Distributed Systems

Chun-Feng Liao

廖峻鋒

Department of Computer Science

National Chengchi University

Dev: 開發
Ops: 維運

Distributed Systems

DevOps: An Introduction

Chun-Feng Liao

廖峻鋒

Dept. of Computer Science

National Chengchi University

Meet

WEALTHGRID

A successful, mid-size financial company

They have competition, but it's not much to worry about.
The players are all familiar and competing within a stable and steady market.

WealthGrid is good company with a good market. Until one day…

同業轉型

‘We want to be a tech company with a banking license’ – Ralph Hamers

08 August 2017 1 min read Listen

8 August 2017

CEO Ralph Hamers has told The Banker that he wants ING to be seen as a tech company with a banking license.

ING亦名荷蘭國際集團，是一個國際金融服務私營企業，成立於1991年，由荷蘭最大的保險公司Nationale-Nederlanden，與荷蘭的第三大銀行NMB PostBank Group合併而成。

異業切入

FINANCE

Amazon could become the third-biggest US bank if it wants to: Bain study

PUBLISHED TUE, MAR 6 2018•4:23 PM EST | UPDATED WED, MAR 7 2018•9:23 AM EST

Thomas Franck
@TOMWFRANCK

SHARE

KEY POINTS

- Bain writes that Amazon's banking services could grow to more than 70 million U.S. consumer relationships over roughly five years, rivaling Wells Fargo.
- Amazon could evade more than $250,000,000 in credit card interchange fees every year if it finds a bank willing to partner.
- The Bain report finds one-quarter of those using voice assistants like Amazon's Alexa would consider using them for everyday banking.

新創競爭

STARLING BANK

full production bank was built in a year

- 2014 Founded by Anne Boden
- June 2014 Kick-off with Regulators
- September 2015 Technical prototypes

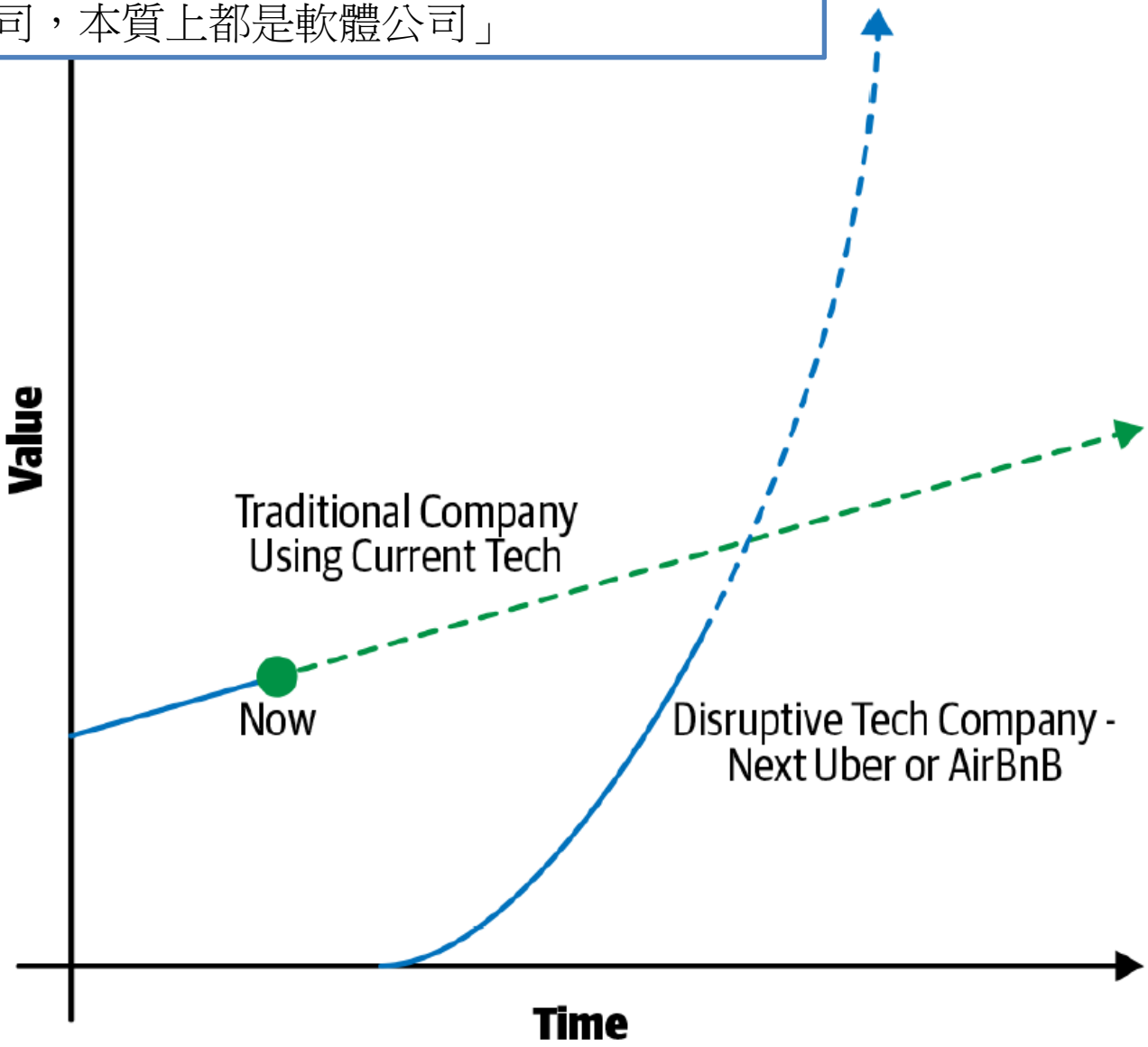
2 engineers

- January 2016 Raise $70m – start build
- July 2016 Banking licence & first account in production AWS account
- October 2016 Mastercard debit cards
- November 2016 Alpha testing mobile app
- December 2016 Direct debits live
- January 2017 Faster payments live

20 engineers

- February 2017 Launched beta testing program
- May 2017 Public App Store Launch

IT技術近10年來正急遽影響日常生動的各個層面
Marc Andreessen(2011): 軟體正在吃下全世界,未來「未來大部份公司，本質上都是軟體公司」



穩定中暗藏的危機

- Disrupt or Be Disrupted (Sharma, 2017)
  - Avg. life expectancy for fortune 500 company
    - 1960: 75 years
    - 2010: 15 years or less
- IT組織追求目標 (Kim et al., 2016)
  - 回應快速變動的競爭格局
  - 提供穩定、可靠、安全的服務
- IT組織惡性循環三部曲 (Kim et al., 2014)
  - 有限時間內，要處理問題、承諾新功能
  - 累積技術債 (受限於技術能力或時程壓力的變通處置)
  - 組織中每個人變得更忙、需要更多溝通、協調和批准

兩個似乎有點衝突，追求高效可能暗藏不穩定

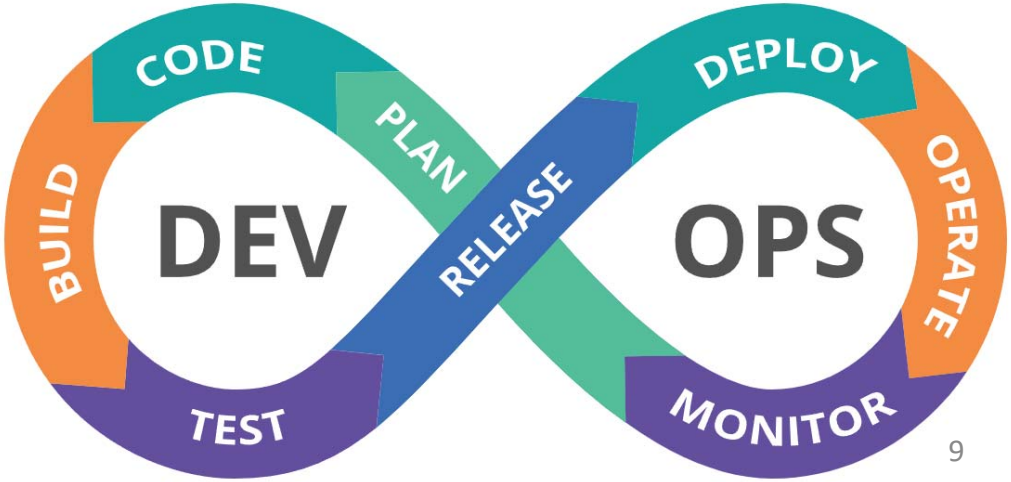
2013-2016 由Kim and Humble進行的實證研究顯示DevOps 為打破此惡性循環的重要實踐

https://www.devops-research.com/research.html

什麼是DevOps?

既快又穩定

- **定義** (CMU 軟體工程學院)
  - A set of practices, which (實踐方法)
  - reduces the time between change to production (快速回應)
  - while ensuring high quality （保持穩定、可靠、安全）
- **關注開發者Dev與維運者Ops順暢的協作模式**
  - 開發者: 程式開發、品保、測試人員
  - 維運者: SP、OP



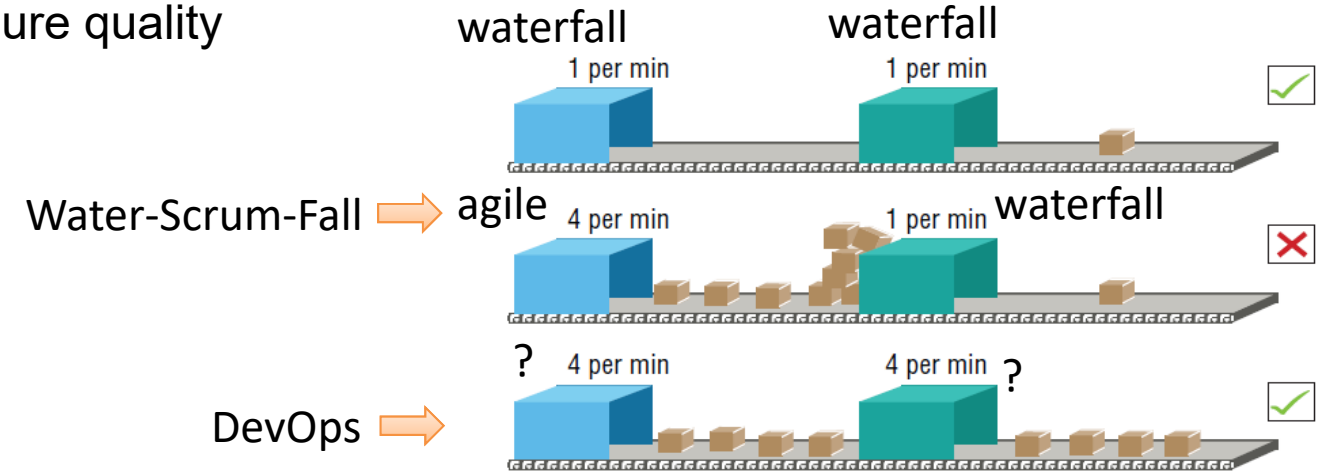
DevOps是long-term evolution guideline
不是立竿見影的特效藥

Dev vs. Ops: 為何會不順暢?

- Dev vs. Ops (Sharma, 2017)
  - Dev's task
    - Create innovation
    - Deliver ASAP
  - Ops's task
    - Provide stable, fast, and secure service
    - Ensure quality

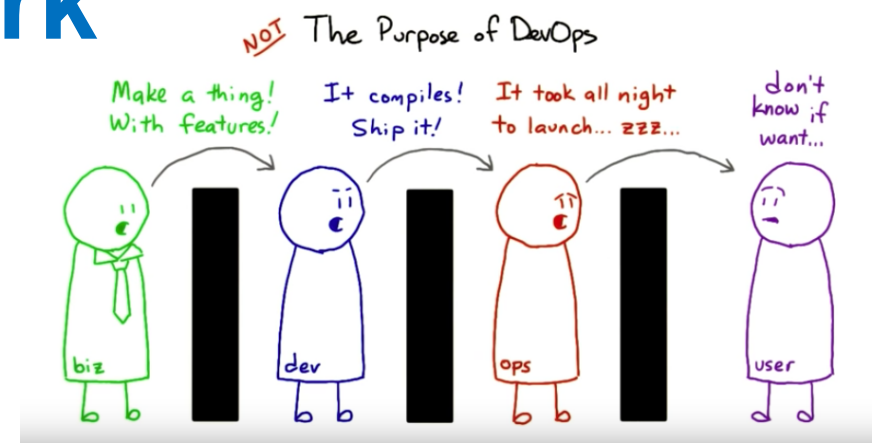
思考: 為何看似tradeoff的Fast和Quality可以並存?
工業革命教我們的事
1. 什麼東西做產品比起人來說更快?
2. 什麼東西做產品做一次和做1000次幾乎一樣品質?

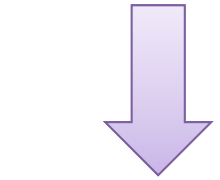
自動化



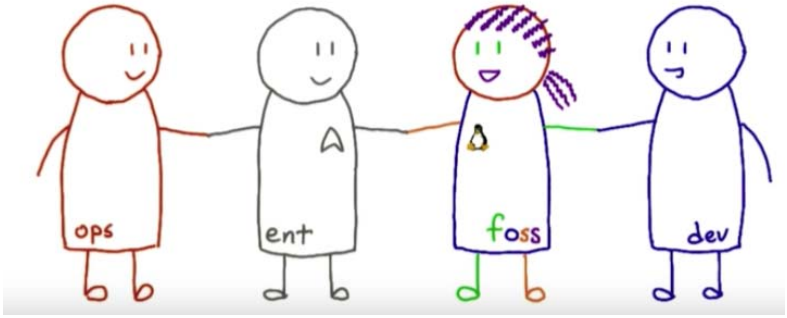
CAMS Framework

- DevOps 要領
  - Culture
  - Automation
  - Measurement
  - Sharing
- 分類
  - 技術: A, M
  - 組織: C, S





The Purpose of DevOps



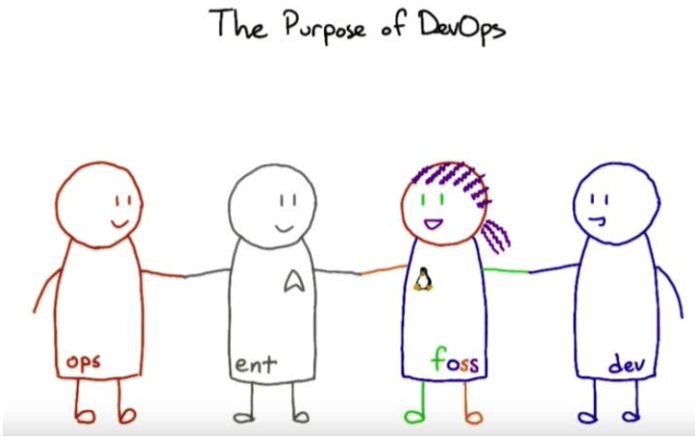
Foss = Free and open-source software

Share

- Dev
  - 向Ops學習
    - 了解平台 (部署環境)的限制
    - 以容器思維封裝程式: 讓開發/部署環境盡量保持一致 (Dev/prod parity)
    - 了解Monitor技巧 and Metrics
  - 與Ops有更好的溝通與合作
    - 與Ops合作建構CI/CD Pipeline

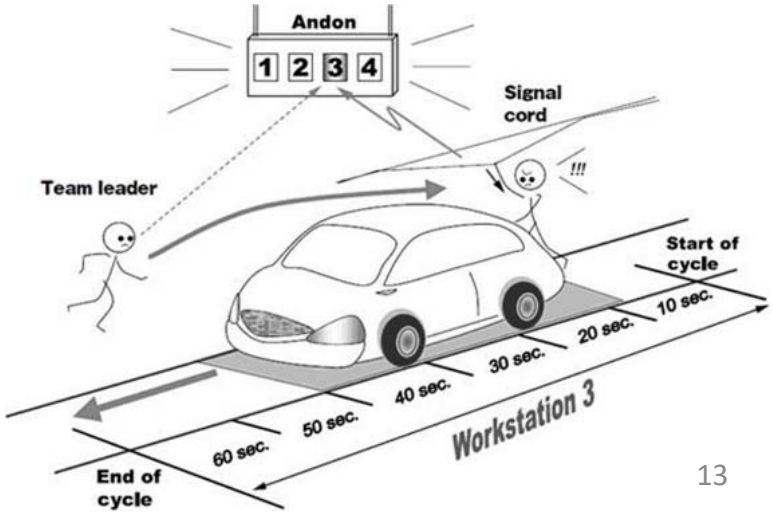
CI/CD = Continuous Integration / Continuous Delivery and Deployment

- Ops
  - 向Dev學習
    - 了解規格、了解code → 知道程式如何影響平台
  - 維運管理方式變革
    - 容器化: 以一致、獨立的方式管理、部署軟體
    - 自動化: 所有工作應編寫為程式
    - 版本化: 環境變動納入版控管理
  - 與Dev有更好的溝通與合作
    - 監控CI/CD Pipeline，即時反映錯誤



Culture

- 目的
  - 早日發現問題，在問題尚未擴大惡化就予以解決 (Shift-Left)
  - 視錯誤、問題為組織學習機會，而非究責根據
- 範例
  - Toyota Andon Cord 安燈索
    - 在生產線上，每個工作站上方都有一根繩索
    - (受訓後的)工人在特定狀況(零件出現瑕疵、損壞、所花時間過多)拉繩
    - 示警後全團隊必須立刻「共同」解決問題，如果一定期間內無法解決，則中止生產線，直到問題解決為止
  - 觀察點: 立刻解決、一起解決
    - 不是有空時再解決



回饋

- 有關Logo
  - 日常有「重視過程而非只重視結果的意義」→這個意義不錯
- 目前學校造成問題的根源與反向標題
  - 包容
    - 目前: 只任用「同溫層」、學門歧視
  - 共創
    - 目前: 資源分配、重大決策未基於一定程度共識
  - 真誠
    - 目前: 許多溝通充滿交易、權謀、算計
- 如何因應新時代的挑戰
  - 強韌性→Adaptive(適應新挑戰)→Generative (演進型)
  - 當前挑戰: 後疫情、後五年五百億(極端學術產出)
- 有關SDG

Culture and Share: 組織變革

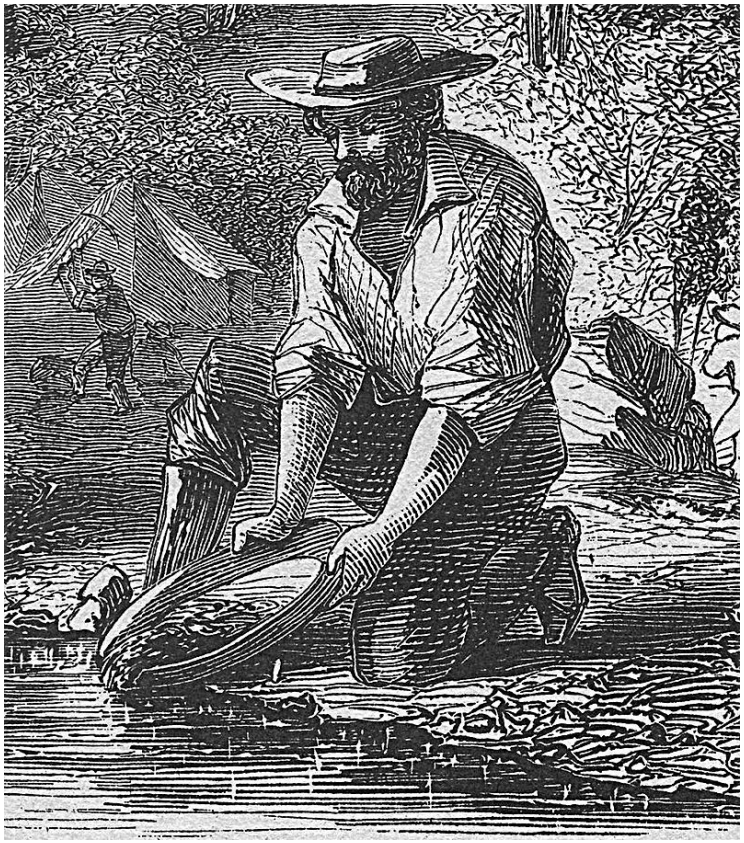
	病態型 Pathological	官僚型 Bureaucratic	演進型 Generative
資訊	Information is hidden	Information may be ignored	Information is actively sought
提出問題者	Messengers are “shot”	Messengers are tolerated	Messengers are trained
責任規屬	逃避 Responsibilities are shirked	切割 Responsibilities are compartmented	Responsibilities are shared
跨團隊合作	Bridging between teams is discouraged	Bridging between teams is allowed but discouraged	Bridging between teams is rewarded
發生失敗時	Failure is covered up	Organizatoin is just and merciful	Failure causes inquiry
新的想法	New ideas are crushed	New ideas create problems	New ideas are weclomed

Figure 8: *The Westrum organizational typology model: how organizations process information (Source: Ron Westrum, “A typology of organisation culture,” BMJ Quality & Safety 13, no. 2 (2004), doi:10.1136/qshc.2003.009522.)*


A NEW AND MAGNIFICENT CLIPPER FOR SAN FRANCISCO.
MERCHANTS' EXPRESS LINE OF CLIPPER SHIPS!
Loading none but First-Class Vessels and Regularly Dispatching the greatest number.
THE SPLENDID NEW OUT-AND-OUT CLIPPER SHIP
CALIFORNIA
HENRY BARBER, Commander, AT PIER 13 EAST RIVER.
This elegant Clipper Ship was built expressly for this trade by Samuel Hall, Esq., of East Boston, the builder of the celebrated Clippers "SURPRISE," "GAMECOCK," "JOHN GILPIN," and others. She will fully equal them in speed! Unusually prompt dispatch and a very quick trip may be relied upon. Engagements should be completed at once.
Agents in San Francisco, Messrs. DE WITT KITTLE & CO.
RANDOLPH M. COOLEY, 88 Wall Street, Tontine Building.
NESBITT & CO., PRINTERS.

1848 加州淘金熱

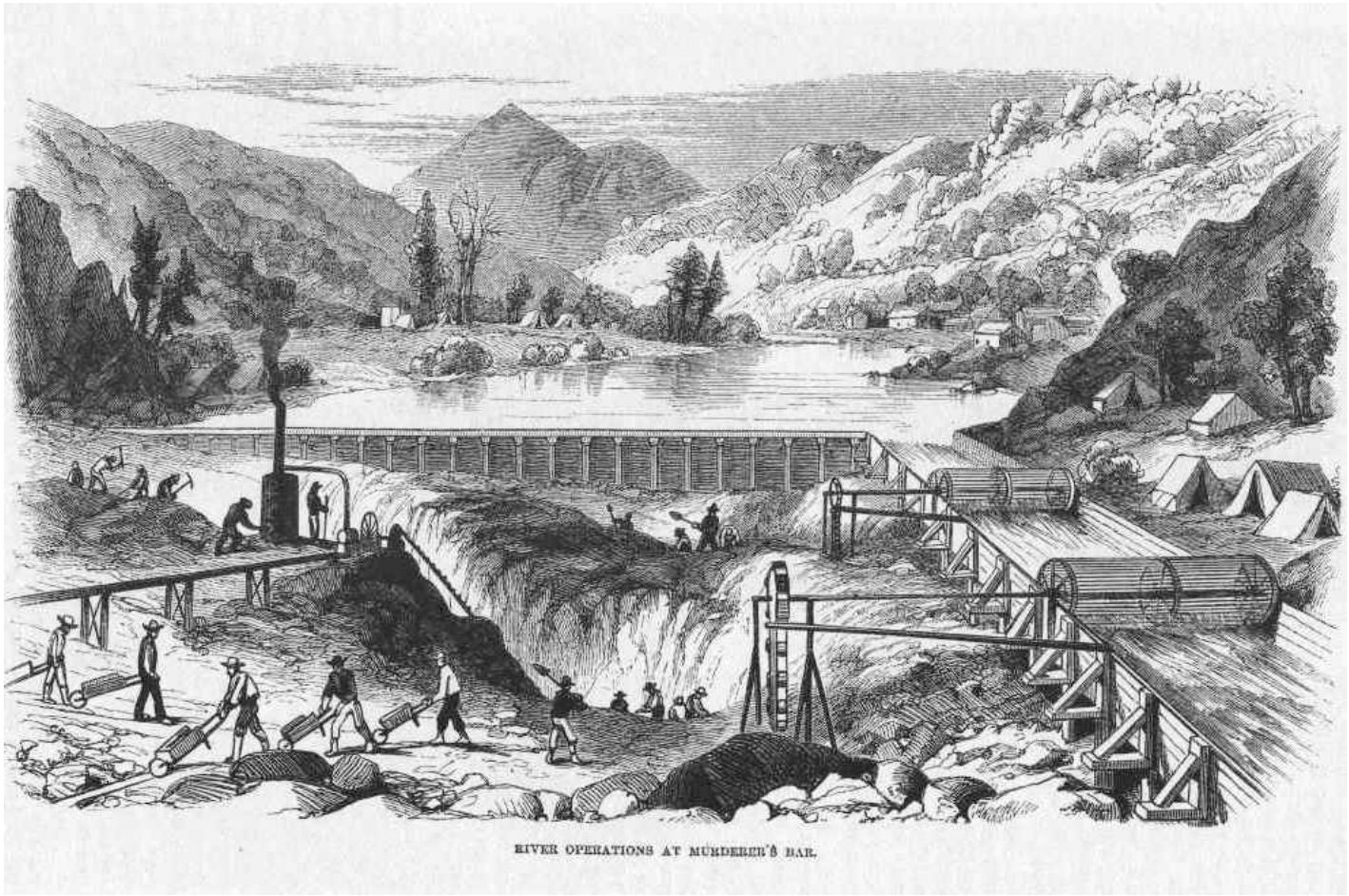
- At first, loose gold and gold nuggets could be picked up off the ground.
- Later, gold was recovered from streams and riverbeds using simple techniques, such as panning.



California, 1848

- More sophisticated methods were developed and later adopted elsewhere.

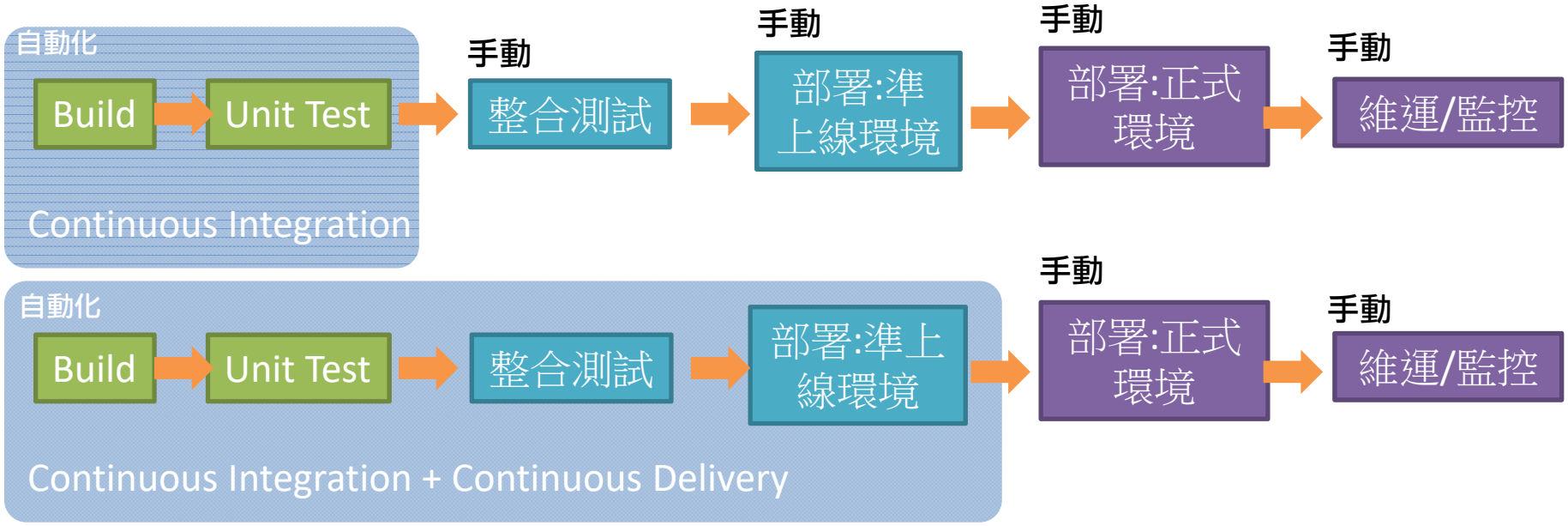
- At its peak, technological advances reached a point where significant tooling and the automated pipeline was required, and collaboration among individuals become important.



RIVER OPERATIONS AT MURDERER'S BAR.

Automation and Measurement: the CI/CD Pipeline

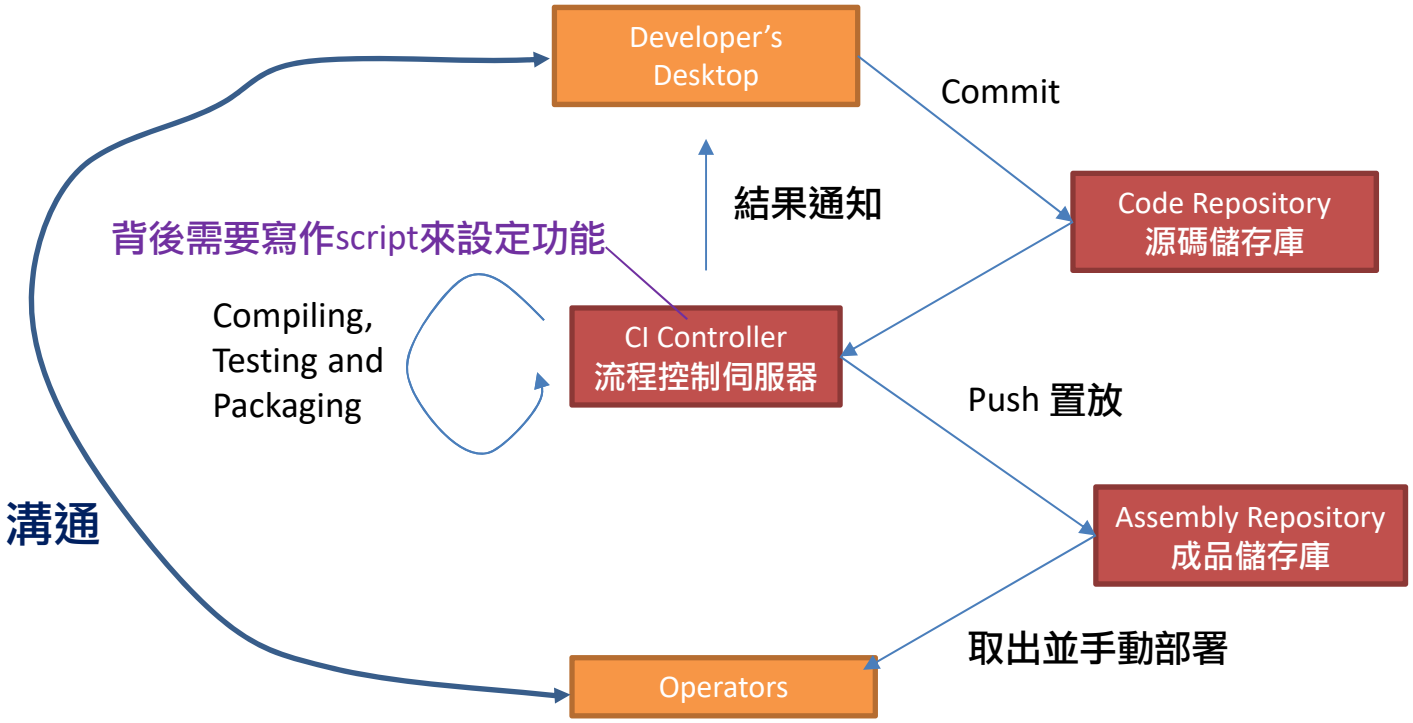
CI = Continuous Integration 持續整合
CD = Continuous Delivery / Continuous Deployment 持續交付/持續部署



Continuous Integration

- 定義
  - frequently integrating new or changed code with the existing code repository
  - merging all working copies to a shared mainline regularly
- 代價
  - 編寫自動化測試碼
  - 維護額外設施 (CI Server)
- 利益
  - 整合時不易出現重大錯誤
  - 開發人員能專心於開發/除錯 (開發到一半被叫去修大錯誤)

The CI Pipeline



紅色 Dev 自動化

橘色 Ops 沒自動化

Continuous Delivery/ Deployment

- 定義
  - 藉由自動化讓軟體產品的產出過程在短週期內完成
  - 讓軟體可持續的保持在隨時可以釋出的狀況

每個版本都可以獨立運行
越後方的版本，功能越多

- 和CI的差異
  - 涉及多個團隊之間的合作（開發、運維、QA、管理部門等）

RD => 研發

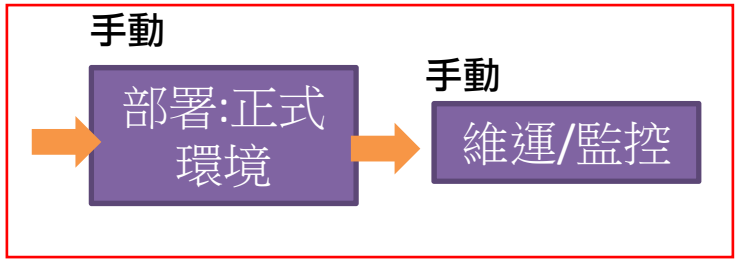
MIS => 維運

維運工作也需要自動化

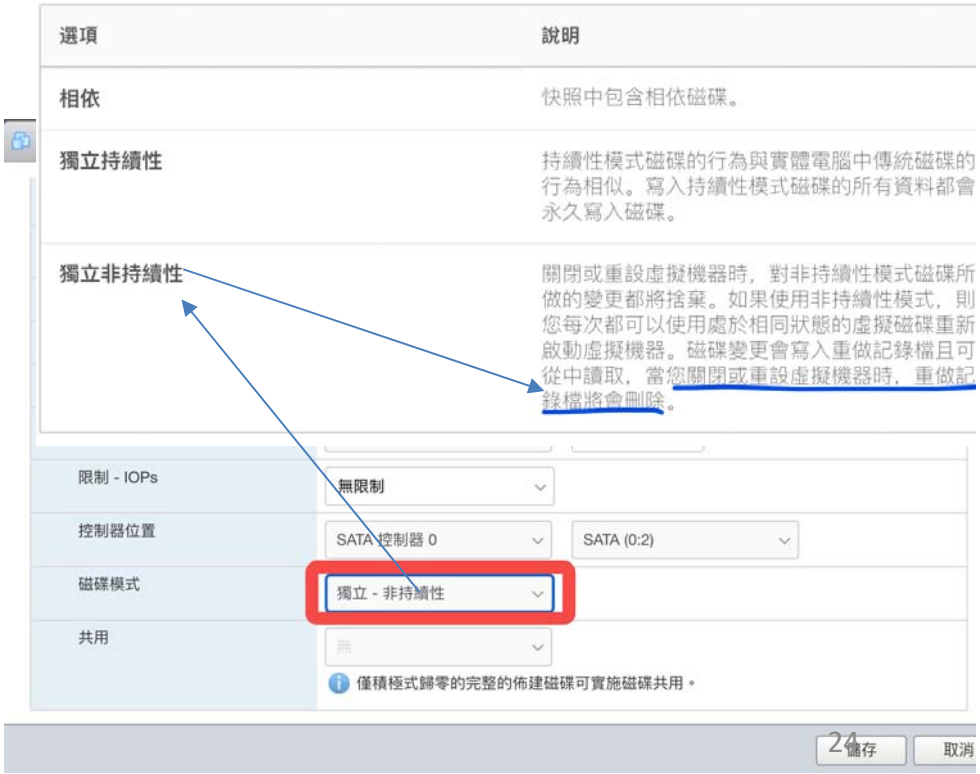
只因按錯鍵！ 2.5萬筆學習歷程檔案遺失 教長要究責

2021-09-26 11:42 聯合報 / 記者潘乃欣／台北即時報導

維運時期也需要自動化降低人為錯誤!



- 系統搬移到新機房運作十餘天之後，因系統更新而重新開機
- 其中有三台硬碟模式設定錯誤，系統更新重新開機後，硬碟被還原
- 沒有每日備份/異地備份

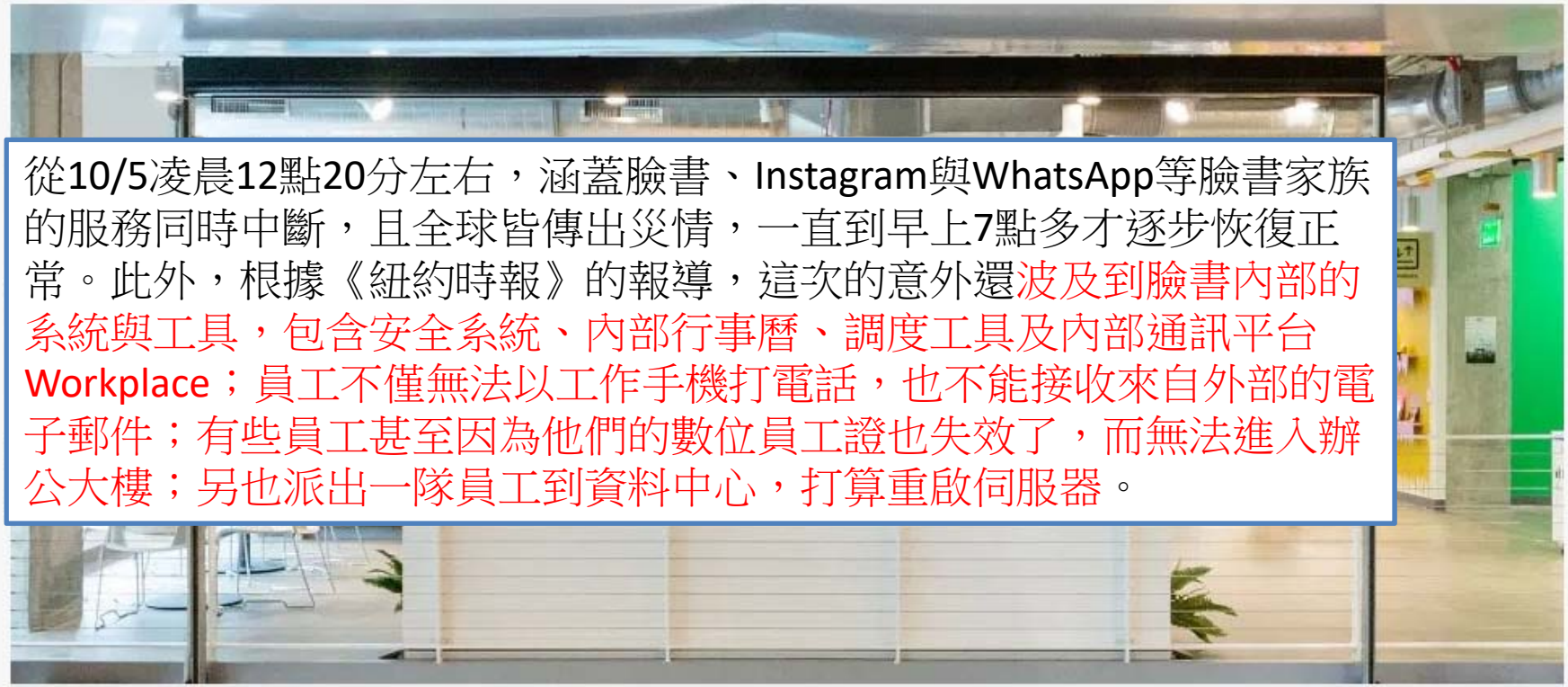


臉書公布肇事原因：配置錯誤造成大當機

臉書工程團隊在追查之後發現，用來協調資料中心之間網路流量的骨幹路由器的配置變更，是造成通訊中斷的主因，它帶來了連鎖效應，波及了臉書的各項服務

文/ 陳曉莉 | 2021-10-05 發表

讚 6.7 萬 按讚加入iThome粉絲團 讚 132 分享



從10/5凌晨12點20分左右，涵蓋臉書、Instagram與WhatsApp等臉書家族的服務同時中斷，且全球皆傳出災情，一直到早上7點多才逐步恢復正常。此外，根據《紐約時報》的報導，這次的意外還波及到臉書內部的系統與工具，包含安全系統、內部行事曆、調度工具及內部通訊平台Workplace；員工不僅無法以工作手機打電話，也不能接收來自外部的電子郵件；有些員工甚至因為他們的數位員工證也失效了，而無法進入辦公大樓；另也派出一隊員工到資料中心，打算重啟伺服器。

臉書坦承用來協調資料中心之間網路流量的骨幹路由器的配置變更錯誤，不僅影響Facebook、Instagram與WhatsApp等臉書服務，還牽連到臉書內部工具與系統，增加了診斷及解決問題的難度。（圖片來源／臉書）

華視誤播「新北市遭共軍導彈擊中」 調查局通知導播、字幕人員說明

新頭殼newtalk | 張柏源 綜合報導
發布 2022.04.21 | 00:11

華視新聞台今早跑馬字幕誤植訊息播出，造成民眾恐慌。 圖：翻攝自華視新聞台

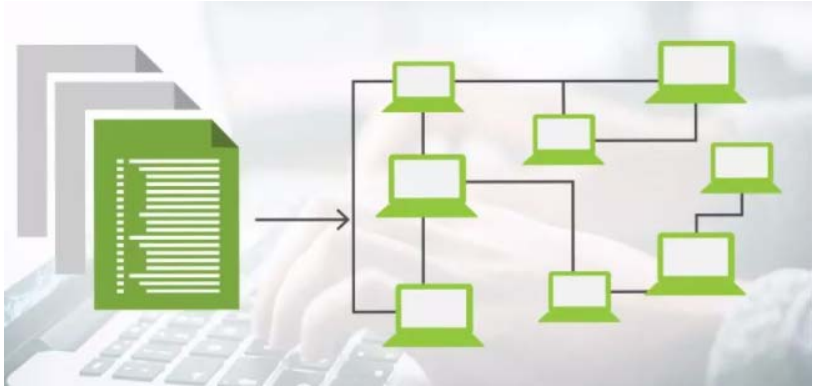
國安單位也立刻追查原因，華視上午公開澄清致歉時，台北市調查局已派調查官到華視了解系統操作流程，NCC也派人到場了解。華視總經理陳雅琳當晚8點45分左右以視訊方式突然現身在鏡頭前為本次的錯誤報導道歉，華視主播也表示，未來電視台將會盡全力配合NCC調查。

台北市調查處20日通知導播、字幕人員協助，了解快訊製作流程，釐清是否有無散布假訊息的故意，詢問後即請回。台北處調查官查明，19日當班導播受託剪接好影片並做好字幕，沒交接給20日值班的導播，而後者上班後，也沒依照華視內部流程的SOP重複確認一遍，以致啟用跑馬燈系統時，連結錯誤的字幕路徑，引發一場虛驚。

兩名華視導播在20日下午主動到台北處再度說明原委，承認2人都有疏忽，但絕非有特定目的或受人指使。台北處認為華視新聞跑馬燈等系統使用封閉的內部網路環境，暫無駭客入侵跡象，兩名導播的說明也與實際狀況相符，初步研判本次事件為不該發生卻發生的單純失誤，後續若有必要，將報請檢察官指揮偵辦。

IaC: Infrastructure as Code

- 定義
  - 將IT日常維運工作，寫成程式，以自動化腳本(as Code)方式運行
  - 基於既有軟體工程品保原則來管理自動化腳本
  - 所有工作化成自動化腳本→ 管理好腳本=做好維運工作
- 好處
  - 降低負擔
    - 自動化提高了效率
  - 降低風險
    - 每個改動都經品保與版控
    - 維運工作的可重現性大幅提高
    - 自動化→不易出錯→品質提高
  - 維運工作內容具體化
    - 不再「只有一個人會」，輪替、交接變容易



維運工作: 安裝、更新、配置、遷移、監控、修復

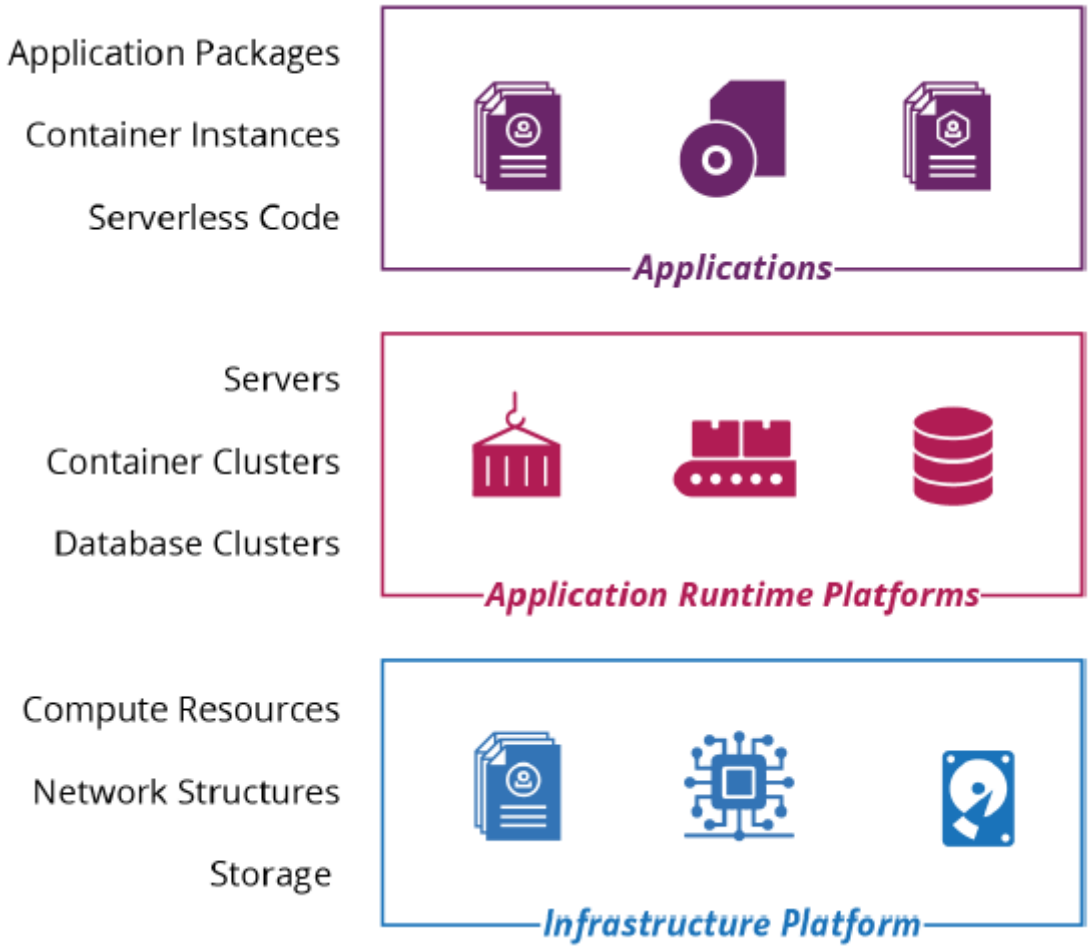
Infrastructure as Code

- Principles
  - Assume systems are unreliable
    - Create disposable things
    - Cattle, Not Pets
      - “CERN Data Centre Evolution” by Gavin McCance
  - Make everything reproducible
    - Ensure that you can repeat any process
    - Pitfall: Snowflake systems
      - 系統管理時，經常為了偶發事件一次調一點設定
        - » 如: 加裝一些模組在OS中、調整一些底層參數
      - 最後管理員本身也不敢輕易更動設定
      - 系統很難重建
  - Minimize variation
    - “管理100台配置相同的server比管理5台配置完全不同的server容易”

Infrastructure as Code

- 主軸
  - Stability comes from making changes?
  - Stability comes from making “disciplined” changes
- Core practices
  - Define Everything as Code
    - Benefits: Reusable, consistent, and transparent
  - Continuously Test and Deliver All Work in Progress (WIPs)
    - Make sure all WIPs are production ready
  - Build Small, Simple Pieces That You Can Change Independently
    - The code should be modularized

那些配置可以As Code?



Server Configuration Code Tools

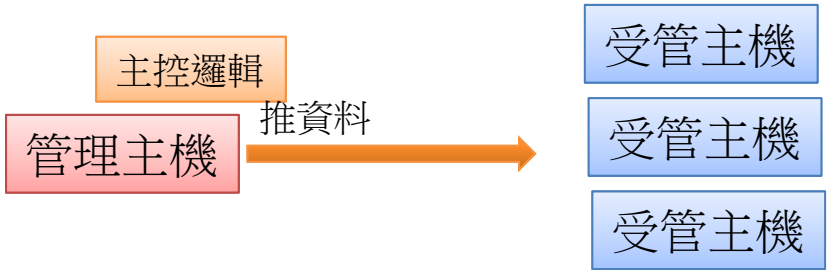
- Tools
  - Ansible
  - CFEngine
  - Chef
  - Puppet
  - Saltstack
- Configuration management
  - Ansible Tower (提供UI, 儲存playbooks)
  - Chef Server (儲存config files)

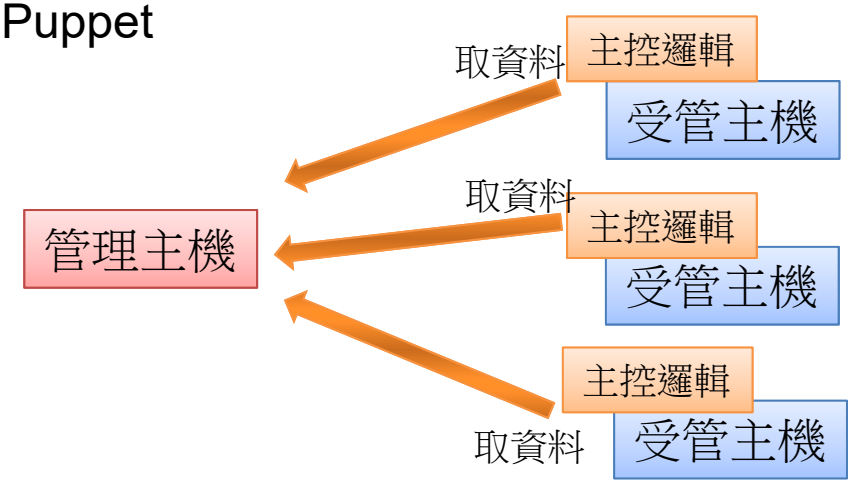
Define Everything as Code

- Two approaches
  - Declarative (functional, “what”)
    - Defining “What”
    - Typically idempotent
    - Ex: Ansible, Terraform, CFEngine
  - Imperative (procedural, “how”)
    - Defining “How”
    - Ex: Ansible, Chef, Puppet

Define Everything as Code

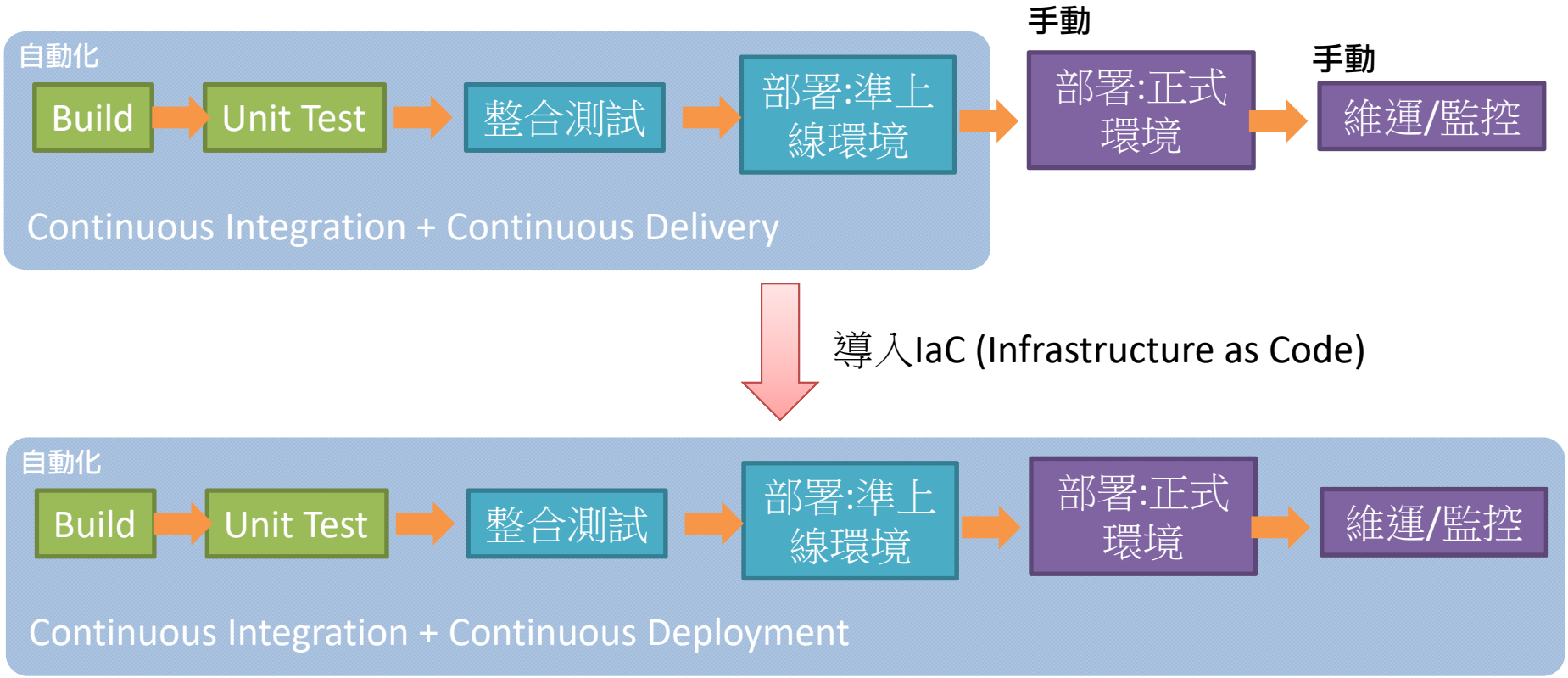
- Two operation modes
  - Push
    - The config scripts are pushed from config server to the target
    - Ex: Ansible、Terraform
  - Pull
    - The config scripts are pulled from config server to the target
    - Chef, Puppet

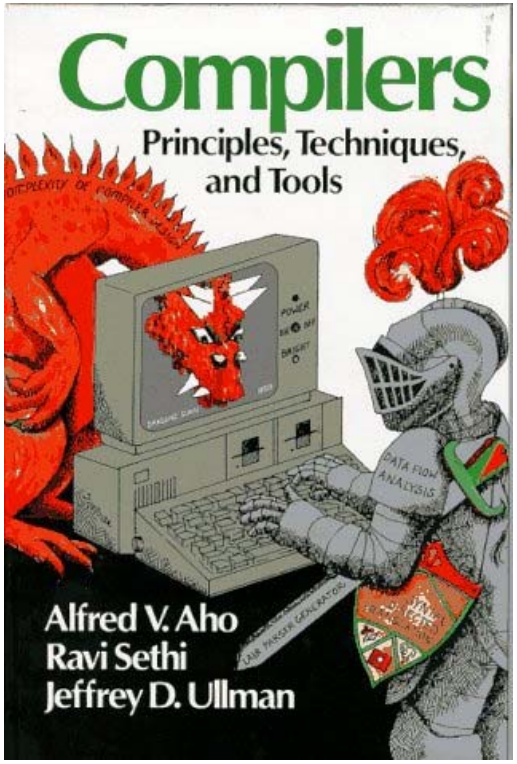


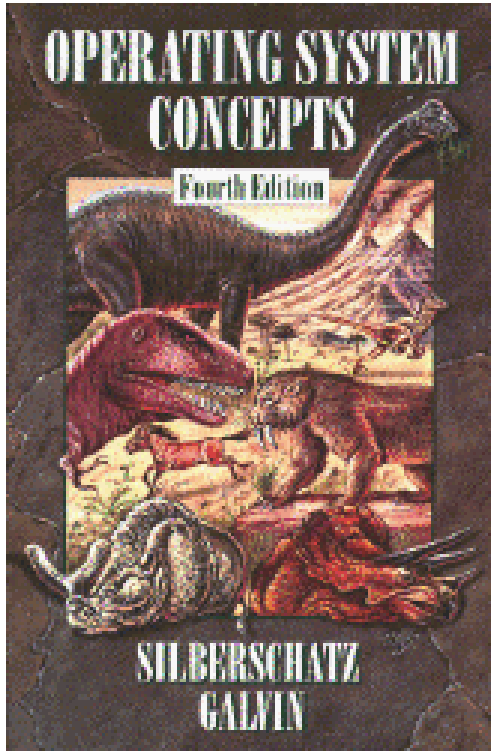


願景: 軟體產線全面自動化

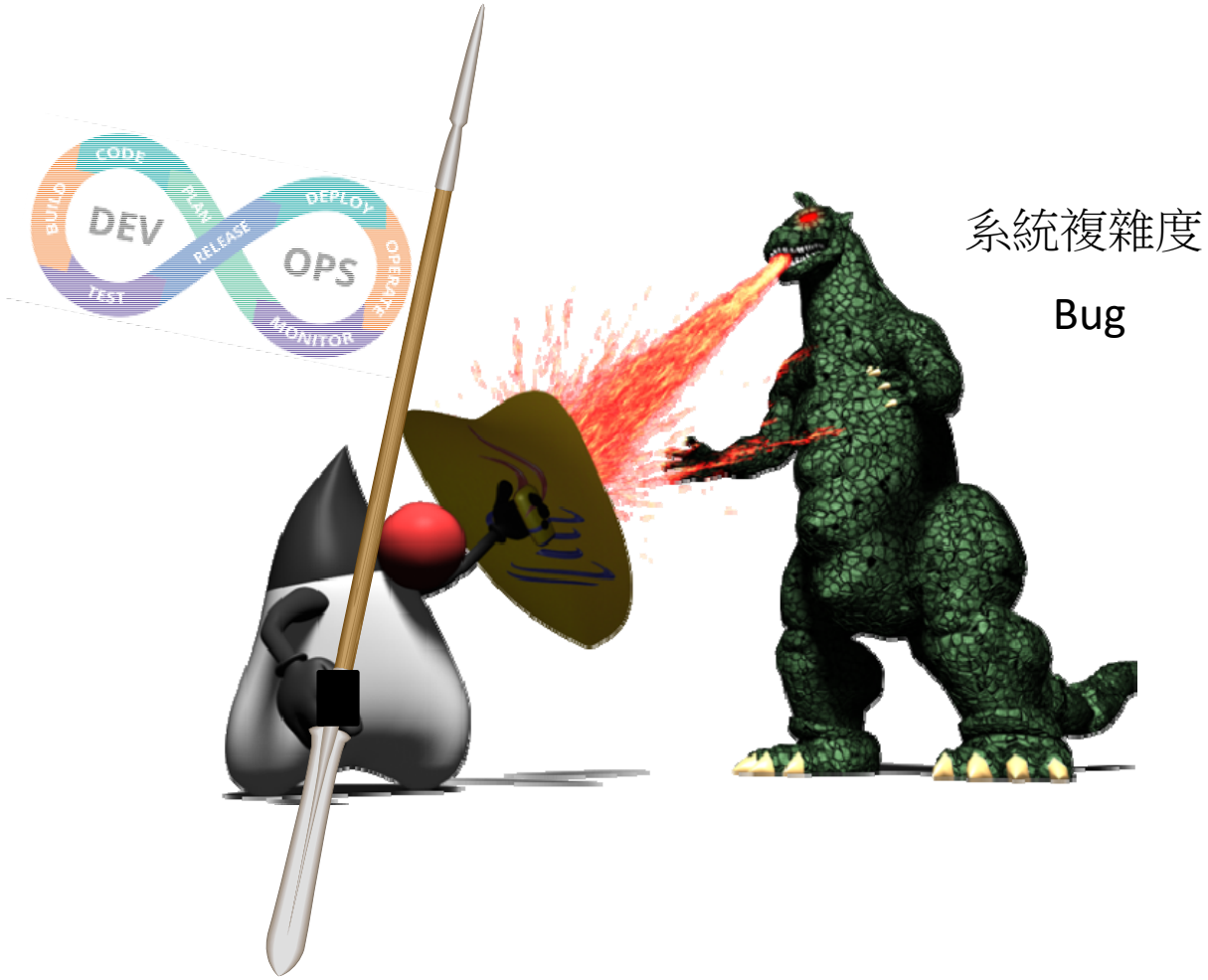
CI = Continuous Integration 持續整合
CD = Continuous Delivery / Continuous Deployment 持續交付/持續部署





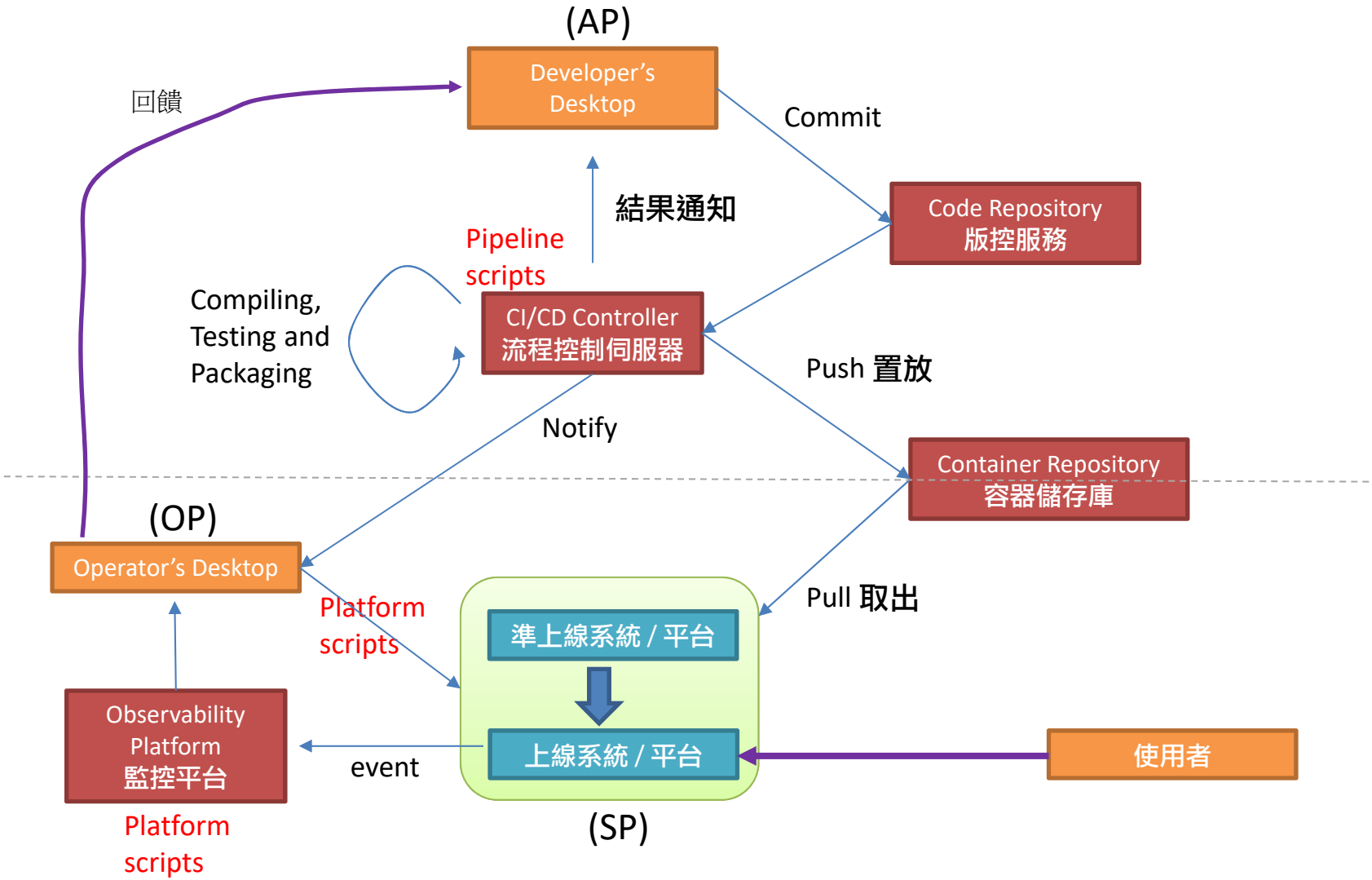


資訊系教科書顯示：開發大型軟體和跟大型怪獸打架一樣難！



DevOps概念與技術幫助團隊在與軟體怪獸纏鬥同時能快速因應各項挑戰

Full CI/CD Pipeline



導入DevOps: 三步工作法

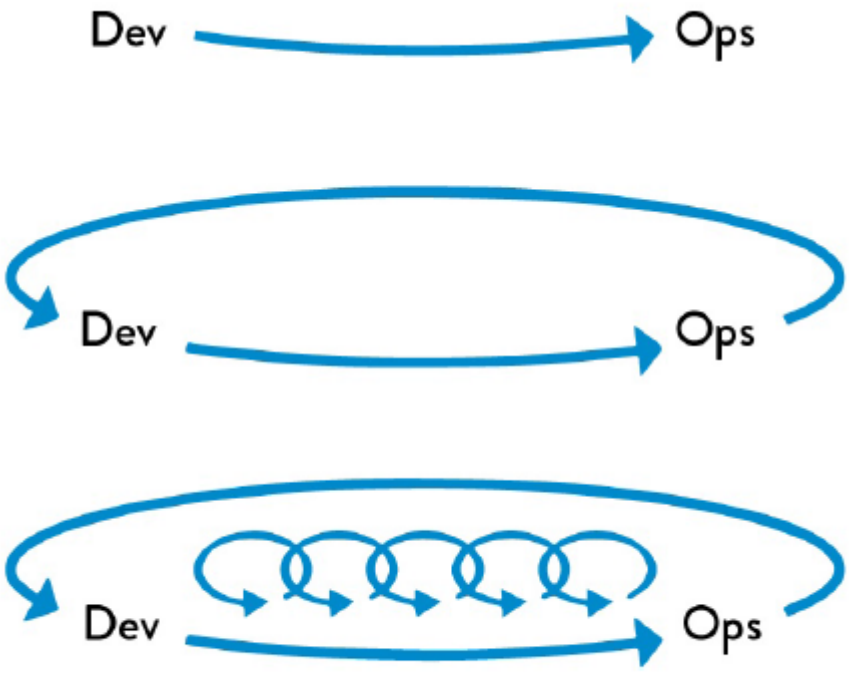


Figure 5: *The Three Ways (Source: Gene Kim, "The Three Ways: The Principles Underpinning DevOps," IT Revolution Press blog, accessed August 9, 2016, http://itrevolution.com/the-three-ways-principles-underpinning-devops/.)*

第一步: 建立Dev→Ops順暢的工作流程

第二步: 建立回饋改善機制、縮短檢測週期、快速修復

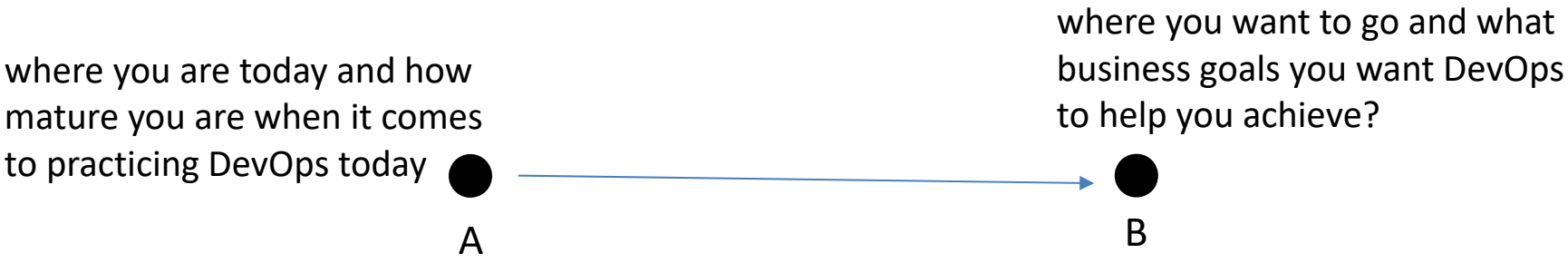
第三步: 建立Dev/Ops互相信任、培育勇於嘗試但以科學方法嚴謹驗證的企業文化。從成功學習，也從失敗中學習。

四個改善領域

- Process Improvement
  - How to make the processes lean and efficient
- Tools for Automation
  - How to automate processes with tools
- Platform and Environments
  - How to make platforms and environments for the delivery pipeline resilient, elastic, scalable, and able to manage configurations
- Culture
  - How to foster a culture of trust, communication, and collaboration

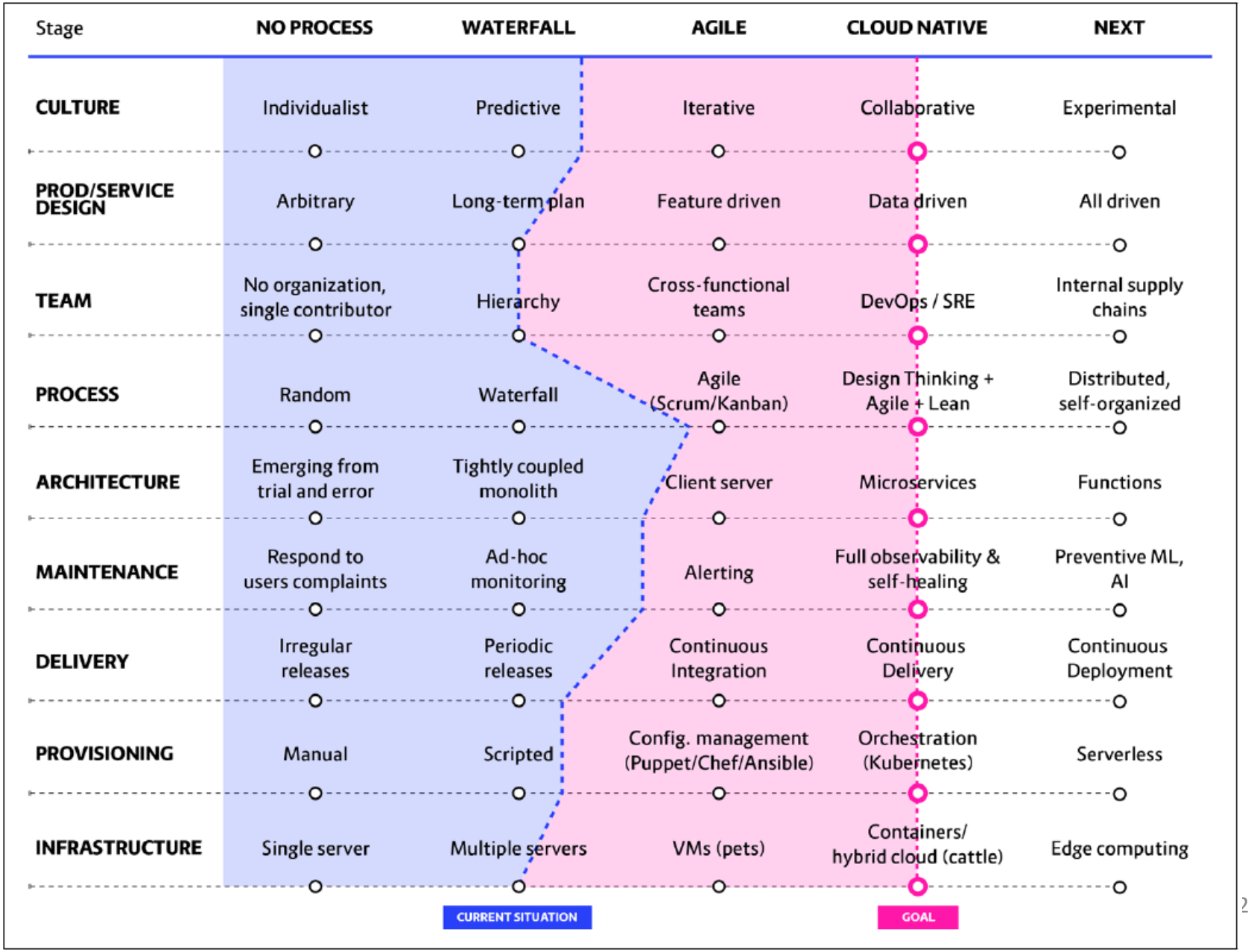
Adopting DevOps

- DevOps is about adopting a philosophy
  - Principle and practices that affect people, processes, and tools
- Adopting DevOps
  - A journey; not a one-and-done project
  - A mindset and culture; not a product or a process
  - A long-term transformation
- Where do I start?



資料來源: The DevOps Adoption Playbook by Sanjeev Sharma (IBM Distinguished Engineer)

Cloud Native 成熟度量表



第一步: 建立Dev到Ops順暢的工作流程

- 限制(每個人)WIP數量
  - WIP=Work in Progress
    - 每個人「同時」負責的進行中專案/任務
  - 為什麼要限制WIP
    - 製造業中生產線中斷是嚴重事故
    - 腦力密集的工作者經常被中斷
      - 表面上嚴重性不如製造業嚴重，其實影響更大
      - Context-Switch: 必須在多項任務、認知規則、不同目標來回切換
      - Thrashing: 大部份時間都花在重新進入(回想) Context
    - 心理學研究: 即使是分類幾何圖形的簡單任務，如果一直被打斷，人的效率也有顯著降低

第一步: 建立Dev到Ops順暢的工作流程

在腦力密集工作中，即使都是同類工作(如coding)，工作內函也有顯著特異性

- 數字代表專案編號
- F, I, Se, St代表不同專案的同類工作

第一步: 建立Dev到Ops順暢的工作流程

- 減少非必要handoffs
  - Handoffs: 工作在不同單位/人之間交接移轉
  - 例: 修改網頁上打錯的名字，簽呈需要蓋20個章

第一步: 建立Dev到Ops順暢的工作流程

- 約束點理論 (瓶頸)
  - Dr. Goldratt (Beyond the Goal)提出的觀察
  - 在流程中經常成為瓶頸的工作
    - 新人到職的開發環境佈建
    - 軟體版本更新(程式碼部署)
    - 大專案的測試準備
  - 解決流程
    - 找出工作流程中的約束點
    - 思考如何改善約束點
    - 將改善上述約束點列為優先工作
    - 著手改善約束點

第二步: 建立回饋改善機制

- 目的
  - Shift-Left: 早日發現問題，在問題尚未擴大惡化就予以解決
  - 文化: 視錯誤、問題為組織學習機會，而非究責根據
- 範例
  - Toyota Andon Cord安燈索
    - 在生產線上，每個工作站上方都有一根繩索
    - (受訓後的)工人在特定狀況(零件出現瑕疵、損壞、所花時間過多)拉繩
    - 示警後全團隊必須立刻「共同」解決問題，如果一定期間內無法解決，則中止生產線，直到問題解決為止
  - 觀察點: 立刻解決、一起解決
    - 不是有空時再解決

第二步: 建立回饋改善機制

- 投入20%的時間投入(與可測得KPI無關)的面相
  - 「看不見的部份」通常有持續而深遠的影響

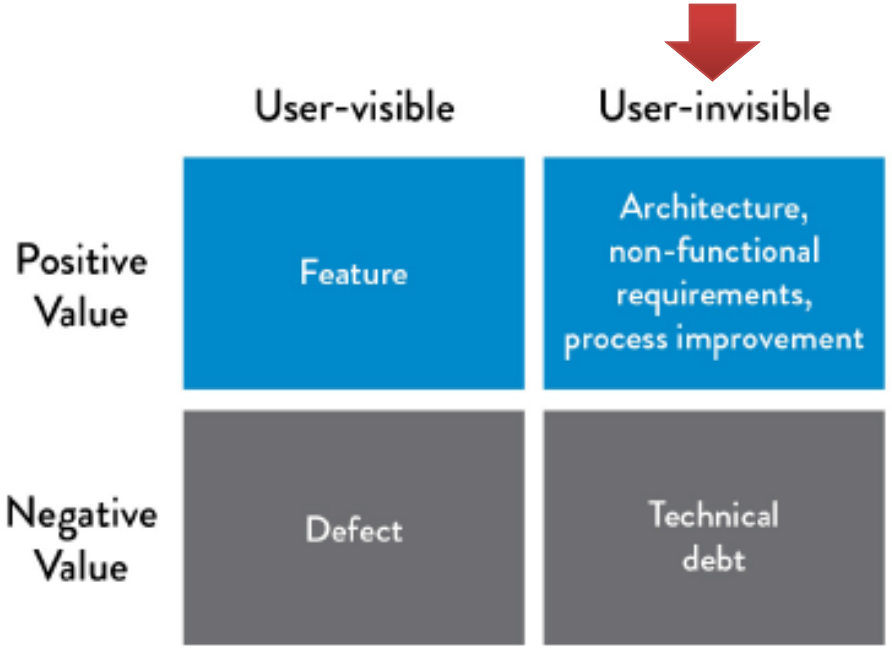


Figure 11: *Invest 20% of cycles on those that create positive, user-invisible value (Source: "Machine Learning and Technical Debt with D. Sculley,"* Software Engineering Daily *podcast, November 17, 2015, http://softwareengineeringdaily.com/2015/11/17/machine-learning-and-technical-debt-with-d-sculley/.)*

第三步: 持續學習型組織

- Ron Westrum: 組織文化對於「安全與績效」有影響
  - 特別是「腦力密集」的產業
- 三種組織類型

	病態型 Pathological	官僚型 Bureaucratic	進化型 Generative
	Information is hidden	Information may be ignored	Information is actively sought
提出問題者	Messengers are "shot"	Messengers are tolerated	Messengers are trained
責任規屬	逃避 Responsibilities are shirked	切割 Responsibilities are compartmented	Responsibilities are shared
跨團隊合作	Bridging between teams is discouraged	Bridging between teams is allowed but discouraged	Bridging between teams is rewarded
發生失敗時	Failure is covered up	Organizatoin is just and merciful	Failure causes inquiry
新的想法	New ideas are crushed	New ideas create problems	New ideas are weclomed

Figure 8: *The Westrum organizational typology model: how organizations process information (Source: Ron Westrum, "A typology of organisation culture," BMJ Quality & Safety 13, no. 2 (2004), doi:10.1136/qshc.2003.009522.)*

第三步: 持續學習型組織

- 領導人的角色

However, there is significant evidence that shows greatness is not achieved by leaders making all the right decisions—instead, the leader's role is to create the conditions so their team can discover greatness in their daily work. In other words, creating greatness requires both leaders and workers, each of whom are mutually dependent upon each other.

從那裡切入

- 綠地專案 vs. 棕地專案
  - 綠地專案: 全新開啟(或尚在規劃)的軟體專案
    - 有機會建構全新應用與基礎架構
    - 不需顧慮既有程式、流程與團隊，相對容易開展
    - 公有雲、私有雲測試、採用新工具的專案
    - 例: National Instruments: Hosted LabVIEW
      - 團隊: 2個架構師、2位開發人員、1個自動化開發人員、3個行政管理人員
  - 棕地專案: 服務長達數年、數十年的軟體系統
    - 通常背負大量技術債
    - 導入風險高、相對困難，但成功後通常帶來巨大商業價值
    - 時機: 系統陷入困境，專案團隊認為傳統方法已無法適用

從那裡切入

- 從包含最多「創新者」的團隊開始
- MIT管理學教授R.Fernandez博士: 如何推動創新改革
  - 發現創新者和早期採用者
  - 建立Critical Mass
    - 考量Silent majority
    - 謹慎避免和反對者衝突
  - 辨識不為所動者
    - 高調且具備影響力的反對者
    - 專案足夠穩固時才處理

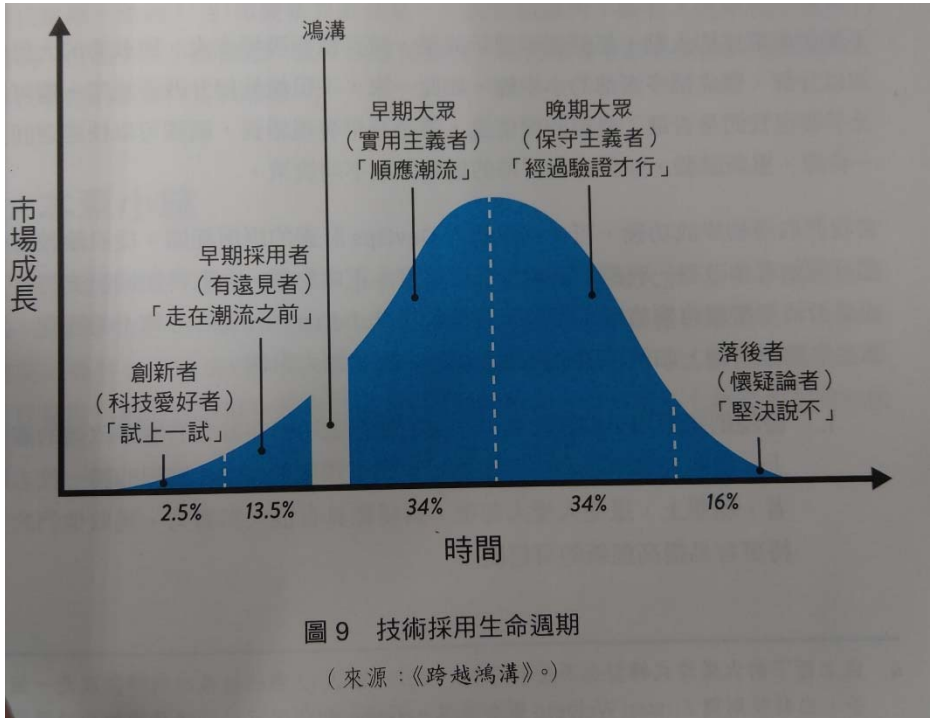


圖 9　技術採用生命週期

（來源：《跨越鴻溝》。）

DevOps與組織慣性(inertia)

- DevOps, at its heart, is a cultural movement
  - 組織成功轉型為DevOps文化的方法，是克服「組織慣性」
    - 定義: An inherent resistance to change
    - Individuals may appreciate the value of adopting DevOps
    - But, as a collective, they resist change and thus have inertia

“This is the way we do things here.”
“Yes, but changing X is not in my control.”
“Nothing is broken in our processes. Why should we change?”
“You will need to talk to Y about that; WE cannot change how THEY work.”
“Management will never allow that.”
“Don’t you know we are in a regulated industry?”
“DevOps is the new flavor of the month. Let’s see how long this effort lasts.”

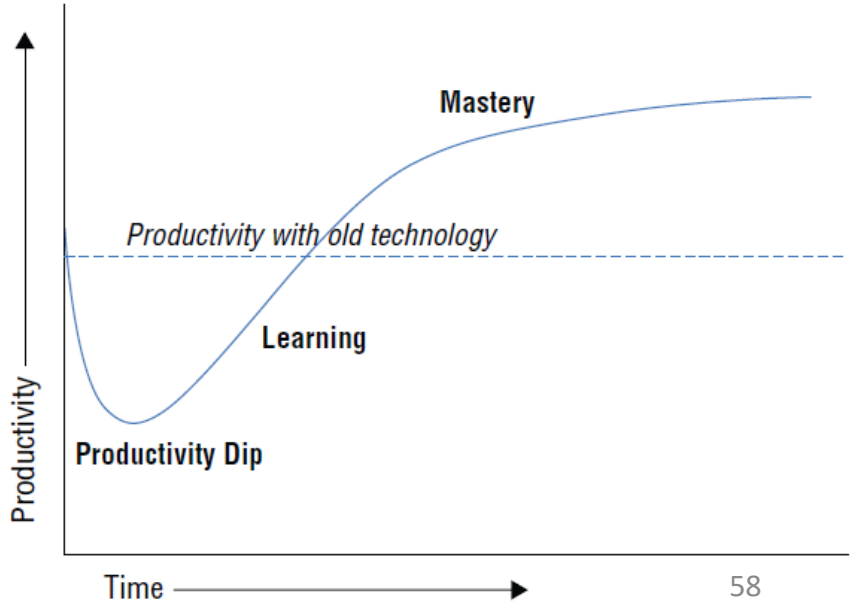
資料來源: The DevOps Adoption Playbook by Sanjeev Sharma (IBM Distinguished Engineer)

DevOps與組織慣性(inertia)

- Inertia產生主因
  - Over time, organizations develop behaviors
    - Process and rules exist, but no one knows why
    - They are “just there”
- 對策
  - Visibility: 資訊(好壞消息)透明、增加互信
  - Effective communication: 減少層層轉達; 增加直接溝通
  - Common measurements: 修訂績效評估標準
    - People will not change their behaviors, unless the way they are being measured matches the new, desired behaviors

Minimizing the Dip

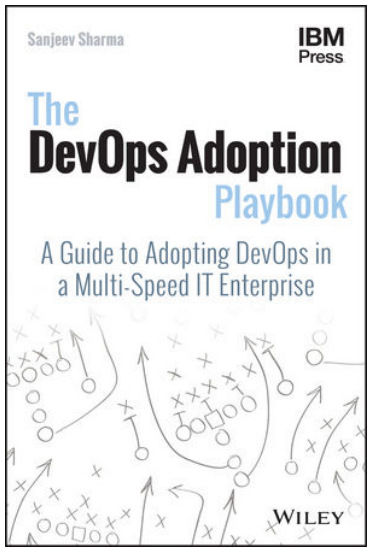
- Dip
  - Introducing any change results in an immediate drop in productivity
  - It is unavoidable
  - A good transformation plans can ensure it is minimized
- Possible solutions
  - DevOps Coach
  - Measurement
    - KPI before and after the change



How to adopt

- Prepare
  - Working on top-down and bottom-up sponsorship and engagement to overcome cultural inertia
- Steps
  - Defining the target state
  - Understanding the current state
  - Picking the right plays/patterns
    - to get from the current state to the target state
  - Preparing to address the dip

資料來源: The DevOps Adoption Playbook by Sanjeev Sharma (IBM Distinguished Engineer)



資料來源: The DevOps Adoption Playbook by Sanjeev Sharma (IBM Distinguished Engineer)

CNT Patterns

- https://www.cnpatterns.org/

Strategy & Risk Reduction

Organization & Culture

Development & Design

Infrastructure & Cloud

Operations

Q & A