速精準的定位故障的根源，因此從技術上要進一步找到優化資料的辦法。如下圖的將立體化的監控資料分成低層次指標與高層次指標，在立體化監控體系中，越接近基礎設施的監控指標，越具備共性，換而言之越不具備業務獨有的監控屬性，對於低層次指標的通用處理辦法是透過日常巡檢等工作實現主動預防，透過規則化的工具實現簡單故障自癒邏輯等。

高層次指標是直接反應業務品質的指標，技術實現多與業務邏輯耦合，是監控業務品質的核心監控技術，這類指標能夠直接作為量化可用性的指標依據，是品質營運的重要資料來源。但是，在大規模的維運監控方案中，高層次指標的數量級已經遠超出人工管理的範疇，儘管可以透過技術收斂絕大部分的無效告警資訊，但是在騰訊社交網路業務的案例中，告警資訊仍然無法靠人工完成梳理。為此，需要有更進一步的提煉監控資料的方案。高層次與低層次監控指標如圖 3-23 所示。

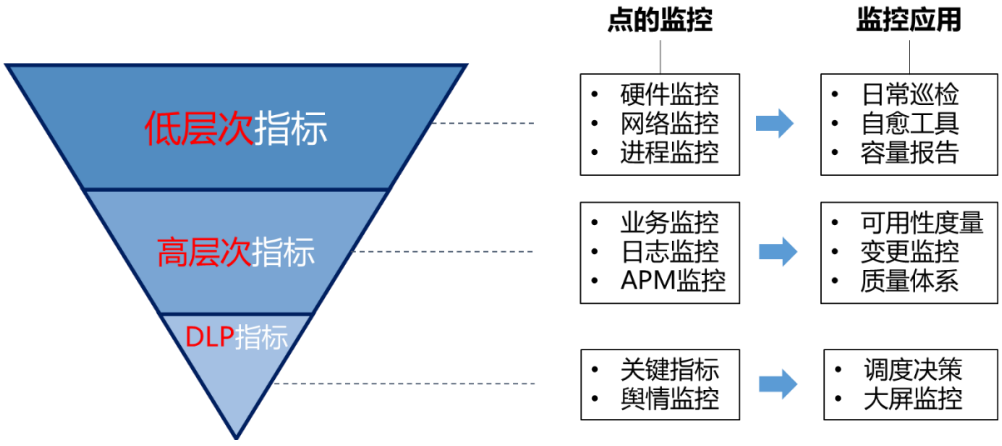


圖 3-23 高層次與低層次監控指標

立體化監控體系的分層思路，依舊是傳統的監控技術實現方案，要突破傳統持續回饋技術的瓶頸，騰訊維運提出了維運數據銀行的技術方案，透過對維運監控資料特徵的提

煉，利用大數據平台技術，實現對異構的時序類維運監控資料的統一處理、分析、決策的功能。

數據銀行將維運數據的數據流程處理分成四大部分：

- 數據採集，該模組遮罩異構監控數據來源採集方式和協定不一致的問題，同時還能主動發現監控數據上報時的髒字段和請求平滑度等指標，從而實現監控的監控；
- 數據分析，對於時序數據處理有很多共性，包括過濾、格式化、分組、聚類、翻譯、轉發、四則運算等等，透過抽象流式資料的處理能力，能夠更好的複用維運大數據技術的成果；
- 數據應用，服務於不同的維運場景，數據視覺化的表現不同，即時高效、關聯分析、多維排障等，都是數據應用於具體場景的方案；
- 預警與告警，持續回饋資料的價值，需要指導維運的工作，扭轉被動維運的局面，甚至形成決策數據，結合作業平台實現智慧化維運的功能。

數據銀行技術架構如圖 3-24 所示。

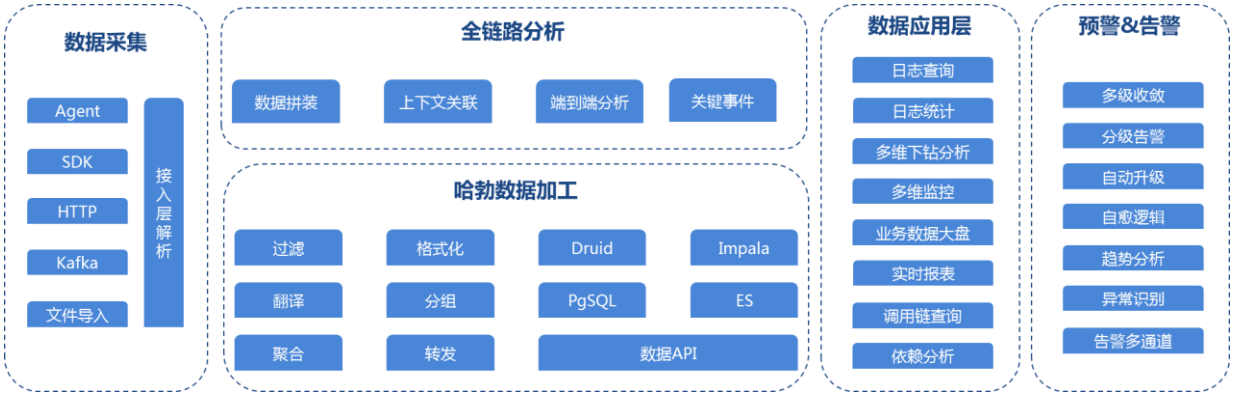


圖 3-24 數據銀行技術架構

（三） 解決方案特點

騰訊持續維運的技術實踐與解決方案，是網際網路技術思路在維運領域的應用，本文對持續維運和持續回饋的技術架構，由不同功能的服務組成，服務與服務之間的訪問

與調用均透過 API 介面實現互動，而維運平台本身也是運行在分散式的微服務架構之上，維運平台本身的佈署與調度則由 K8S 平台完成。

一體化平台的整體建設思路是透過實現 CMDB 配置中心，利用作業平台的工具編排能力建構場景化的自動化維運，再利用數據銀行統一採集和處理，有效的把維運資料轉換成決策依據，從而反向驅動維運自動化工具，逐步迭代收穫無人職守的維運能力。

3.3.4 總結

（一）經濟/社會效益

本方案是純軟體的維運解決方案，適用於雲計算時代，傳統企業的維運團隊在面對 IoT、「網際網路+」等不同源自業務側的規模化的挑戰下，平滑轉型升級的需求。讓傳統企業在不拋棄不重構原有維運平台的前提下，最大化的整合異構的軟體和異構的監控技術，實現一體化維運平台的目標。

（二）用戶評價回饋

騰訊在雲計算時代也推出了騰訊雲的商業化產品，其中本文介紹的一體化維運解決已經在金融、能源、汽車、醫療等行業龍頭企業中實施，透過與騰訊先進的維運經驗的結合，幫助企業從傳統的維運模式與技術，逐步迭代過渡到網際網路的維運模式與技術，實現了維運價值化轉變的過程。

（三）小結

以服務於騰訊社交網路業務的維運平台織雲的技術實踐案例，介紹了一體化維運平台的幾個關鍵技術架構與設計思路。希望幫助企業在建構持續維運與持續回饋的技術道路上，更能看清 DevOps 技術能為企業帶來的價值與變化，讓企業維運從成本中心轉變成價值中心，從維運到技術營運。

3.4 華為—華為雲軟體發展服務 DevCloud

3.4.1 案例簡介

（一）案例背景

華為雲軟體發展服務(DevCloud) 是集華為近 30 年研發實踐、前瞻研發理念、先進研發工具為一體的一站式雲端 DevOps 平台，面向開發者提供包括項目管理、源碼託管、流水線、源碼檢查、編譯建構、測試管理、移動應用測試、佈署、發佈、CloudIDE、研發協同等基礎功能的研發工具服務。覆蓋軟體發展全生命週期，支援微服務開發、移動應用開發、IoT 開發等主流研發場景，讓軟體發展簡單高效。

（二）案例特點

DevCloud 平台針對網際網路開發運營、產品軟體提供商（ISV）、傳統企業網際網路+轉型、新創育成園區、高效培訓機構以及軟體外包等方面有著多年研發、應用經驗，滿足為大量企業提供不同的應用需求。

3.4.2 需求分析

如今，想要加速傳統 IT 模式的革新，瀑布模型已經不再適用，需要更快捷、更符合快速迭代開發的敏捷開發模型。但是，在嘗試轉型敏捷開發的過程中，由於沒有配套的工具、流程指引，開發效率反而不如之前用瀑布模型的時候。

其次，企業在處於快速增長期時，業務劇增，這些情況導致業務源碼量急劇加大、源碼品質得不到保障，如果再透過人工檢查的方式去做，不僅效率低，也需要投入大量的人工成本。

最後，產品在以前都是維運人員手工佈署升級，但是為了回應用戶需求變化，滿足市場要求，過快的需求變更，讓維運人員痛苦不堪，連續長時間工作，導致維運人員極易出錯、影響業務。目前，在各方面存在的問題有：

（一）網際網路開發運營

網際網路企業在面對市場高速變化，產品盈利窗口窄時，經常由於研發工具難以滿足項目實際需求，導致難以及時交付高品質的產品給客戶。另外，企業的研發能力也難以度量，無法數據化依據判斷新專案的接單能力。

（二）產品軟體提供商

ISV 企業在研發過程中，存在開發人員辦公地點不同，研發工具、環境不統一，導致溝通交流困難；客戶需求變化快，導致專案極易返工，需要快速應對需求變化；另外自動化的持續整合也尤為重要。

（三）傳統企業網際網路+轉型

傳統企業在進行網際網路+轉型的過程中，由於對網際網路行業瞭解不足，以及本身傳統管理模式中存在的弊端，導致轉變方向不明確，核心競爭產品研發效率低下，技術手段落後，轉型難以推行。

（四）新創育成園區

軟體園區/育成中心在進行多企業協作式研發時，由於各企業辦公地點不同，研發工具、環境不統一導致的專案協作成本高，數據及資訊共用流程複雜不規範等問題。

（五）高效培訓機構

受應試教育影響，學生對課堂理論知識接受能力強，而運用知識解決實際問題偏弱；多數學生在個人能力發展過程忽略了對動手能力、職業素養、團隊協作意識等方面的培養；精心制定的教學計畫與內容難以跟隨 IT 行業快速變化的技術理論與前瞻趨勢；學科競賽、實驗專案推進、綜合實訓缺少統一規範化的流程與平台。

（六）軟體外包

軟體發包方難以掌控產品以及專案的進度，對於產品的品質在交付階段才可以驗證，

希望可以隨時瞭解產品研發進展以及查看產品現有功能。軟體接包沒有平台化的工具進行協作對接、分析判斷，難以應對發包方快速變化的需求和高標準的品質要求。

在這些需求背景下，華為 DevCloud 如及時雨般的出現並給出了一種新的解決方案。

3.4.3 解決方案

（一）總體技術架構

DevCloud 軟體發展雲架構設計遵循全面解耦原則，管理平面和業務平面分離；管理平面包括營運服務和維運服務，業務平面按軟體發展業務邊界拆分八大核心工具服務，平台性公共服務按橫向領域拆分成微服務，將服務之間的耦合降到最低。

以八大軟體發展工具服務為核心，全面服務化和元件化，引入新服務治理框架，所有服務介面可見、可控；服務逐步微服務化，可獨立開發、建構、測試、發佈、佈署，解決方案可以靈活組合，滿足多種應用場景。

DevCloud 總體邏輯架構圖如圖所示。軟體發展雲提供多種接入方式：APP 行動端、Web 端、OpenAPI，可以隨時隨地的進行軟體交付。

它還提供八大核心服務：專案管理、源碼託管、源碼檢查、編譯建構、流水線、測試佈署、發佈。支援把業務軟體佈署到開發環境、整合環境、生產環境等不同的研發與運營環境，並支援發佈到應用超市。

對於管理面來說，支援維運服務和運營服務。其中，維運服務包括監控資料、日誌處理、告警處理等。運營服務包括產品定價、計費、訂單管理等。DevCloud 技術架構圖如圖 3-25 所示。

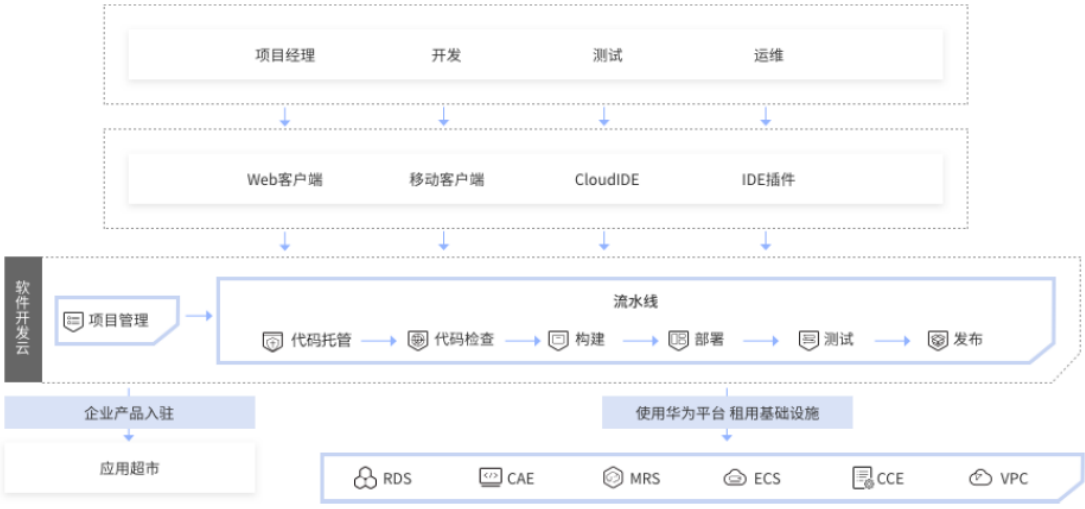


圖 3-25 DevCloud 技術架構圖

（二）具體技術方案

1) ISV 場景

DevCloud 開發流程如圖所示，專案管理功能以 Scrum 敏捷開發方法為基礎，從專案規劃開始，引導用戶進行 Epic、Feature、Story 的設計劃分。在計畫會議中引導用戶將 backlog 中的 Story 進行迭代劃分，再在迭代列表中按照相應的任務一步步執行即可。員工們再也不用為不懂敏捷開發、不知道下一步做什麼而煩惱，只需按照軟體發展雲的流程逐步推進即可。

其次，軟體發展雲總結華為近 30 年研發實踐，提供近千條的源碼檢查規則，只要將源碼傳到華為軟體發展雲的源碼倉庫中，然後選取相應的源碼檢查規則，一鍵點擊，就可以自動進行編碼風格、架構設計、編碼安全等問題檢查。針對檢查出的問題，華為軟體發展雲還給出了相應的正確示例、錯誤示例等，新員工們可自行進行檢查和修正，節省老員工大量的人工檢查以及後續的跟蹤修正時間。

最後，華為軟體發展雲提供的流水線功能，是對 DevOps 前瞻理念的踐行。透過華為軟體發展雲將源碼檢查、編譯建構、佈署功能集合成一條流水線，每次需要變更需求、

源碼修正時，只需點擊流水線的開始執行按鈕，即可按已設計好的流水線自動執行，真正做到了持續交付，解放了維運人員的同時，也大大提高了工作效率跟品質。DevCloud 開發流程如圖 3-26 所示。

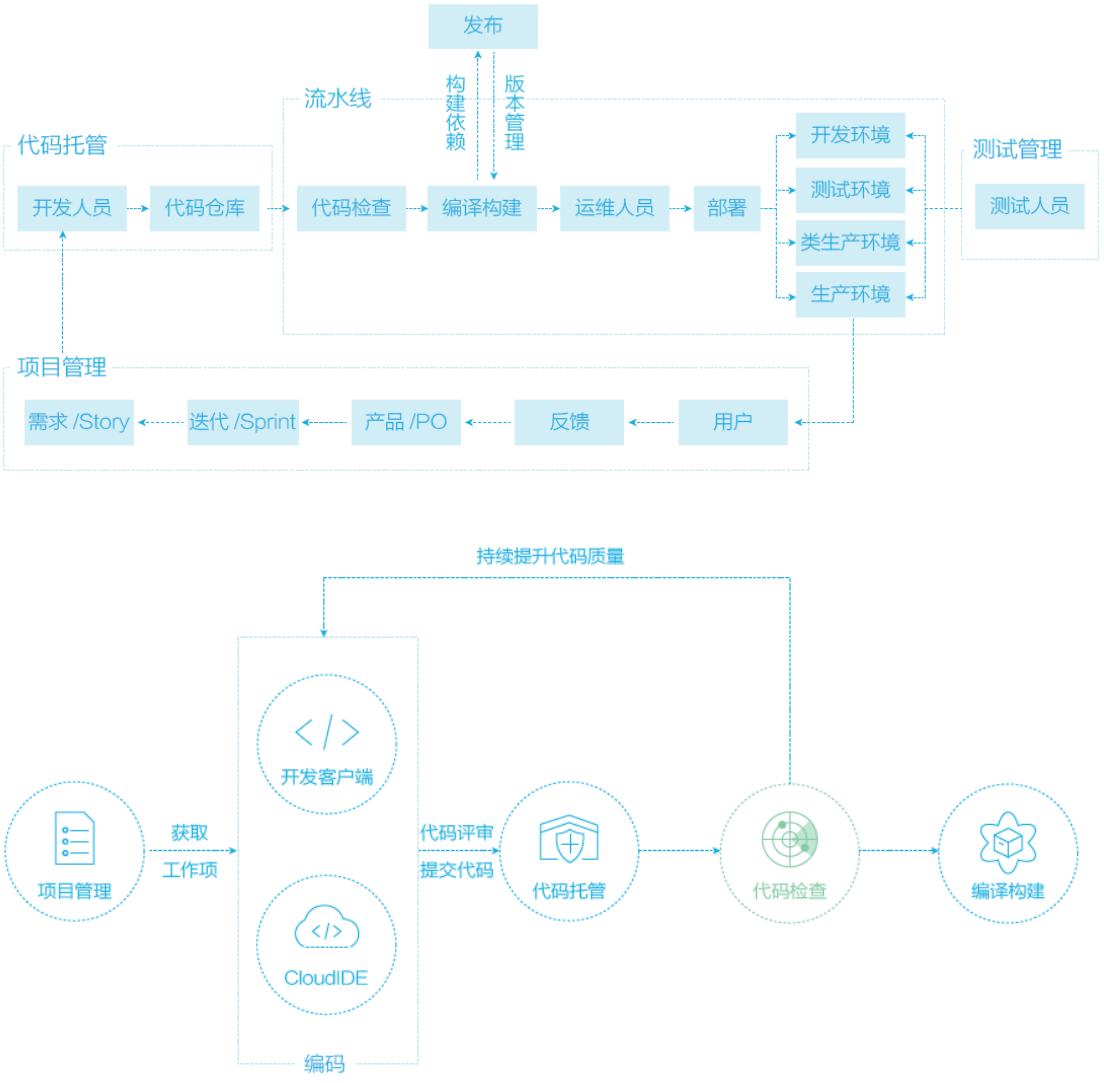


圖 3-26 DevCloud 開發流程圖

使用 DevCloud 前後專案狀態對比如圖 3-27 所示。

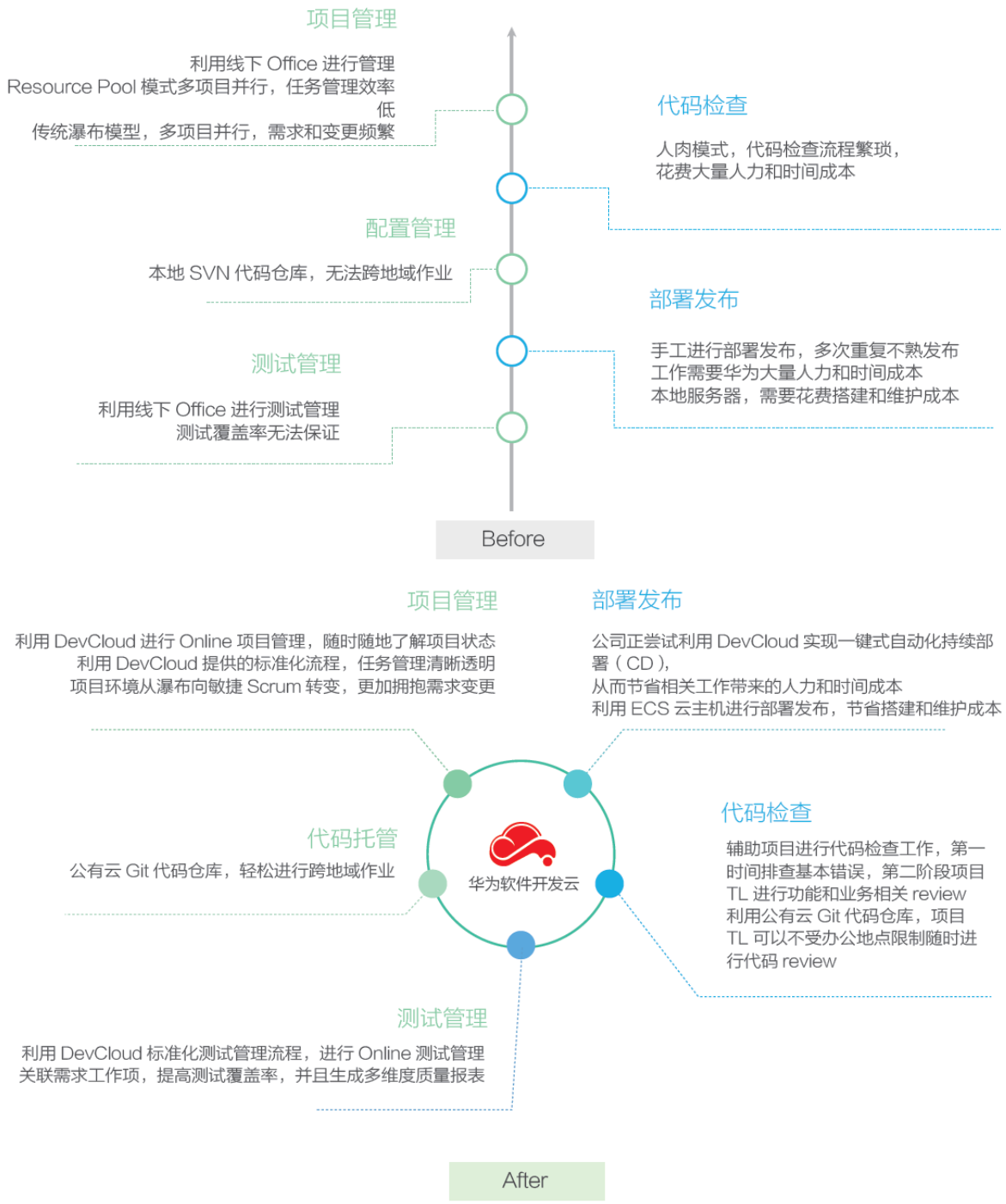


圖 3-27 使用 DevCloud 前後專案狀態對比

2) 傳統企業網際網路+轉型場景

傳統企業在進行網際網路+轉型的過程中，由於對網際網路行業瞭解不足，核心競爭產品研發效率低，轉型很難推行。而軟體發展雲提供的端到端整合工具鏈，適用于各種應用場景。軟體發展雲面向開發者提供一站式雲端 DevOps 平台。透過軟體發展雲資深

專家現場指導，讓軟體研發簡單高效，讓傳統行業轉型更加便捷。

華為軟體發展雲提供的雲端專案管理功能，使得專案發包方跟外包公司在同一個專案中進行協同開發與交付。發包方在項目中提出需求，並跟蹤專案進展，完成產品驗收。外包公司在專案中對需求進行分析、開發、交付與問題修復。並且軟體發展雲專案管理提供歷史紀錄以及工作項關聯源碼等功能，使得二次開發的時候，能夠有據可循、有理可依。

另外，華為軟體發展雲提供的源碼檢查功能，支援 Java、JavaSccript、CSS、HTML、Android 等主流開發語言，透過華為軟體發展雲的源碼檢查功能生成的測評報告，源碼品質一目了然，各種源碼問題也被一一羅列，徹底解決了無法對源碼品質進行把關的難題。另外華為軟體發展雲提供的行動應用測試功能可以線上對 APP 進行相容性及 APP 性能採集，並提供上下文截圖及異常日誌，同時可提供 CPU、記憶體使用率及回應時長等關鍵數據。資料化、專業化的測試，為 APP 的品質做了良好的保證，同時也省去了購買大量真機的費用。

流水線服務將開發與維運相結合，真正實現一鍵佈署。雲主機功能也徹底免除了企業獨自搭建、維護的工作。

3) 多企業協作式研發場景

在大眾創業、萬眾創新的新浪潮席捲全國時，華為軟體發展雲為眾多初創團隊提供貼身的「資本、技術、人才、市場」的全方位服務。在某些專案中，需要分散在不同領域的多家在孵化企業進行聯合開發。雖然這種協同交付可以實現最大化的合作，但是在實際聯合交付中，新創工廠及企業面臨了諸多挑戰。主要問題在於：

- 管理流程與工具不統一，跨地域團隊協同難度大

由於項目交付中涉及眾多企業，企業間在管理流程、研發模式、工具平台等方面不統一，這使得協同難度加大。源碼的維護和協同管理成本很高、最佳實踐經驗無法共用，導致研發效率低、開發品質不盡如人意。

- 客戶需求、專案進度資訊無法及時同步

專案的有效實施得益於企業與客戶的及時溝通，但在專案實施中，專案進度來源於一線的人員收集匯總後向客戶彙報，客戶無法第一時間瞭解專案情況。同樣客戶的需求變化也無法及時同步給專案開發團隊，甚至還存在資訊不一致、溝通滯後等情況，給專案運作帶來不必要的成本。

- 軟體生命週期相關環節缺失

專案版本源碼合併需要手工完成，無法支援分散式開發；版本靠手工編譯，持續整合時間太長，沒有自動化流水線；沒有體系化的源碼檢查工具，測試用例管理工具等。

華為雲端多團隊聯合開發解決方案如圖 3-28 所示，軟體發展雲為多企業協作研發提供了：

- 統一管理流程與工具，立足提供軟體發展生命週期全流程支撐；
- 透過迭代管理，讓客戶以及專案管理者能夠及時準確掌控專案品質與進度，在把控風險的同時最大化的回應客戶不斷變化的需求；
- 面向開發者提供的基於雲的服務，即開即用，包括雲主機、雲源碼託管，只要有電腦和網路就可以使用；
- 基於 Git 的高可用源碼託管，完美實現跨地域協同開發，讓參與專案的開發者不再是孤立的個體，增強團隊間的協同能力，高效、快捷的解決了跨地域的問題；
- 整合了近 30 年研發經驗範本的源碼檢查功能。

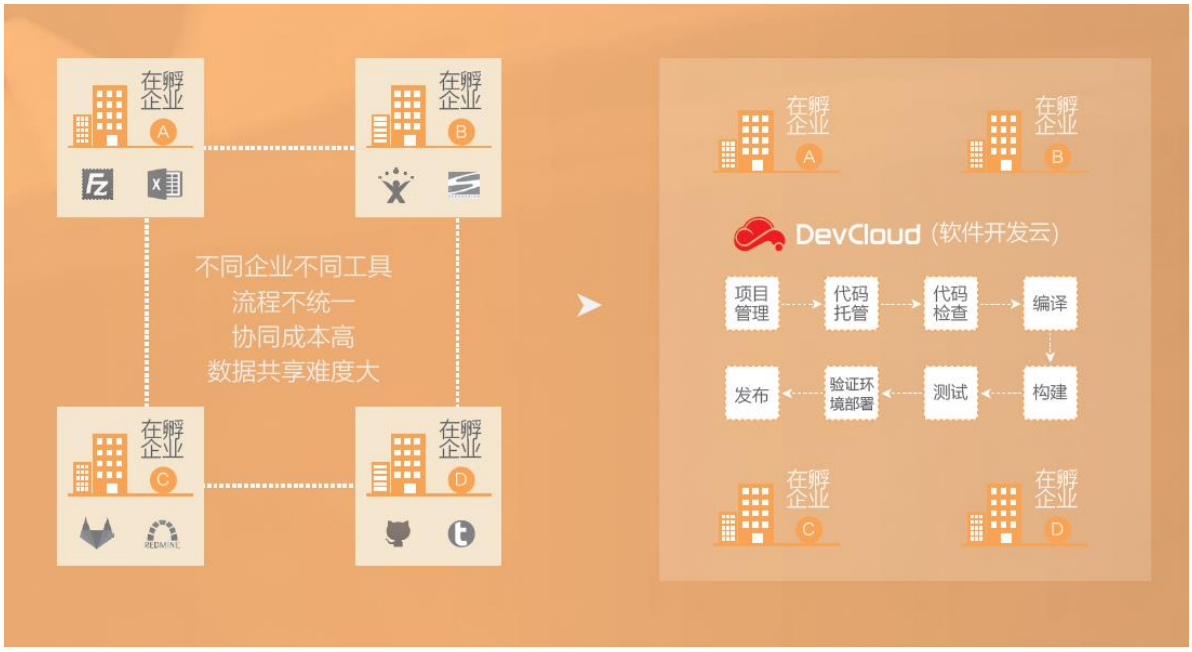


圖 3-28 華為雲端多團隊聯合開發解決方案

3.4.4 總結

（一）經濟/社會效益

在 ISV 方面，依託於華為推出的軟體發展雲產品，某公司得以擺脫傳統軟體發展方式的束縛，解決了許多開發過程中的痛點。使得公司軟體發展可以在一個完整的閉環中進行，大大提高了技術人員的工作效率，同時對工作情況提供了量化的指標，方便管理人員進行高效透明的管理。華為軟體發展雲可以說是送給軟體公司的一把利器，該公司快速實現了敏捷開發模式的轉型，實現了自動化維運佈署的實踐。在工作方式上，可以更加專注於優勢業務領域這一關鍵點發力，推動產品化改革快速進行。

在傳統企業網際網路+轉型方面，某企業團隊將一些流程化的事務全部交由軟體發展雲自動化實施的流水線模組後，無需再做重複而沒有變化的工作，減少了工程師的工作量，將更多精力放在提高使用者體驗和軟體品質上。軟體發展雲整個流程基於敏捷開發的理念，幫助該企業用小步快跑的開發迭代形式，取代傳統的瀑布開發模式，使團隊的開發效率較之前提升 30%。

在企業協作式研發場景方面，華為軟體發展雲實現了某智慧城市項目的提前交付，並在客戶驗收中獲得一次性通過。在整個開發過程中，客戶與研發團隊保持了良性互動，項目交付週期由當初的六個月縮短到三個月，版本編譯建構時間降低到分鐘級，缺陷率下降了 20%。

（二）用戶評價回饋

「華為軟體發展雲，為我們帶來了包含專案管理、源碼檢查、編譯建構、測試佈署、發佈等在內的一站式可持續交付解決方案，很好地解決了公司在專案併發管理、敏捷迭代開發過程中遇到的各種難題，在實現了專案高效跟蹤、透明化管理的同時，又以低廉的成本滿足了可持續交付過程中的業務需求，給我們帶來了新的體驗，從而驅動了業務的創新發展。」某大連公司總經理深深感悟到。

透過在軟體發展雲平台上對系統遷移專案的開發實踐，某公司產品的開發效率和品質都得到很大幅度的提升，縮短了開發週期，節約了人力成本的付出，同時硬體環境成本付出也有較大的消減。更有信心為客戶提供更為快速、低價的服務，提升了企業在行業內的服務競爭力。該公司第三事業本部部長表示：「軟體發展雲充分站在開發者的視角，為消除軟體發展企業的煩惱提供了一個優秀的解決方案。從專案管理、源碼檢查、編譯、佈署、發佈，貫穿整個專案開發週期，為軟體發展者提供了比較完善的服務。雖然我們只對其中一部分服務做了實踐，但是對於開發工作已經起到了較大的幫助。提升了作業品質與工作效率，消減了較大專案開發支出。」

「公司專案應用軟體發展雲後，極大提高了專案管理效率，在統一的平台，就可以完成整個軟體發展週期，需求分析、思維導圖、源碼管理、自動建構、測試和統一發佈。同時華為開發雲支持佈署到其他雲平台，真正站在用戶的角度去考慮問題。」某國際公司劉工表示。

（三）小結

隨著新 IT 時代的到來，創新技術已經開始顛覆傳統行業，發展速度飛快，IT 服務需求進入新的旺盛期，需求更趨於服務化、碎片化、定制化。華為軟體發展雲公測，作為全球領先的 ICT 方案提供商，華為透過雲服務的方式將公司近 30 年積累的軟體工程能力和優秀實踐開放出來給廣大的企業使用，提供了從項目創建到源碼託管、源碼檢查、編譯建構、測試、發佈等軟體發展全生命週期服務，為客戶和軟體服務商提供了一個一站式線上開發平台，極大的提高了軟體發展的效率，更快速的應對客戶需求的變化，能為開發者提供更便捷的雲上開發服務。

第四章 海峽兩岸 DevOps 關鍵技術

DevOps 的出現是為了實現軟體發展人員和維運人員之間更好的溝通協作，自動化的流程使得軟體建構、測試和發佈更加快捷、頻繁和可靠。透過 DevOps 的佈署實施，團隊內部的「隔閡牆」被打通，使用者需求鏈、軟體產品交付鏈及創造價值鏈形成快速、高效運轉的精益閉環，溝通協作更加快速、交付目標更加明確、開發損耗更加精簡、資源配置更加優化。

企業要轉型和實現 DevOps，必須要完成內部的開發、測試、佈署及維運等資源的雲化。雲計算帶來的 IT 資源彈性伸縮、按需供用等技術紅利，已經讓許多企業嘗到甜頭：

- 測試環境的自動化管理，讓用戶可以永遠輕鬆地使用到健康的環境；
- 帳戶、角色及 IP 配置的建立，讓每個用戶擁有獨立、安全的開發環境；
- Docker、Kubernetes 等輕量化工具的出現，讓實際大型軟體的開發、測試和佈署變得從未有過的快速、簡單。

DevOps 在彼時資訊時代的歷史轉折風口上，被人們寄予了更多的希冀。現概括性地給出若干重要 DevOps 技術發展路線內容，僅供讀者參考。

4.1 Docker 及應用編排技術

應用編排是實現 DevOps 的重要環節之一，它是指將系統中各個元件按照其依賴關係進行自動化的安裝、佈署和啟動，從而可以實現系統自動佈署和快速伸縮。隨著業務的不斷擴充以及雲計算技術的不斷進步，基於傳統虛擬機器的應用編排技術逐漸暴露出啟動速度慢、資源細微性大和額外開銷大等缺陷。Docker 的出現讓 DevOps 邁出了更加堅實的一步——容器化。以 Docker 為代表的容器技術在各方面均達到或超過以 KVM 為代表的虛擬機器技術，前者不僅繼承了後者應用編排技術的優勢，同時還在一定程度上彌補了後者不足之處、提高了系統資源利用率。

Kubernetes、Docker swarm、Mesos 和 Azure 等容器編排技術工具的出現，使得集群維護和管理變得更加便捷。近年來，Docker 實現了突飛猛進的發展，甚至在業內部分專家以團隊採用 Docker 與否作為是否具備 DevOps 的標誌。以 Docker 為中心的完整資料中心解決方案在不斷的整合開源社群的零散工具並形成最佳實踐，各大廠商也相繼開始推廣自己的企業級整體收費解決方案，這也表明以 Docker 為核心的資料中心方案正在逐漸走向成熟，Docker 及其編排技術依然具有巨大的發展前景。

此外，實際應用環境中會包含多個且跨越伺服器主機佈署的容器，利用 Kubernetes 可提供大規模容器的佈署、编排與管理能力，並實現建構多容器應用服務，完成容器的集群调度、伸缩以及狀態管理。Kubernetes 透過將容器分類组成「Pod」來解決容器劇增帶來的附加問題，並進而對 pod 進行負載均衡以保障各項任務的合理負載。同時，Kubernetes 還需依賴網路、監控、儲存、安全等組件來提供包括但不限於跨主機編排、合理硬件資源利用、自動化佈署、儲存擴展、狀態修復等服務內容。Kubernetes 的主要優勢在於可提供實體機或虛擬集群上容器的調度運行平台，實現可依賴性的容器化基礎設施。

4.2 微服務應用

在 DevOps 技術應用方面，微服務將是其主要的發展領域。微服務是指將整塊複雜的應用系統切分為多個簡單、獨立的應用，其不同的 XaaS 需求使得功能模組更小化和服務化，提高了跨平台的服務可複用性。企業需已具備 DevOps 能力是實現微服務「快速發佈、獨立佈署」的必要前提。

目前，如何拆解微服務是其技術應用的最大難點之一，領域驅動設計是一種比較理想的微服務拆解方法論，而社會化源碼分析則可幫助團隊透過更精確的資料找到更加合適的拆分點；另一方面，微服務系統之間的通信是一個很重要的問題，Kafka Streams 和

OpenTracing 是目前比較廣泛採用的成熟分散式訊息系統；最後，無伺服器架構（Serverless architecture）是 DevOps 技術在微服務領域的極致理論應用。當應用程式執行環境的管理被新的程式設計模型和平台取代後，既省去了很多環境管理中對設備、網路、主機及其對應的軟體和配置工作，又在一定程度上降低了團隊採用 DevOps 的技術門檻，使得團隊的交付生產率得到進一步提升。

4.3 安全工具

在安全方面，組織結構的轉型迫使企業打通「部門牆」，這意味很多傳統控制流程不再適用。另外由於 DevOps 技術中大量來源於開源社群，缺乏強大技術實力的企業在應用相關技術時難免有所擔憂，這促使安全將成為推動 DevOps 全面發展的重要力量。

目前，採用 Git-crypt 工具可以保證在開發過程中原始程式碼內部的資訊安全，而 HashiCorp Vault 工具則提供了脫離應用程式碼的秘密資訊儲存機制，使得應用在運行過程中得到有效保護。InSpec 作為合規性即源碼的提出者和實現者，透過自動化手段確保伺服器在佈署後的維運生命週期中依然保持安全與合規。

若從整體系統安全角度來看，可安排不同的開發階段進行不同的測試，但較常見的是在上線前確保安全無虞，避免系統對外服務時發生資安風險，例如網站弱點、源碼檢測、滲透測試、授權掃描、弱點掃描等，透過這些一連串的資安檢測，才能盡可能降低資安風險，而市場上相關資安檢測工具也已經發展相當成熟，例如 HP webinspect 檢測網站弱點, Nessus 的弱點掃描、BlackDuck 的授權掃描、Rapid7 的滲透測試以及 Fortify 的源碼檢測等等。

對於協作與共用的 DevOps 團隊來說，安全的重要性關乎到每一個人。同時，在未來 DevOps 技術的發展歷程中，更多易用且安全的工具將不斷出現，在降低 DevOps 所帶來的安全風險的同時，也提升了團隊開發過程的順暢性和用戶便利性。

4.4 自動化測試

在自動化測試方面，目前已有研究機構在研發相關的驗測機制，利用透過整合不同階段和面向需求的驗測方法，以確認 DevOps 應用的成熟度與效用指數，例如功能、效能、資安等方面，或驗證在開發、上線以及維運等不同階段中的關鍵測試指標，如事件流程、工具、檔報表和測試資料等。透過自動化測試機制，發現和聚焦 DevOps 佈署過程中的孱弱環節，並回饋以強化改善實施方法，加強軟體發展管理品質，提升佈署和導入的具體效益。最後，非功能性自動化測試工具的逐漸完備以及更加完善的工具集，將使得 DevOps 更加精益，也將引領更多企業佈署 DevOps。

過去 DevOps 主要致力於解決開發和維運之間的代溝，但在未來與組織目標的結合可讓 DevOps 發揮其更大的價值。目前在大多數企業中，DevOps 依然僅聚焦於開發團隊和維運團隊間的協同工作，和企業的戰略目標鏈結不深，所以未來在制度、流程以及工具等方面還有非常大的演進發展空間。

4.5 流程管理

在流程管理方面，過去軟體開發流程受限於工具和技術架構的發展，以及專業分工的思維，因此瀑布式開發盛行，然而這種開發經常以年為週期的方式，在現今產品生命週期縮短只有數月的產業環境中，已無法因應，因此需要依靠 DevOps 打破組織藩籬，重塑更有效率的開發維運流程，來減少企業環境中常見非必要、重複的工作流程。透過建立 SOP 檢核表將有助於企業的流程管理，例如流程可訂定最優先的工作是找出作業瓶頸點，因為把力氣花在瓶頸點之前的改進，只會讓更多待辦工作卡在瓶頸點；而在瓶頸點之後的任何改進，只會造成後面的人和設備更多閒置，處於等待瓶頸點送來任務，形成時間和人力資源的浪費。因此，可發展類似檢核表來確認目前流程是否有符合 DevOps 的精神，或確認工作流程具備邏輯順序能夠被配置為自動化執行，來提供企業作為導入

步驟的參考依據。

理論上來說，大部分自動化操作都可在 Jenkins 平台上實現。Jenkins 是一個被廣泛應用於持續建構的可視化工具，可實現多種不同項目的自動化編譯、打包、分發和佈署。它支援 Java、c#、PHP 等多種語言，兼容 ant、maven 等第三方建構工具，同時可與 Github 等託管網站整合，并可自由佈署在 Windows、Linux、Mac 等各類平台，Jenkins 就好比是一個框架集，盡可能地整合任何內容，以實現建構持續整合體系。

4.6 領域型解決方案

在兩岸產業發展過程中，軟體產業在規模經濟的條件下有著明顯的差異，因此在 Devops 的需求上也有不同的思維角度，例如大陸有百度、阿里和騰訊規模的網際網路企業，以及為數眾多的衛星配合廠，因此 DevOps 的應用快而普及；臺灣則以半導體與 ICT 生產製造較為擅長，也同樣有為數眾多的供應鏈上下游廠商，但上述廠商的核心業務畢竟不同，營收的來源也不一樣，因此即使都有程式開發和系統維運的需求，但關心的重點不同，因此，製造業廠商除了希望瞭解 DevOps 的通用標準或工具之外，亦關心是否有面向製造業而展開的 DevOps 解決方案或特殊標準，來協助企業調適，進而享受 DevOps 能帶給企業的競爭優勢，其概念一如同為雲計算領域，但因製造行業的特殊方面，因此又分支出了工業雲，未來 DevOps 是否有機會演化調適出不同產業應用方案，仍有待後續觀察，另外，對於為數眾多的中小企業來說，如何面對 DevOps，是否有不同的應用方式或工具，也是未來值得關注的議題。

第五章 市場潛在 DevOps 標準化需求

5.1 DevOps 標準化需求分析

自 DevOps 概念產生以來，一直難以在企業中實現真正落地，其中一個原因在於目前佈署 DevOps 的門檻較高，比如 DSL 語言、解耦以及功能工具應用等，仍急缺大量人才和相關培訓工作。另外還有一個更重要的因素在於還未形成 DevOps 的相關產業標準化。

在 DevOps 初期，人們嘗試了很多技術路線，但最後都失敗了。主要是因為個別企業標準不代表整個行業標準，很難統一考量並對其進行推廣壯大。而 Docker 的出現之所以為 DevOps 帶來了蓬勃的生機和落地的曙光，這是因為：

- Docker 開發使用簡單，其在開發過程中無需關注實體機器的運行環境，能更加清晰地規劃開發和維運之間的聯繫；
- Docker 抽象層次高、解耦徹底，且已逐漸形成行業的通用標準。

由於各類企業的管理流程、價值導向及企業文化都不盡相同，其它許多實現 DevOps 的技術工具，不一定適用於所有企業。對容器而言，KVM 虛擬機器其實僅僅只是一個進程，而容器則可認為是增強版的進程，它相當於給每個進程做了一個「箱子」，雖然容器都是運行在作業系統中，但彼此之間相互做了隔離。「容器」好比是碼頭的集裝箱，在集裝箱發明之後，整個運輸行業實現了標準化、自動化，效率得到了極大提升。這是 Docker 的價值所在，是 DevOps 重振旗鼓登上舞台中央的重要契機，更是 DevOps 行業發展的標準化需求價值體現。

標準是 DevOps 發揮其價值的必要基礎，DevOps 標準化工作是推動其相關技術、產業及應用發展建設的重要基礎性工作之一。為積極研究和明確 DevOps 標準化工作思路，加快 DevOps 基礎術語、技術產品及服務等相關標準研製，促進有關產業與國際 DevOps

標準化建設的協調發展，現給出部分 DevOps 標準化建議，僅供讀者參考。

5.2 DevOps 標準化建議

針對目前 DevOps 迅速發展面臨的挑戰，結合用戶需求、行業內外 DevOps 應用情況和技術發展狀況，建議 DevOps 標準化方向可以從基礎標準、產品/解決方案標準、安全標準、服務標準以及管理標準等方向展開，DevOps 標準化方向如圖 5-1 所示。

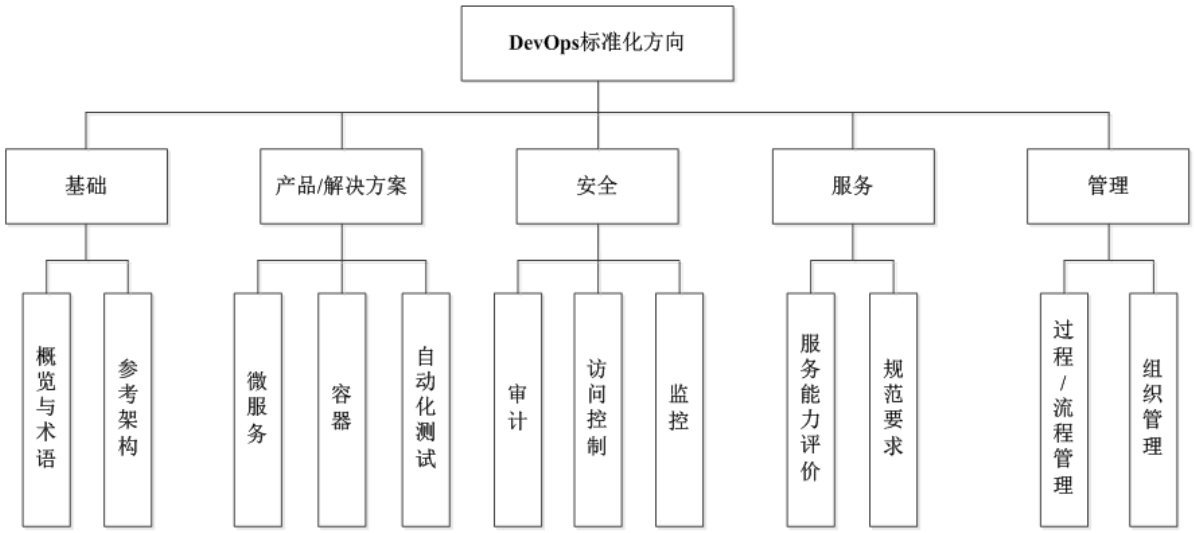


圖 5-1 DevOps 標準化方向

5.2.1 基礎標準

當 DevOps 如雨後春筍般爭相冒出大地的時候，也給業界帶來了一絲困惑。DevOps 產業的相關用語專業性、碎片性和繁雜性等特點，以及較專業化的概念，讓其在落地階段就受到些許冷淡的回應。另外，在技術參考架構和設計模式等方面，也存在百花齊放、相較不下等情況。在這個博弈的過程中，無形中也增加了概念落地的複雜度，這將對 DevOps 的佈署和落地帶來一定難度。

因此，在 DevOps 基礎標準中，形成較一致性、標準性和權威性的行業術語、技術參考架構等，對 DevOps 的概念落地、行業推廣和應用是基本且重要的前提保障，也是極其必要的。

5.2.2 產品/解決方案標準

傳統開發模式週期長、人工成本大、上線慢，已無法滿足目前的研發運營過程。Docker 為 DevOps 帶來了新的生機和模式，同時也引領了新一輪的應用工具開發潮流和架構模式創新風暴。同時，微服務+自動化測試等也為 DevOps 注入了新生血液。越來越多的性能優益的產品或解決方案，已顯著地提高了 DevOps 效率。微服務是鬆耦合的，因此，微服務更容易被測試和實現 CI/CD。而容器的輕量化讓微服務啟動很快，同時容器的跨平台性保證了微服務可以在不同的平台啟動起來。最後，要實現持續整合源碼，可透過自動化測試來降低快速合併後驗證的風險，而且若沒有自動化測試，測試環境很可能成為整個開發環節的瓶頸。

另外，DevOps 為網際網路企業需求的快速更動帶來一套完整的軟體開發解決方案，讓軟體開發速度加快的同時，亦兼顧品質，但不同的產業屬性對於軟體開發要求重點不同，例如製造業所使用的生產系統，來自於客戶的需求變動並不頻繁，而用於管控廠區內的設備功能遠大於提供對外服務，在種種使用條件上和網際網路企業有明顯的差異，因此，適用於 BAT 企業類型的 DevOps 解決方案或許有機會做某種程度的調整或修改，形成針對不同類型產業的解決方案標準，例如製造業 DevOps 解決方案，來滿足該領域獨有的企業特性。

產品/解決方案標準中包括微服務、容器、自動化測試及領域型解決方案如製造業等內容，主要針對行業中重要的技術產品或解決方案給出較詳細、完整及全面的理論支撐、技術架構及應用參考。

5.2.3 安全標準

DevOps 的落地讓軟體交付速度不斷提升，而安全問題也越來越凸顯。常常出現的問題有：系統上線後才發現安全性漏洞，而損失已經發生；同樣的漏洞在不同團隊中多

次出現，無法分析漏洞影響範圍；開發人員安全意識弱，造成 SQL 注入導致資料洩漏等漏洞。據《2017 年 DevSecOps 社區》調查資料顯示：

- 67%的 IT 企業將 DevOps 描述為非常成熟或者提高成熟度；
- 58%的成熟 DevOps 團隊將自動化安全作為持續整合實踐的一部分；
- 47%的傳統開發和運營團隊報告安全團隊和政策正在放緩。

為了將安全理念融入到 DevOps 實踐中，提高企業安全意識，最大化企業價值，目前主流的解決方法是引入自動化的安全機制，並貫穿整個 DevOps 生命週期。

安全標注重點關注存取控制、審計等方面，主要包括持續掃描、深度掃描、即時告警、內容驅動等要求。

5.2.4 服務標準

在企業完成 DevOps 佈署後，如何用、用什麼以及平台能力建設和評估等方面同樣也需一套完整的標準化方法進行指導和規範。同時，這對還未佈署或還不夠滿足實現 DevOps 能力的企業來說，給出一種相對較權威性和建議性的參照，形成一種規範和標準化的評估方法，更對整個 DevOps 行業的良性運轉、更新改進以及標準規範等方面都將起到積極的推進作用。

服務標準重點關注通用規範要求、服務能力評估等方面，主要包括佈署內容規範、服務能力要求、服務能力評估等內容。

5.2.5 流程/過程管理

DevOps 將傳統瀑布式開發的階段式流程改變成為開發和維運一體化流程，透過不同單位與人員之間的溝通、協作以及整合來落實 DevOps，其流程不同於以往的部分在於整體過程中嵌入更多的自動化步驟來取代傳統人工手動，例如自動整合、自動佈署以及自動測試，以及響應這些自動化過程而改變的程式。因此，發展標準化的開發維運一體

化流程管理程式，將有助於企業在導入 DevOps 過程中，進行量化管理、預測產出和持續改進，透過標準化流程(SOP)可以協助企業用檢查表(checklist)來檢視自己導入過程中是否有重工或瓶頸，以便快速改進。

5.3 總結

任何技術的產業落地若能有國際標準的指引，理當能發揮出標準的最大參考價值和技術紅利價值。目前，DevOps 相關國際標準還未正式發佈，僅有部分國際標準提供框架式指導。以臺灣雲端服務發展為例，在過去雲端運算國際標準尚未公佈時，即根據當時區域性產業規範或認證機制來發展一套準標準化的評估原則，例如根據當時美國 NIST 雲端特性定義、ECSA 歐洲雲端星級驗證、CSA 雲端安全聯盟認證、日本 ASP-SaaS 雲端認證等定義和要求，發展出一套具體而微的驗測專案（驗測表）、驗測方法、測試環境與測試工具的整體方案，並以服務方式輸出技術，協助產業及機關實現雲端服務創新設計和提升品質，即使正式雲端服務水準國際標準如 ISO/IEC 19086 許久後才公佈，但許多企業的雲產品或機關雲服務因先前已透過驗測改進，因而具備無縫接軌國際標準的品質實力。

為協助和組織企業實現、完善 DevOps 佈署，並提供具體的實踐方法和步驟，以及給出包括驗證、測試和認證等流程規劃，推動實現 DevOps 真正應用落地，並在行業內外發揮其應有的應用價值，方召集各方專家及研究機構形成此《DevOps 兩岸共通標準研究報告》。

附錄 1：名詞術語對照表

兩岸文字使用不盡相同，文化也有些許差異，因此對專有名詞與術語也有不同的表達方式，茲整理如下表，供讀者參考。

DevOps 共通標準研究報告名詞術語對照表

序号	陸方名詞	台方名詞
1	云计算	雲端運算
2	组织	社群
3	云计算服务	雲端服務
4	移动 APP	行動 APP
5	综合	綜整
6	通过	透過
7	运维	維運
8	运营	營運
9	测量	量測
10	构建	建構
11	大数据	巨量資料
12	软件	軟體
13	硬件	硬體
14	应用程序	應用程式
15	信息	資訊
16	部署	佈署
17	数据库	資料庫

18	服务器	伺服器
19	配置/设置	配置/設定
20	文件	檔案
21	默认	預設
22	网络、互联网	網路、網際網路
23	存储	儲存

附錄 2：DevOps 共通標準研究報告參編單位及人員名單

台方參編單位（排名不分先後）：台灣雲端物聯網產業協會、財團法人資訊工業策進會、台灣軟體工程學會、研華股份有限公司、走著瞧股份有限公司

台方參編專家（排名不分先後）：資策會副執行長 余孝先、資策會顧問 王瑋、研華技術長 楊瑞祥、台灣軟體工程學會秘書長 馬尚彬、台灣軟體工程學會理事 葉浚豪、走著瞧股份有限公司技術長 葉秉哲、研華資深研發經理 康寧、研發主管 靳秀華、研發工程師 周文文、數位無線總經理 陳文裕、得寬科技系統維運工程師 陳正瑋、工研院巨資中心資深研究員 周翊婷、資策會數位轉型所所長 何文楨、主任 陳立群、副主任 施賀建、專案經理 傅世卿、資策會智慧系統所副主任 李麗鳳

陸方參編單位（排名不分先後）：中國電子技術標準化研究院、華為技術有限公司、騰訊雲計算（北京）有限責任公司

陸方參編人員（排名不分先後）：中國開源雲聯盟秘書長 中國電子技術標準化研究院軟件工程與評估中心雲計算研究室主任 楊麗蘊、中國電子技術標準化研究院軟件工程與評估中心軟件工程實驗室/雲計算研究室技術負責人/高級工程師 陳志峰、中國電子技術標準化研究院軟件工程與評估中心雲計算研究室 陳行、騰訊雲計算（北京）有限責任公司 梁定安、舒季、聶鑫

附錄 3：中電標協和電子標準院簡介

中國電子工業標準化技術協會

中國電子工業標準化技術協會，簡稱中電標協。協會成立於 1993 年，是民政部批准的全國電子資訊技術領域標準化專業社團組織，業務主管部門是工業和信息化部。協會會員主要來自於企事業單位、相關部委、地方主管單位代表、高校研究機構、諮詢與檢測機構等。協會以“服務會員、服務產業、服務社會”為宗旨，宣傳政府方針政策，組織會員開展標準化研究、制定和國內外技術交流等標準化相關活動。

目前，協會在熱點和重點領域先後成立了高性能電腦、移動存儲、數位家庭、平板電視結構、海量存儲、企業資訊化、汽車電子、薄膜太陽能、醫療電子等十餘個標準工作委員會，為會員和企業搭建了標準化開放工作平台。為促進標準更好地服務於產業，協會還組建了版式技術、車載資訊服務、汽車電子基礎軟體、三維數位社會管理等創新產業應用聯盟。近期，協會在智慧財產權、節能減排以及企業社會責任等綜合性領域又成立了六個工作委員會。透過積聚廣大會員和社會標準化力量，推動相關成果在行業中的應用，服務產業發展。

中國電子技術標準化研究院

中國電子技術標準化研究院（工業和信息化部電子工業標準化研究院，簡稱「電子標準院」），創建於 1963 年，是工業和信息化部直屬事業單位，是國家從事電子資訊技術領域標準化的基礎性、公益性、綜合性研究機構。

電子標準院以電子資訊技術標準化工作為核心，透過開展標準科研、檢測、計量、認證、資訊服務等業務，面向政府提供政策研究、行業管理和戰略決策的專業支撐，面向社會提供標準化技術服務。電子標準院承擔 55 個 IEC、ISO/IEC JTC1 的 TC/SC 國內技術歸口和 17 個全國標準化技術委員會秘書處的工作，獲批國家物聯網、雲計算等多個領域國家重點實驗室，與多個國際標準組織及國外著名機構建立了合作關係，為標準的

應用推廣、產業推動和國際交流合作發揮了重要的促進作用。

附錄 4：華聚基金會和台灣雲協簡介

華聚產業共同標準推動基金會

華聚產業共同標準推動基金會現任董事長為陳瑞隆先生。基金會由創會董事長江丙坤先生與台灣高科技業多位元重量級人士經過充份協調溝通，以民間組織形式成立，積極與大陸相關標準單位和企業攜手合作，建立起以大中華市場為基礎之資訊產業發展平臺，基金會定名為《華聚產業共同標準推動基金會》，簡稱華聚基金會。

取名華聚，乃引用「華人」和「聚合」之概念，意含聚合華人智慧，共創兩岸雙贏，也意味聚合兩岸資源，發揮海峽兩岸資訊產業之聚合效應，在全球舞臺上為華人爭光。基金會以凝聚產、官、學、研之力量，加強兩岸產業標準交流合作，協助兩岸企業佈局全球智慧財產權制高點，提升國際競爭力為成立宗旨。

共通標準下載地址：

中國電子工業標準化技術協會 www.cesa.cn

華聚產業共同標準推動基金會 www.sinocon.org.tw

台灣雲端物聯網產業協會

基於協助會員發展產業技術的使命，財團法人工業技術研究院（簡稱工研院）、資策會、中華電信股份有限公司與中華資訊軟體協會聯合規劃籌組「台灣雲端物聯網產業協會」(Cloud Computing Association in Taiwan)，邀請具有代表性的業者，加入協會成為會員，並成立社團法人協會。本協會宗旨為：推動台灣三大類雲端應用服務，也就是基礎結構、平臺及軟體服務(IaaS、PaaS、SaaS)，促進台灣成為雲端運算科技創新基地。發展高度軟硬體整合的雲端系統平臺，充份結合能源科技(ET)以展現具有綠能效益的特色，提升台灣成為更具競爭優勢的全球雲端設備研發製造的重鎮。協助台灣產業朝系統解決方案及軟體服務的結構轉型，奠定台灣產業雲端服務整案輸出的基礎。