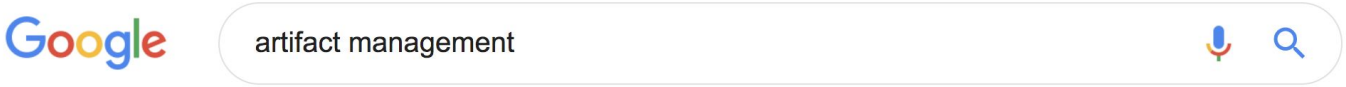
軟體交付的濫觴
談產出物管理
Artifacts Management

Rick Hwang
2019/03/28

DevOps Taiwan Meetup #22



約有 132,000,000 項結果 (搜尋時間：0.44 秒)

提示： 只顯示繁體中文搜尋結果。您可以在使用偏好中指定搜尋語言

Artifacts Management

Artifacts Management - Complete Think

https://rickhw.github.io/2018/07/08/DevOps/Artifacts-Management/

2018年7月8日 - Artifacts 是CI/CD 很關鍵的點，他承接著接下來整個Development Pipeline 的樞紐因子，能否有靈活、有彈性的佈署，都仰賴於Artifacts Management。

Learning from Open Source ... · 目的 · Artifact Types · Artifact Metadata

Why Is Artifact Management Important? | Perforce

https://www.perforce.com/blog/vcs/why-artifact-management-important ▾ 翻譯這個網頁

2018年6月21日 - An artifact repository organizes and manages binary files and their metadata in a central place. Artifact repository tools are important as teams ...

Binary repository manager - Wikipedia

https://en.wikipedia.org/wiki/Binary_repository_manager ▾ 翻譯這個網頁

跳到 Artifacts and packages - Artifacts and packages inherently mean different things. Artifacts are simply an output or collection of files (ex. JAR, WAR ...

Introduction · Universal package manager · Relationship to ... · Metadata

今天主題的特色：

1. 沒啥存在感
2. 很少人知道這是什麼
3. 大部分『持續交付』的書不會談

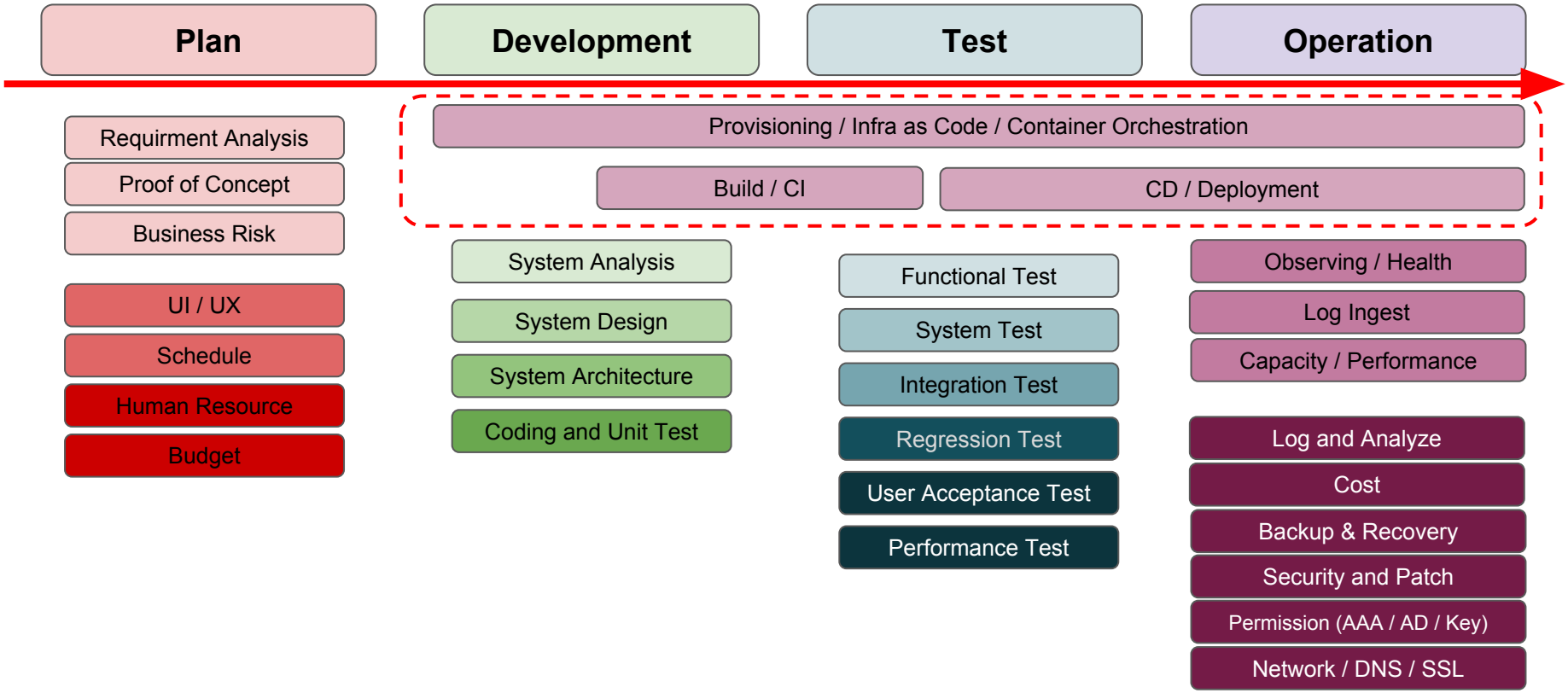
調查

- 你的公司是新創**草創期**（１～３年）？
- 你的公司是新創**成長期**（３～５年）？
- 你的公司業務有**多市場**(北美、歐洲、日本)？
- 你的工作有多個環境部署需求？

聊三件事情

一、一段經驗談
二、一個觀察
三、真實世界

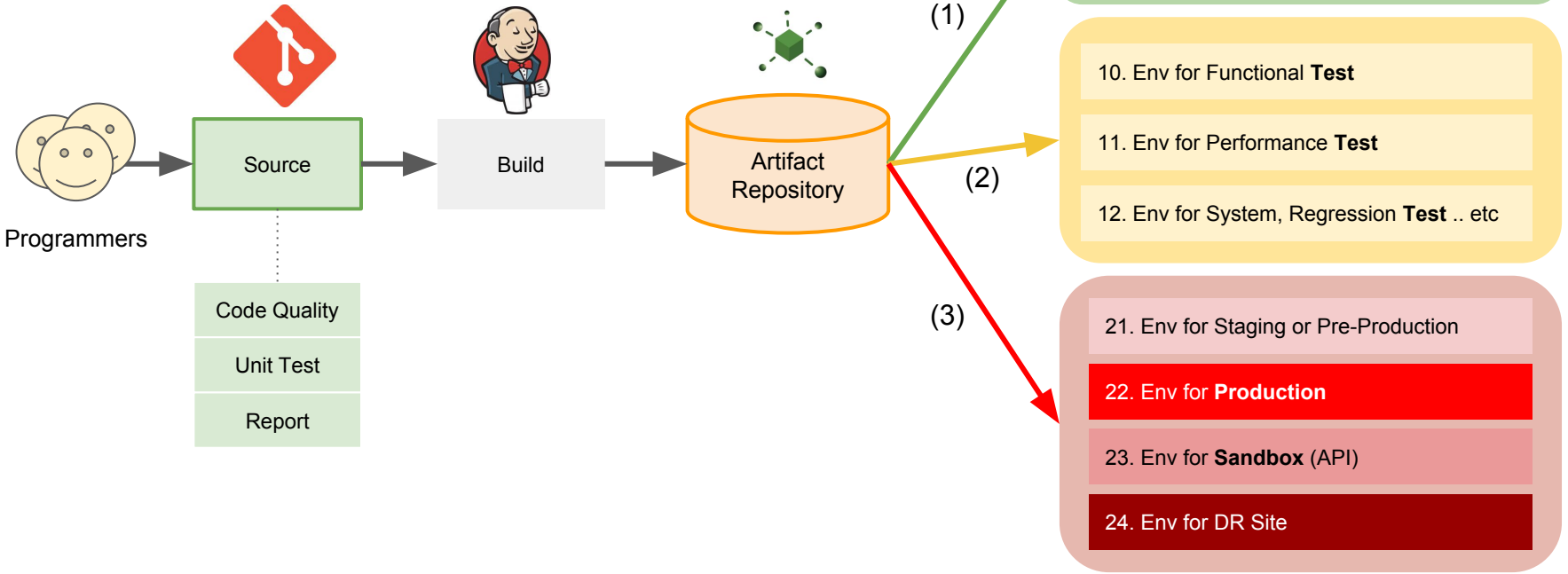
Software Development Lifecycle (SDLC)



source: [Software Development Lifecycle]

持續交付流水線

(1) 研究與開發:技術選擇與決策
(2) 探索與驗證:功能與非功能
(3) 部署策略與風險控管



Continuous Integration

Continuous Delivery

第一件事　一段經驗談 ...

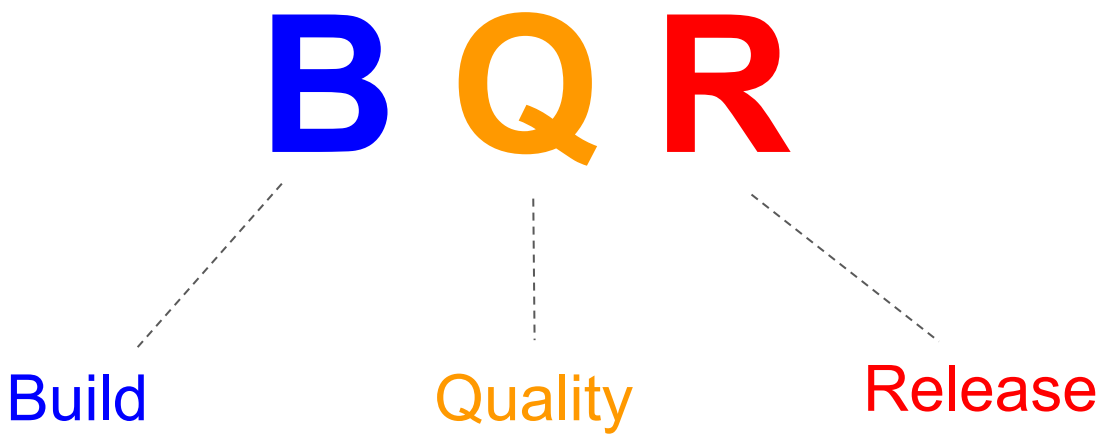
新創事業前期

- IoT 產業
- Prototype 完成一年
- Development Team: 20+
  - Cloud Services, Web: Java / Netty, Live Streaming (TW / US)
  - Mobile APP: iOS / Android / WinPhone (TW / US)
  - Devices (Embedded): IPCam, Senors (WH / ZH)
- PM / Manager: US

剛到的第一個月 ...

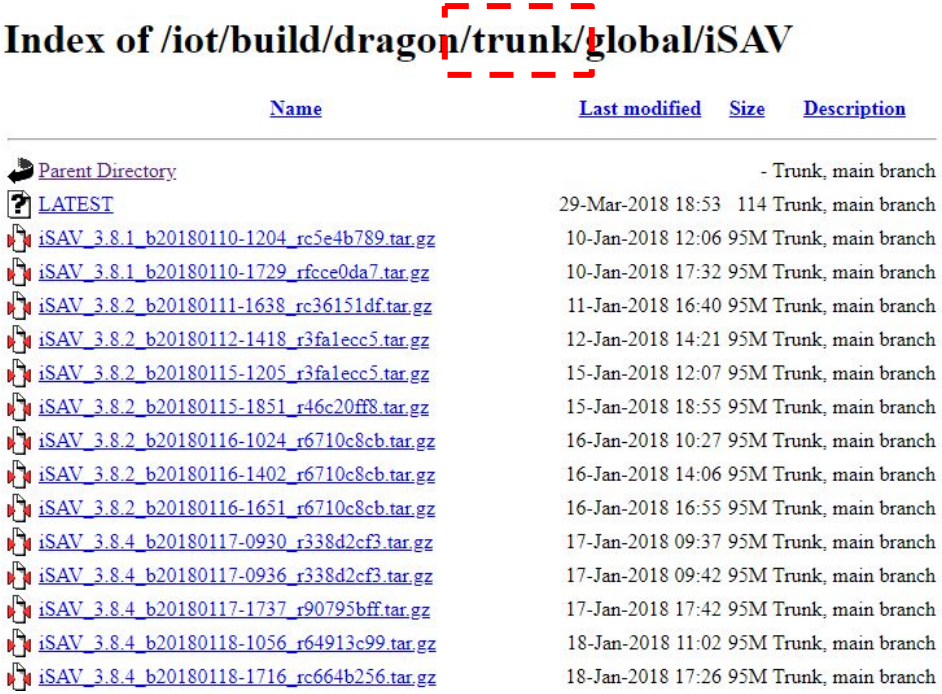
- 狀況是這樣的
  - Server Team (TW / US): RD 自己裝的, 不知道 Server 在哪
  - Mobile (TW): RD Email 給我 apk, ipa .. app.apk
  - Devices (WH): 從來不知道怎麼裝, RD 在武漢
- 了解產品全貌: 架構、產品目標、確立目標、組織與團隊 ...

快三週後 … 老闆指令來了
只有三個字



第二個月 : 標準化 - Build Process

- Semantic Versioning
  - 定義 Components 名稱
  - 定義 Artifact Meta
  - 定義目錄結構與檔名
  - 定義 Init Version 以及生命週期
  - Versioning Dependencies
- Config 標準化: 定義 endpoints
- 調整分支策略: Trunk based
- Build Strategy
  - 開發 autobuild: config, log, notification, build script
  - allocate 資源: Mac mini, Linux, Windows, Storage
  - rsync to remote sites: TW / US

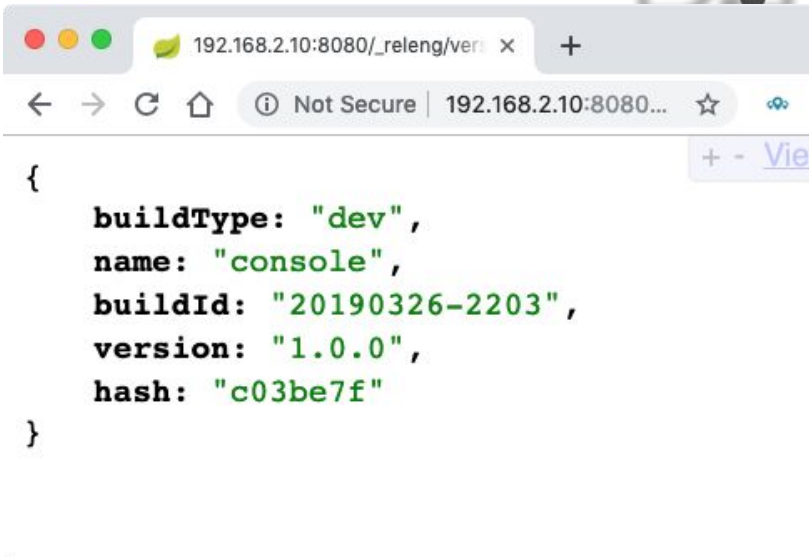


Index of /iot/build/dragon/trunk/global/iSAV

Name	Last modified	Size	Description
Parent Directory			- Trunk, main branch
LATEST	29-Mar-2018 18:53	114	Trunk, main branch
iSAV_3.8.1_b20180110-1204_rc5e4b789.tar.gz	10-Jan-2018 12:06	95M	Trunk, main branch
iSAV_3.8.1_b20180110-1729_rfcce0da7.tar.gz	10-Jan-2018 17:32	95M	Trunk, main branch
iSAV_3.8.2_b20180111-1638_rc36151df.tar.gz	11-Jan-2018 16:40	95M	Trunk, main branch
iSAV_3.8.2_b20180112-1418_r3fa1ecc5.tar.gz	12-Jan-2018 14:21	95M	Trunk, main branch
iSAV_3.8.2_b20180115-1205_r3fa1ecc5.tar.gz	15-Jan-2018 12:07	95M	Trunk, main branch
iSAV_3.8.2_b20180115-1851_r46c20ff8.tar.gz	15-Jan-2018 18:55	95M	Trunk, main branch
iSAV_3.8.2_b20180116-1024_r6710c8cb.tar.gz	16-Jan-2018 10:27	95M	Trunk, main branch
iSAV_3.8.2_b20180116-1402_r6710c8cb.tar.gz	16-Jan-2018 14:06	95M	Trunk, main branch
iSAV_3.8.2_b20180116-1651_r6710c8cb.tar.gz	16-Jan-2018 16:55	95M	Trunk, main branch
iSAV_3.8.4_b20180117-0930_r338d2cf3.tar.gz	17-Jan-2018 09:37	95M	Trunk, main branch
iSAV_3.8.4_b20180117-0936_r338d2cf3.tar.gz	17-Jan-2018 09:42	95M	Trunk, main branch
iSAV_3.8.4_b20180117-1737_r90795bff.tar.gz	17-Jan-2018 17:42	95M	Trunk, main branch
iSAV_3.8.4_b20180118-1056_r64913c99.tar.gz	18-Jan-2018 11:02	95M	Trunk, main branch
iSAV_3.8.4_b20180118-1716_rc664b256.tar.gz	18-Jan-2018 17:26	95M	Trunk, main branch

Artifact Meta (基本款)

- Name: 名稱
- Description: 描述
- Version: 版本, x.y.z, Sementic Versioning
- BuildId: 時間, YYYYmmdd-HHMM
- Hash: VCS hashcode / Serial Number
- BuildType: dev | rel



```
23:42:55 gtcafe-dev.rickhwang-dev proj.cloud.console.v2 ⑂ master  java -jar target/console_v1.0.0_b20190326-1414.jar
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/Users/rick/Repos/rickhw/gitlab/GTCafe.Navigator/nav.poc/proj.cloud.console.v2/target/conso
20190326-1414.jar!/BOOT-INF/lib/logback-classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/Users/rick/Repos/rickhw/gitlab/GTCafe.Navigator/nav.poc/proj.cloud.console.v2/target/conso
20190326-1414.jar!/BOOT-INF/lib/slf4j-jdk14-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder]


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.0.RELEASE)
==========================================

Application Name: [console]
  - Id: [com.gtcafe.cloud]
  - Version: [1.0.0]
  - Description: [Application to manage cloud resources]

Artifact Meta:
  - Build Type: [dev]
  - Build Id: [20190326-2203]
  - Commit Hash: [c03be7f]

Versioning API: http://192.168.2.10:8080/_releng/version

==========================================
```

第三個月:建立 QA 團隊

- Planing (短中長), Hiring
- 準備工作
  - 建立 Lab, 採購 PC
  - 網路規劃
  - 設備採購:手機、IPCam、UART
  - IT 資源:AWS Account / VPN
- Training
  - 介紹產品
  - 了解技術架構
  - 訓練環境建置: Servers / DB / App / Embedded / SQL Init Data ...
  - 標準化測試流程
  - 標準化 Report Process

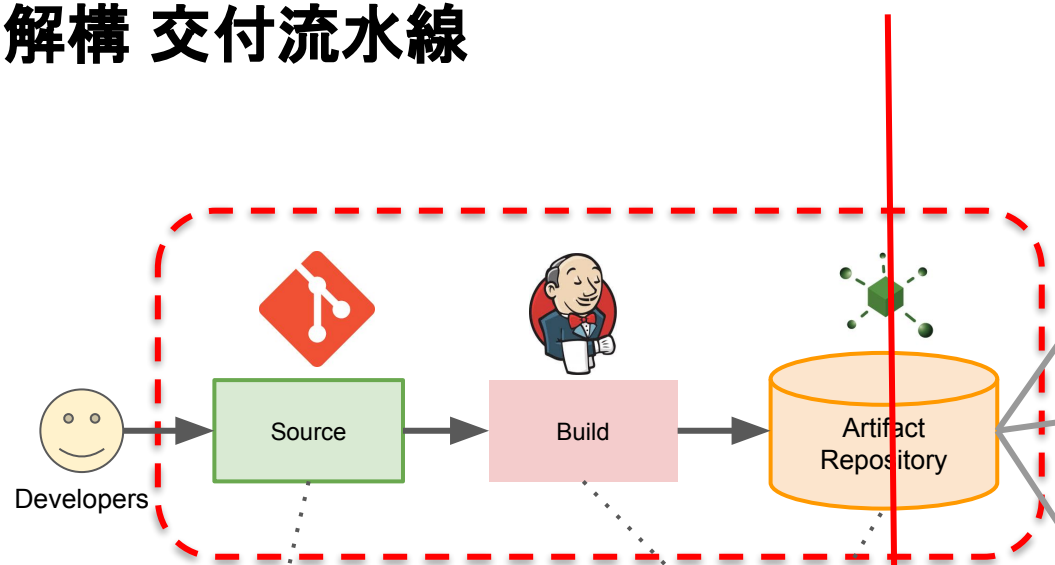
標準化 - Report Process

- Version
- Environment
- Log

```
## Precondition -- optional
Precondition for this issue.

## How to reprocedure the problem

1. install APP form AppStore
2. open app with 886929168168/12345678
3. goto video view, and click the active device
4. ..

## Expected result

The video should be transferred immediately.

## Acturl Result

The exception occurs.

## Log _optional_

put server, app or device log to here


## Environment

* Phone: iPhone5, iOS 7.0.2
* server version: server_2.3.0_b20130116-1200_r23765.tar.gz
```

Source: 如何有效的回報問題 (How to Report Problems Effectively)

解構 交付流水線



1. Semantic versioning and dependencies
2. Component naming
3. Artifact Meta
4. Branch strategy: trunk base

1. autobuild
2. allocate resources
3. rsync to remote site

00. Evn for **R**esearch and Proof of Concept

01. Env for **D**evelopment

10. Env for Functional **Test**

11. Env for Performance **Test**

12. Env for System, Regression **Test** .. etc

21. Env for Staging or Pre-Production

22. Env for **Production**

23. Env for **Sandbox** (API)

24. Env for DR Site

Continuous Integration

Continuous Delivery

這段過程做的事 ...

- 定義 Component Name / Semetric Versioning 的共識
- 建立 Build 流程標準 (Artifact Management 的基本)
- 每個 QA 都能夠**獨立建置**測試環境、**隨時部署**應用程式，不依賴於 RD
- 問題回報標準化，RD 只需要依據回報內容修改 Defect，不需要幫 QA 維護環境
- 硬體 / 韌體的交付工廠生產 IQC 標準化，讓 NPI 量產順利進行
- 新創事業要先做的事情
- 當時的時空背景這件事不難做

不一一過，用聊天的方式聊。

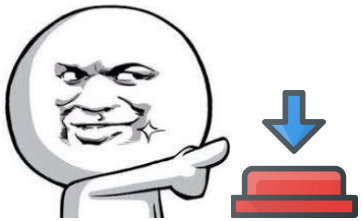
重點

一刀劃出 CI / CD 的界線

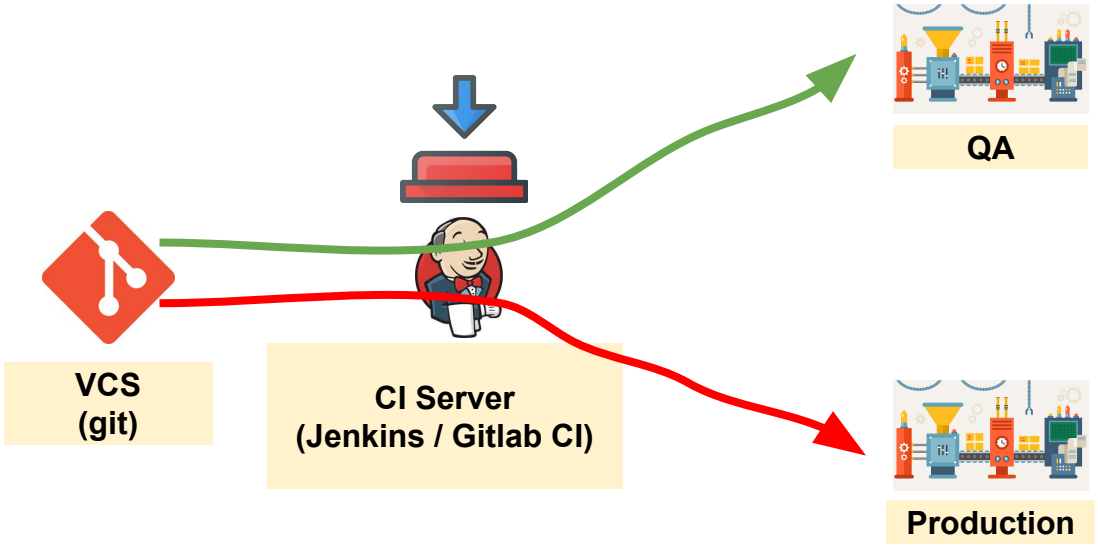
#解偶 開發流程 依賴性
#定義 清楚定義 溝通介面

第二件事　一個觀察

關於『一鍵部署』

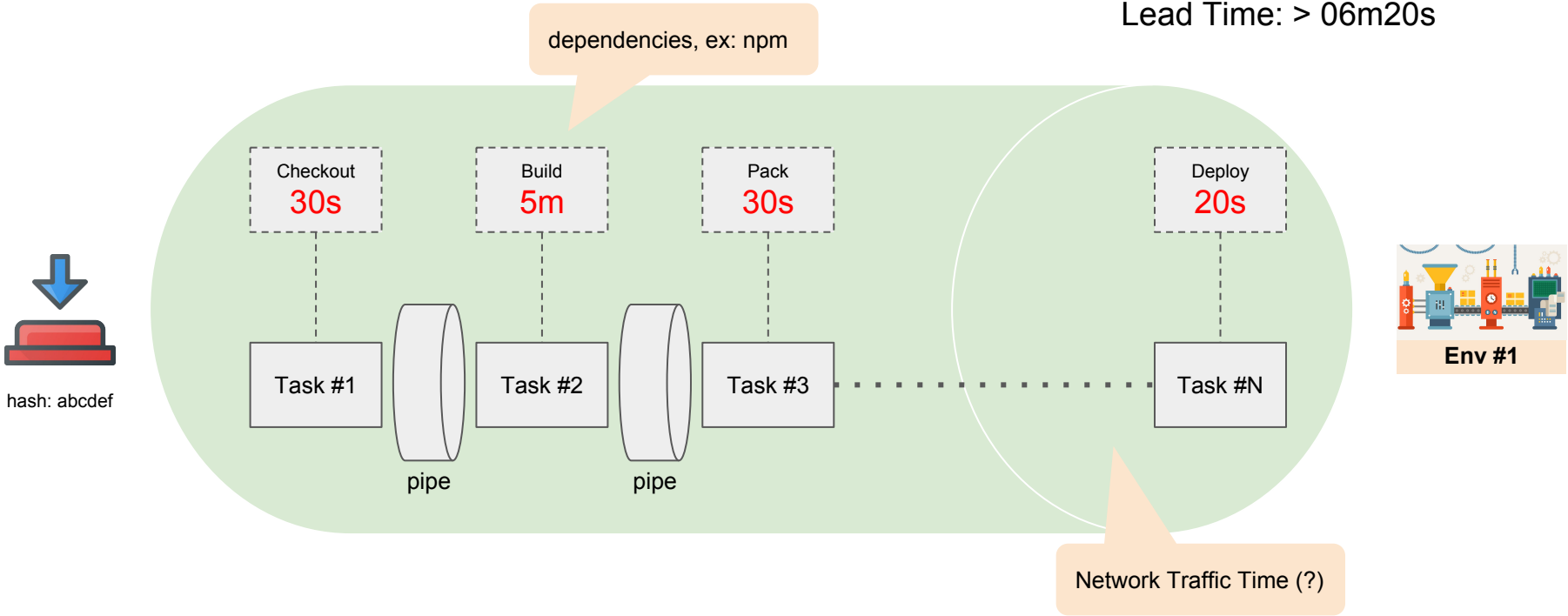


一鍵部署的『表象』: Web Application

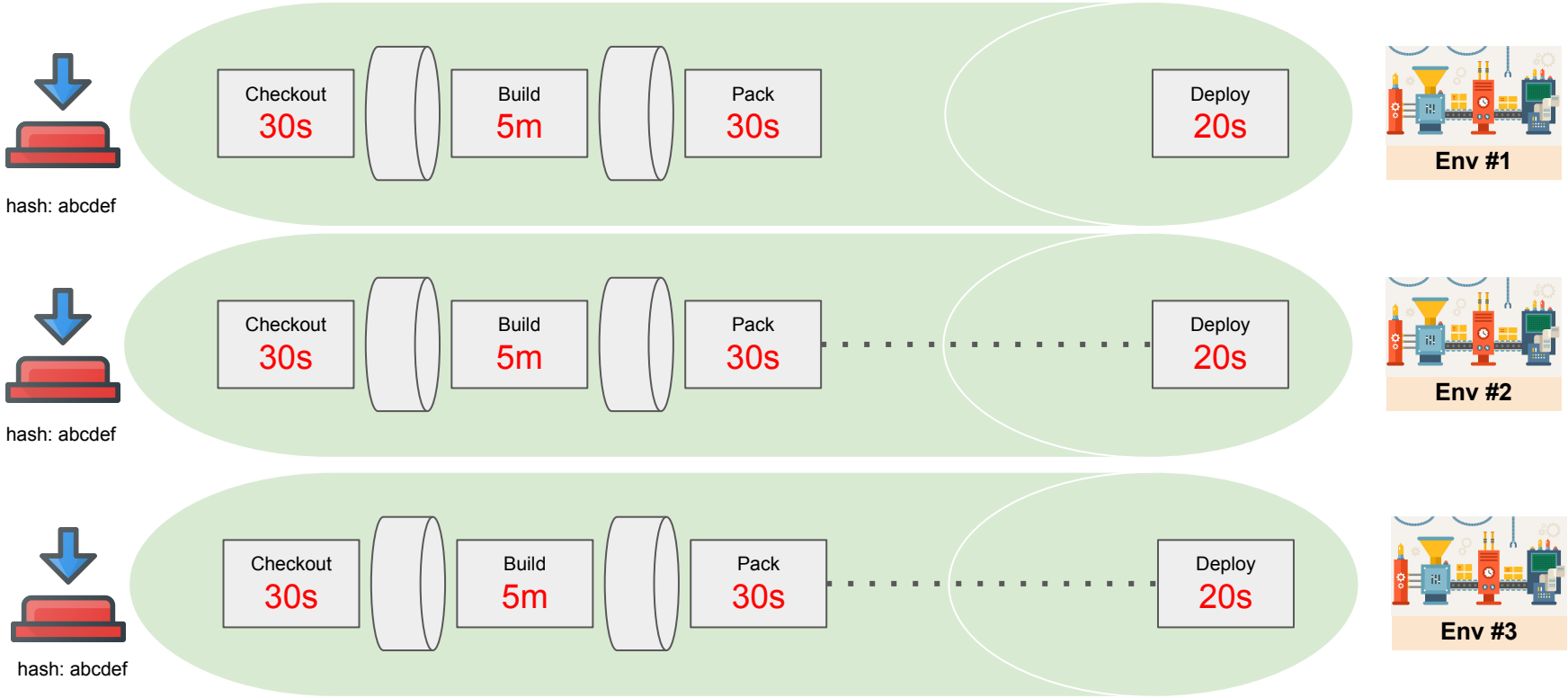


部署流水線的樣子

Lead Time: > 06m20s



新增一個環境 (測試或業務需求)



可能的問題 ...

- 每次都**重新 Build,** Image **沒有**重複使用
  - 重 Build 、打包浪費時間
- 如何確認 Build 的 Image 是同一個？
  - 同一個 source 的 hashcode，可以保證跑出來結果一樣？
  - Dependency 都沒變？
- 沒有把 **Config 獨立**出 Source Code
  - 無法重複使用 Image 部署
- Image 傳輸時間沒考慮(現在是雲世代)
- 沒有任何『版本』的資訊
  - 溝通成本

使用 Open Source 時會是怎樣

- 每次都**重新 Build,** Image **沒有**重複使用
  - 固定下載位址，有 md5 checksum
  - 節省 Build 的時間
- 如何確認 Build 的 Image 是同一個？
  - 看 checksum，Docker hashcode
- 沒有把 **Config 獨立**出 Source Code
  - Config 通常要自行注入 (DI)、或者有預設值
- Image 傳輸時間沒考慮（現在是雲世代)
  - Image 透過同步機制，平行傳輸
- 沒有任何『版本』的資訊：溝通成本
  - 大部分 OSS 都有版本，例如『**K8s 1.14** 上市啦～～～』
  - OSS 預設版本為 **latest**

觀察到的現象 (不是每家公司都有)

- Pipeline 的耦合問題
  - 動作都黏在一起，沒有彈性
  - 動作並沒有設計，輸入輸出不清楚
  - Job 的可重用性很低
- 為了自動化而自動化
- Config 沒有抽離、也沒有提供 Template

低耦合的例子

Install kubectl binary using curl

macOS | Linux | Windows

1. Download the latest release:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-rele
```

To download a specific version, replace the `$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)` portion of the command with the specific version.

For example, to download version v1.13.0 on macOS, type:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/darwin/amd64/kubectl
```

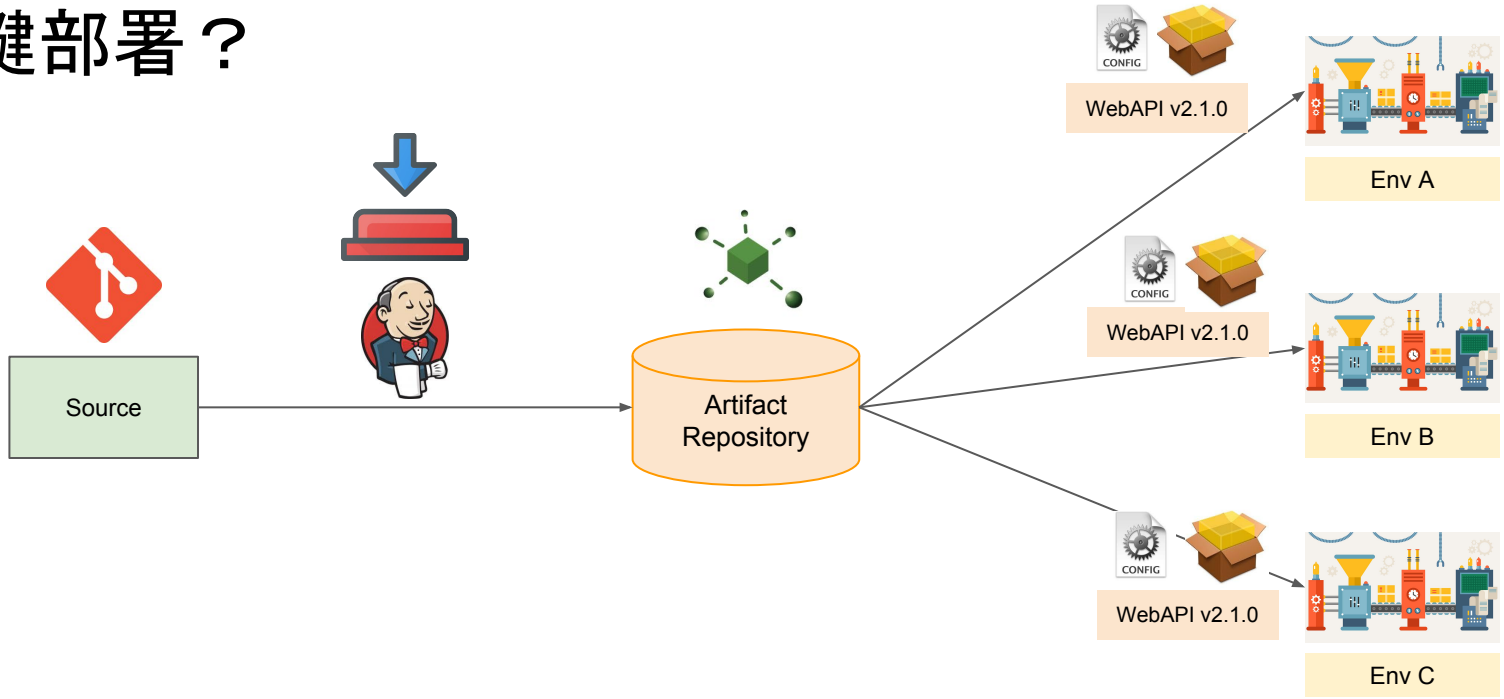
See: https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-binary-using-curl

Single Codebase, Multiple Delivery

```
# See: https://github.com/kubernetes/kops

curl -LO https://github.com/kubernetes/kops/releases/download/$(curl -s
https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name | cut -d '"' -f
4)/kops-linux-amd64
```

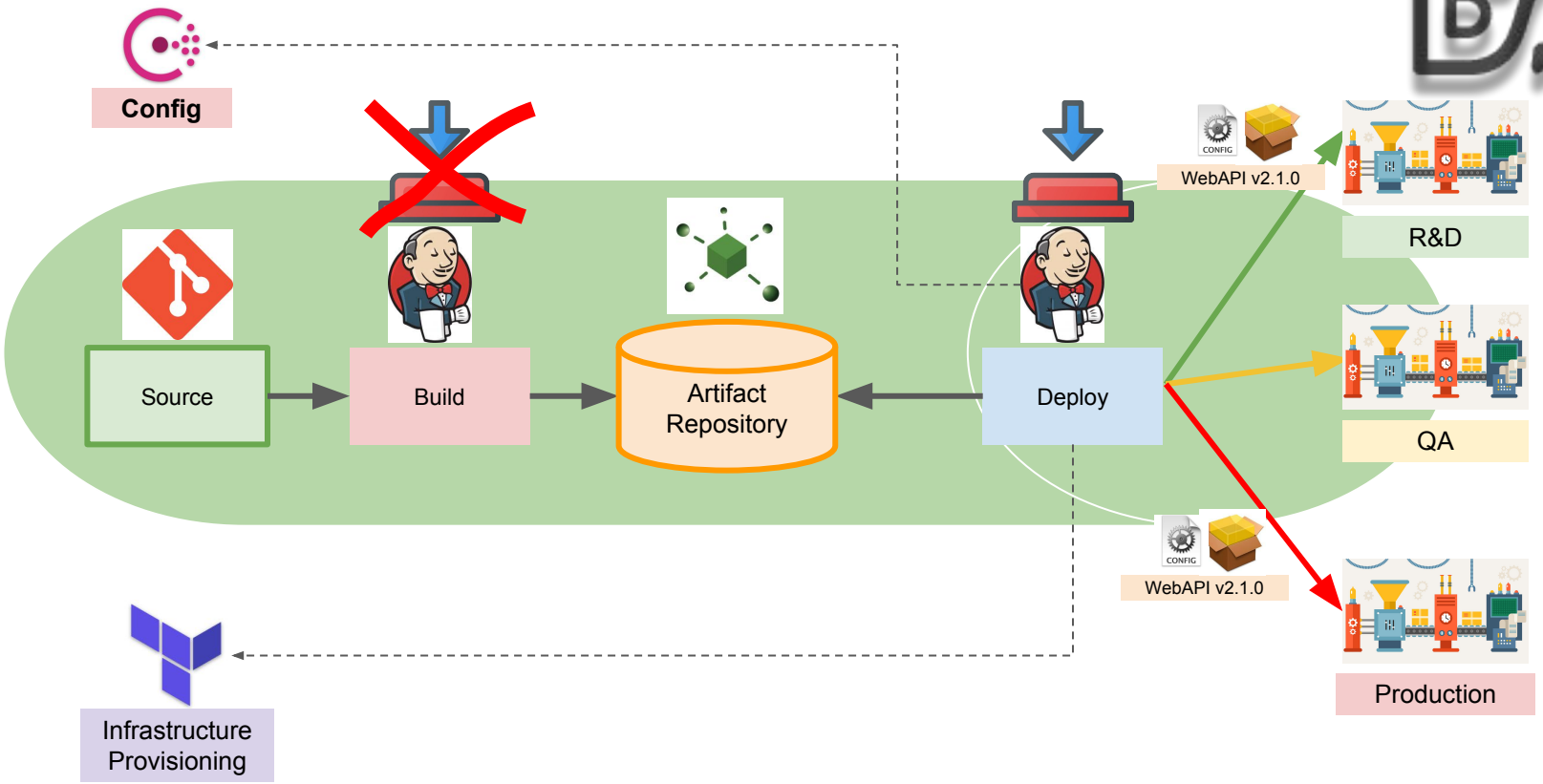
https://rickhw.github.io/2018/07/08/
DevOps/Artifacts-Management/

一鍵部署？



Continuous Integration

Continuous Delivery



Continuous Integration

Continuous Delivery

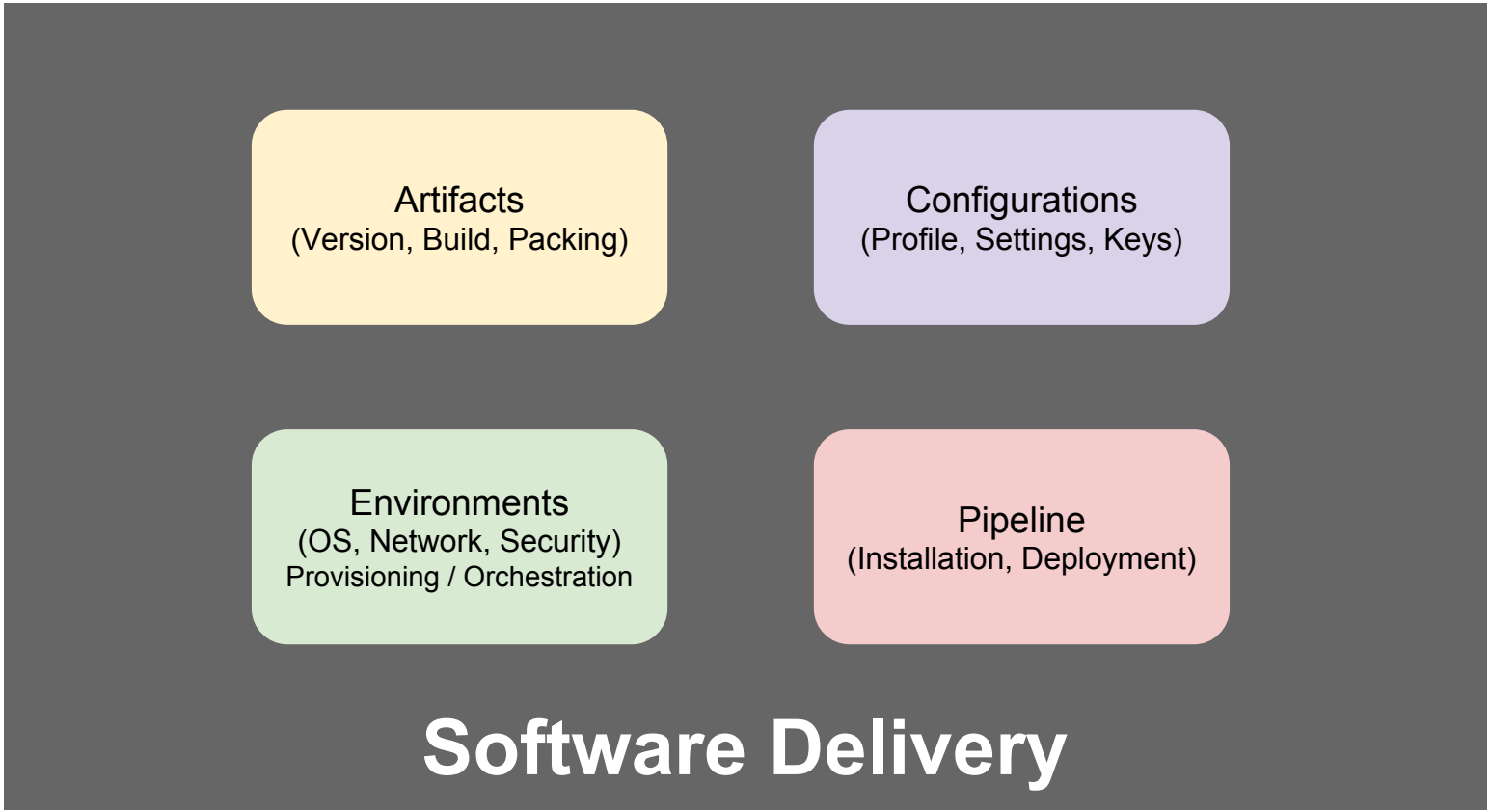
部署流水線的設計原則

- Job 之間應該是：高內聚、低耦合
  - 能夠彈性調度：手動、半自動、全自動、智慧 (AIPipeline)
- 跟工具無關
  - 工具應該是幫助效率化
  - 盡可能不要被綁死
  - Jenkins、GitlabCI、GitHub、Drone、Ansible …

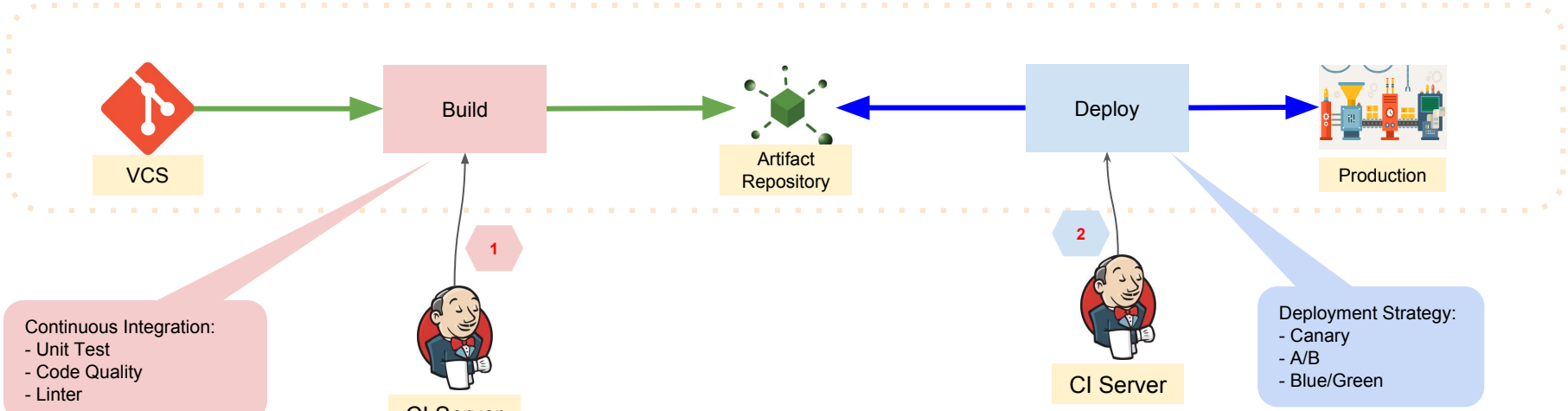
部署流水線的角色

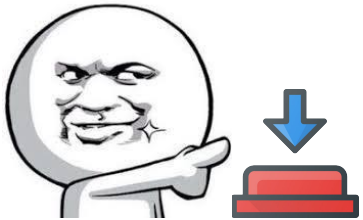
- 支柱角色：Source Control、Artifact Repository
- 目標環境：Test, Staging, Production → IaC
- 注入 (DI)：Config / Settings
- 流程控制角色：CI Server
  - 一鍵部署
  - 觸發器

軟體交付的**四大支柱**


Artifacts
(Version, Build, Packing)
Configurations
(Profile, Settings, Keys)
Environments
(OS, Network, Security)
Provisioning / Orchestration
Pipeline
(Installation, Deployment)
Software Delivery

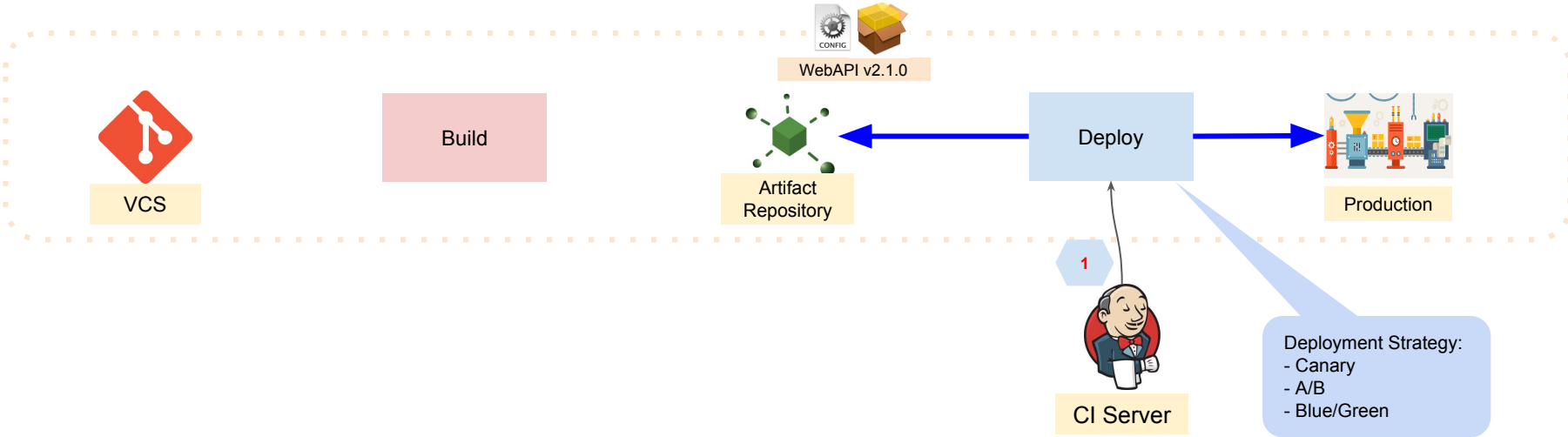
一鍵部署情境一：Sprint 交付 (Regular Release)

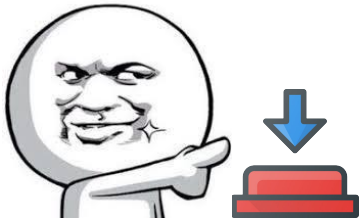




Release Manager, DevOps, SRE

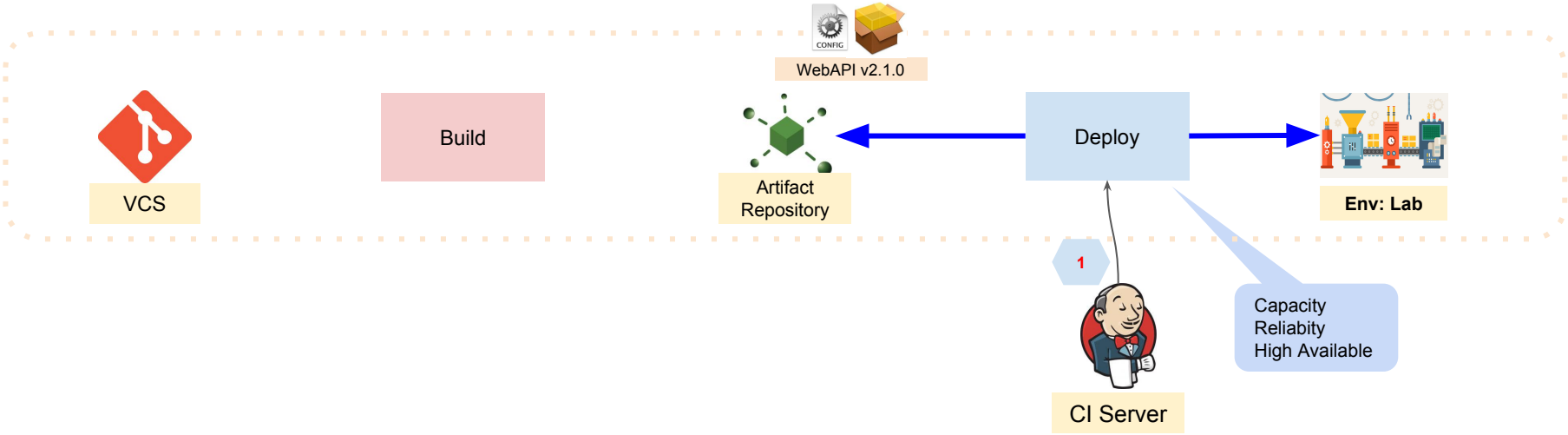
一鍵部署情境二：新業務，獨立環境

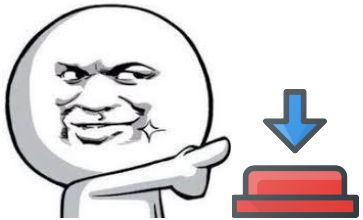




Release Manager, DevOps, SRE

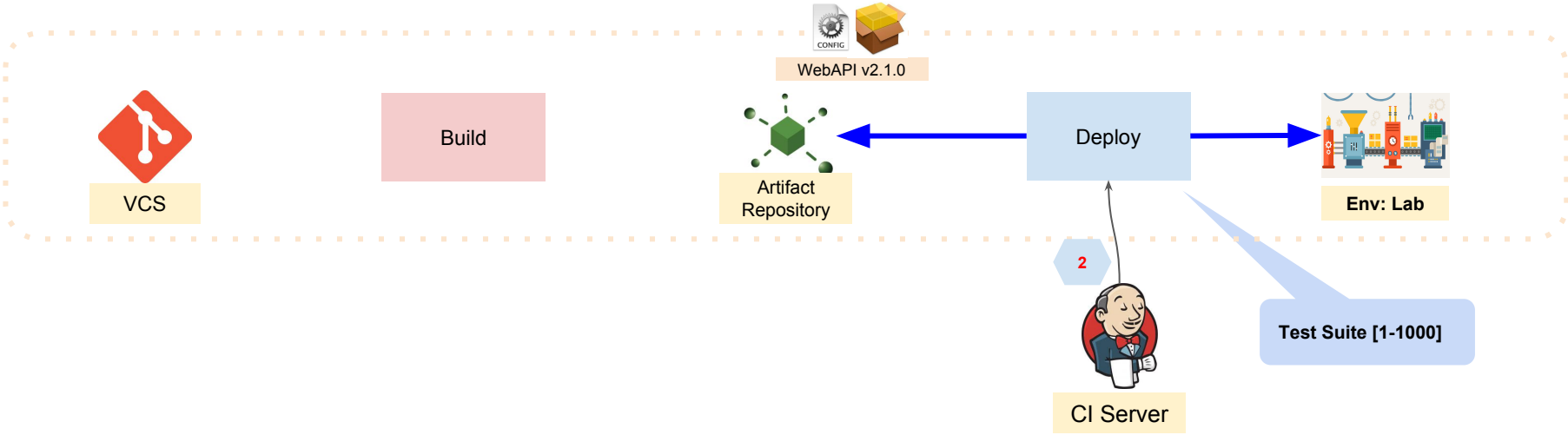
一鍵部署情境三：效能測試

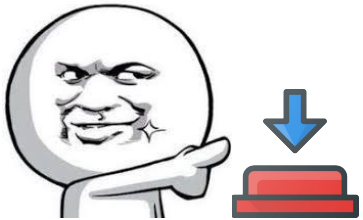




Performance Engineer

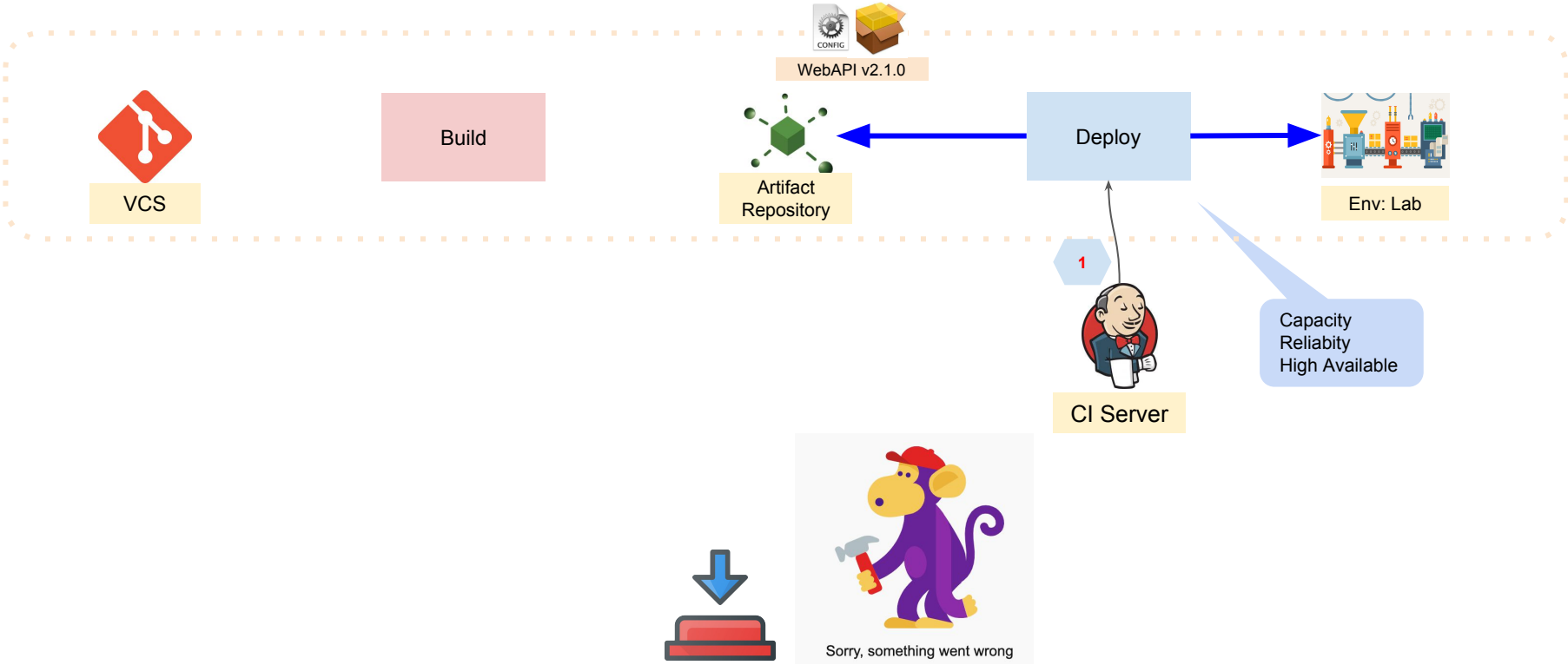
一鍵部署情境四 : 回歸測試





Test Engineer

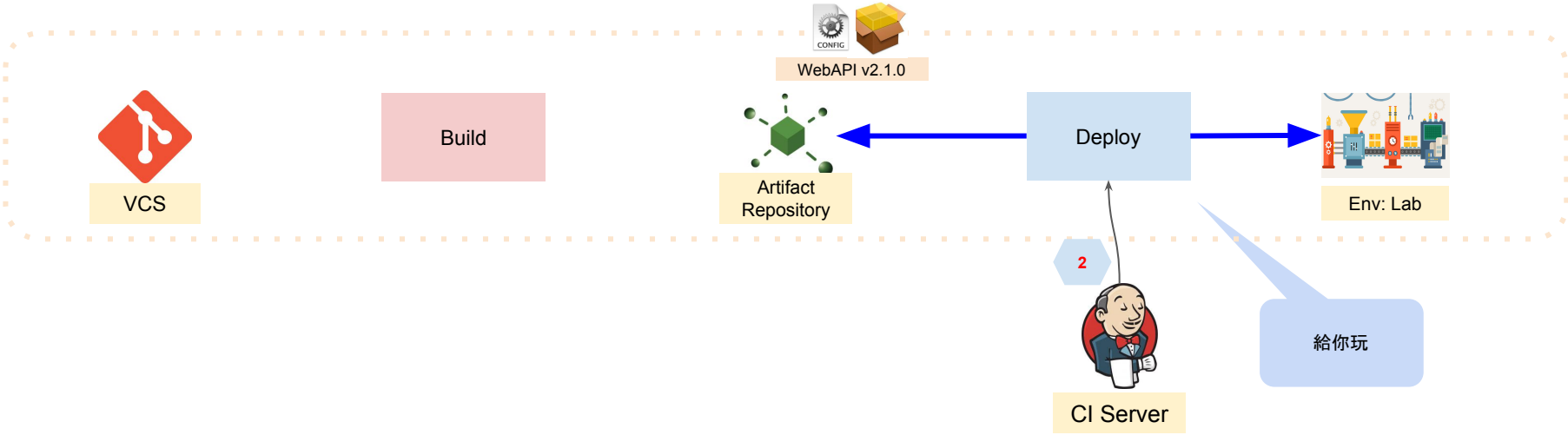
一鍵部署情境五：Chaos Monkey

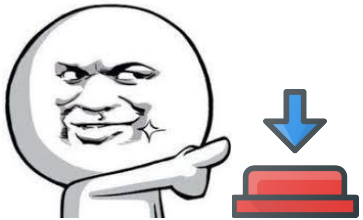


Rick Hwang

https://www.youtube.com/feed/library

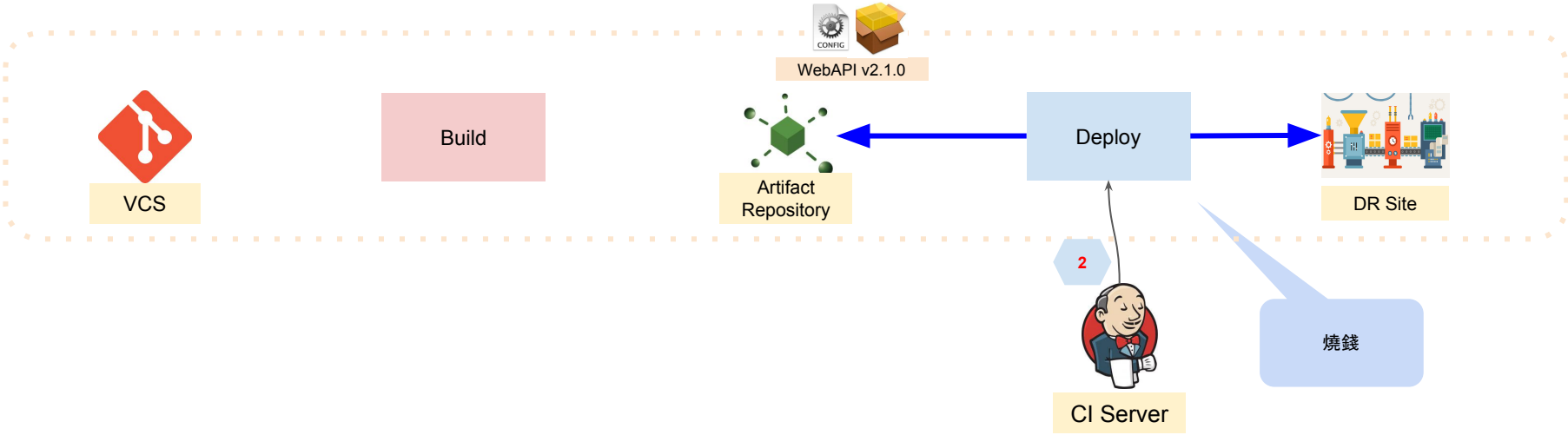
一鍵部署情境六:新人訓練

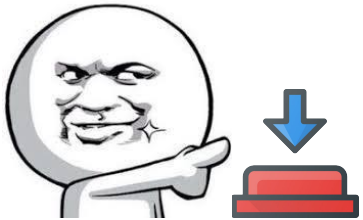




菜比巴

一鍵部署情境七 : 災難還原 (DR Site)





苦力 SRE

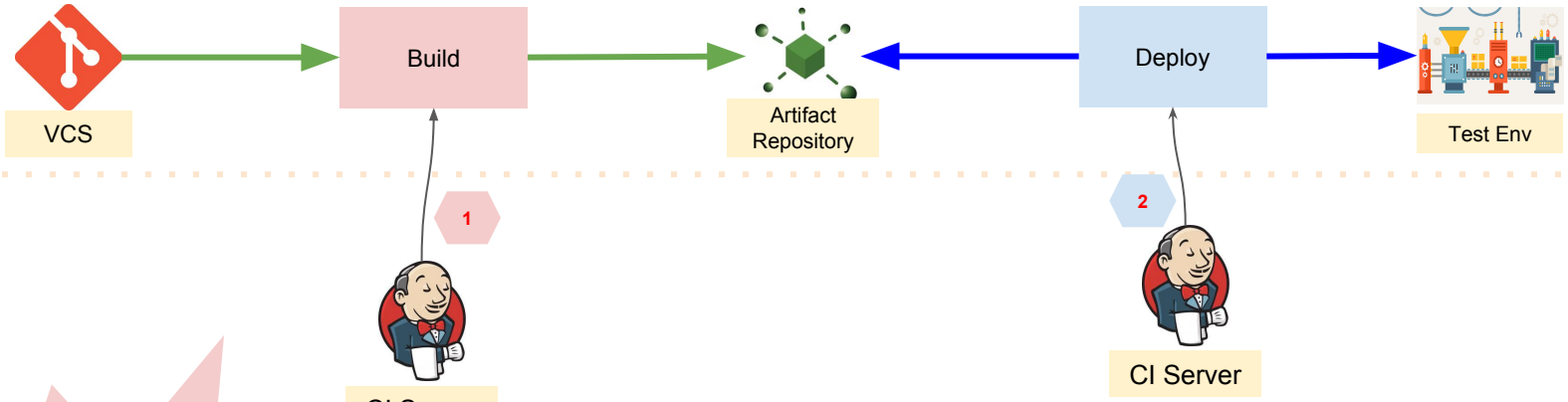
只有情境一
需要從頭跑到尾

(日常開發很常發生)

還有一個情境
也需要從頭跑到尾

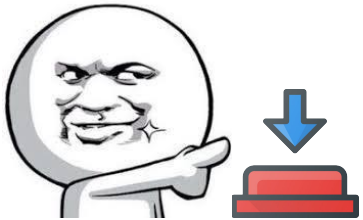
(日常開發很常發生)

每個 Sprint：驗證功能



請注意：

是**驗證者**自行驅動部署，
而不是被部署



驗證者：Tester / QA / 團隊

重點

一鍵部署，看誰部署

#角色的目的，決定一鍵部署的價值
#Artifact 可以依賴反轉 (IoC)，讓需要者自行決定如何部署

第三件事 真實世界

軟體交付
在大型組織裡的問題

考量 Artifact Types

最重要

App Images

NPM

maven

NuGet

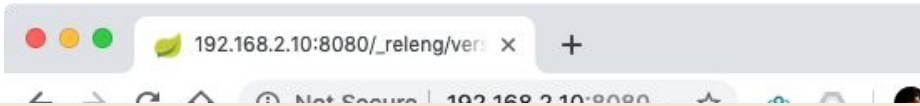
AWS AMI

docker

python
PIP

AWS

統一 Artifact Meta



需要效率化的時候

規格化標準介面是必要的過程。

https://rickhw.github.io/2015/02/11/SoftwareEngineering/Version-Control/

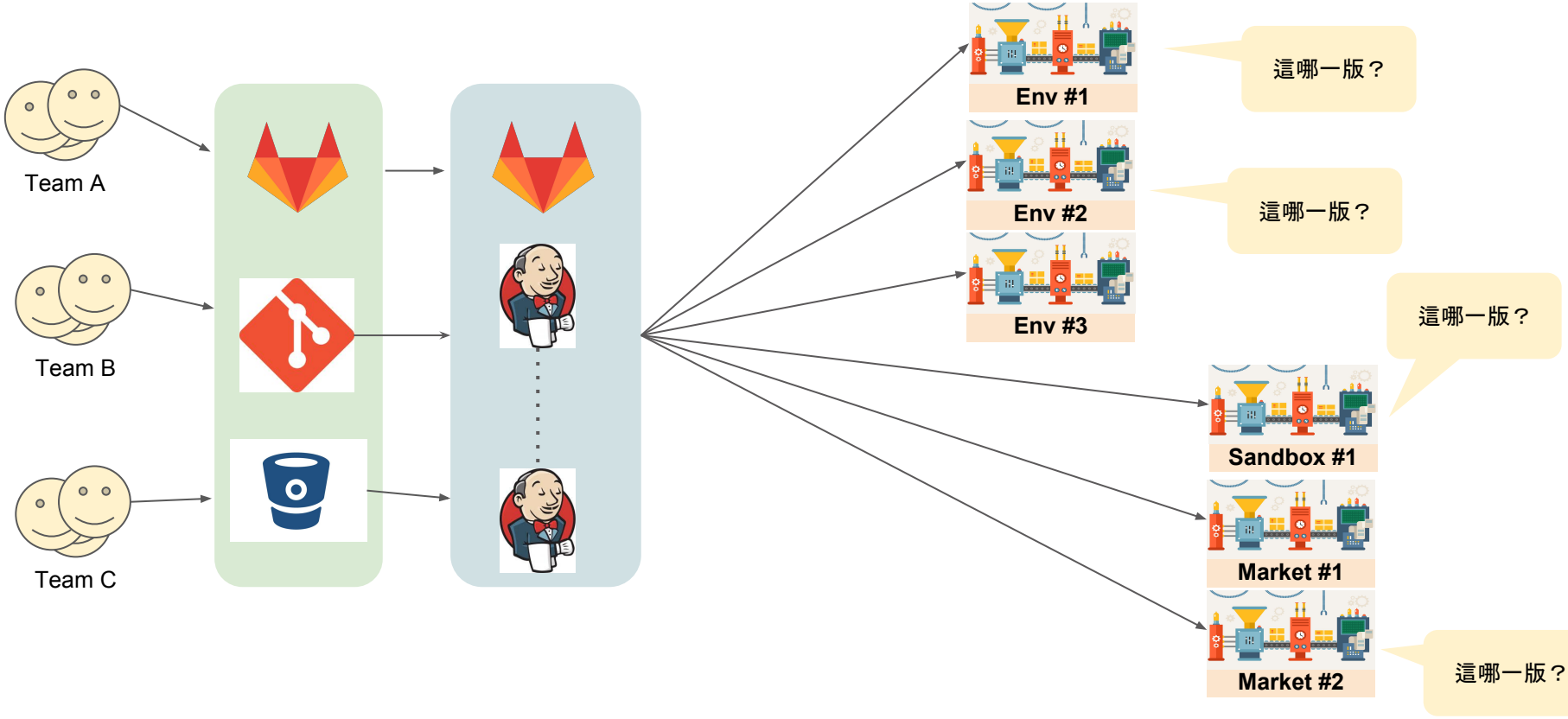
幾個『多』

- 多種 Source Code Providers: Gitlab、Bitbucket、GitHub、CodeCommit …
- 多種 CI Servers: Jenkins, GitlabCI, Bash, ...
- 多種 Artifact Types: Raw, npm, docker, maven, nuget, ...
- 多種部署環境: 測試、正式
- 多團隊協作: 溝通複雜

通常：

- 一定有 VCS (git)
- 一定有目標環境 (QA, Production)
- **沒有 Artifact Repository**

在大型組織裡，軟體的交付會是怎樣？



Continuous Integration

Continuous Delivery

- 鍵部署情境二：新業務，獨立環境
- 鍵部署情境三：效能測試
- 鍵部署情境四：回歸測試
- 鍵部署情境五：Chaos Monkey
- 鍵部署情境六：新人訓練
- 鍵部署情境七：災難還原 (DR Site)

這些事都會變得異常複雜 ....

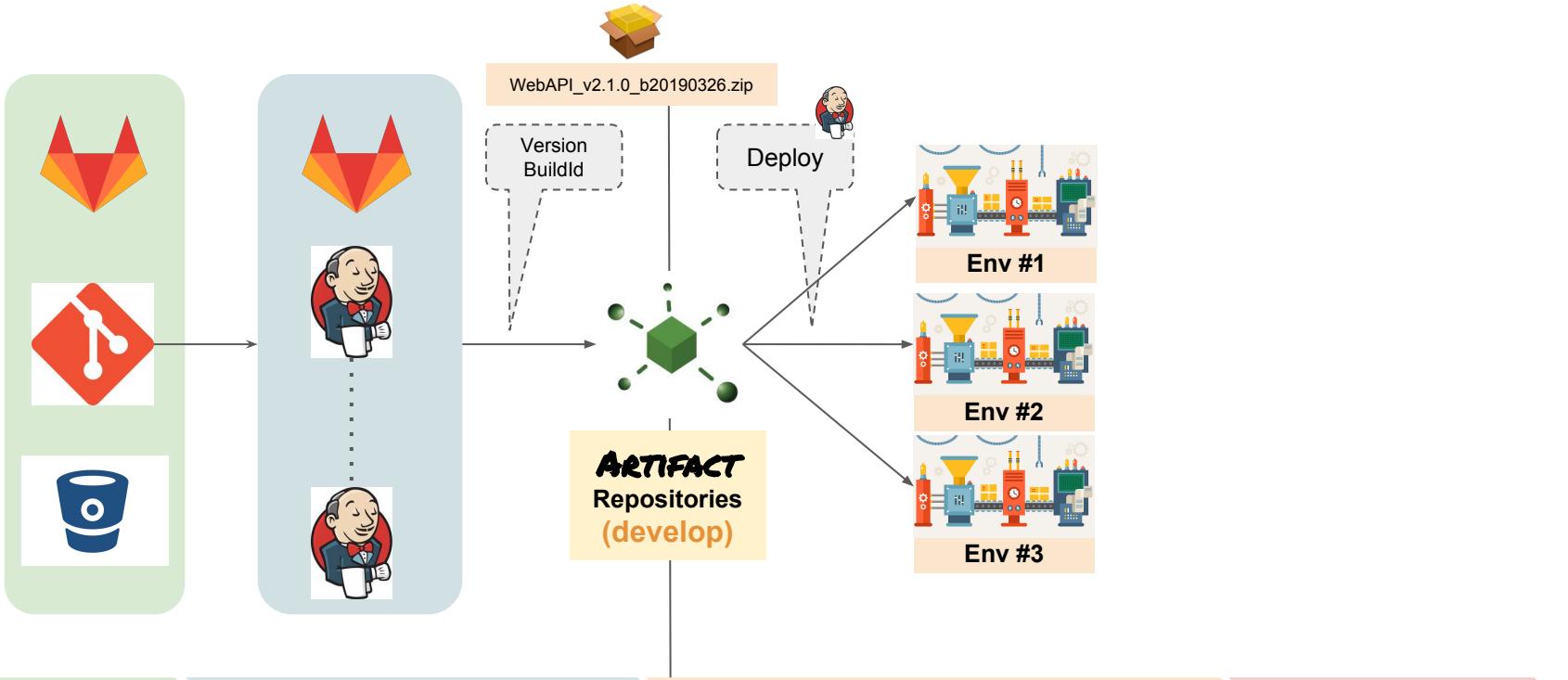
- 一鍵部署情境二：新業務，獨立環境
- 一鍵部署情境三：效能測試
- 一鍵部署情境四：回歸測試
- 一鍵部署情境五：Chaos Monkey
- 一鍵部署情境六：新人訓練
- 一鍵部署情境七：災難還原 (DR Site)

如果是這樣：

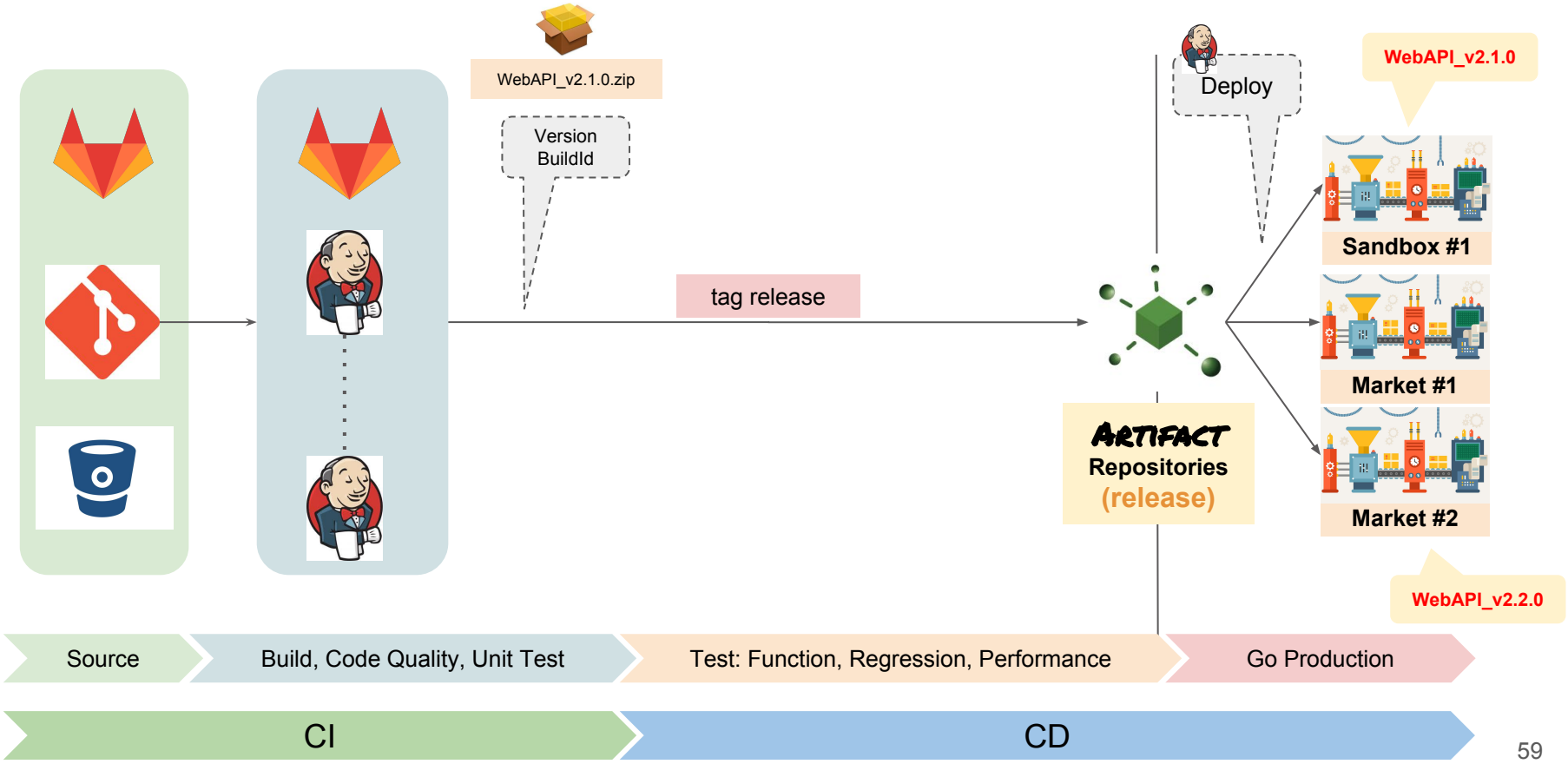
- 有 VCS (git)
- 有目標環境 (QA, Production)
- **有 Artifact Repository**

在大型組織裡，軟體的交付會是怎樣？

開發與測試 (Release 前)

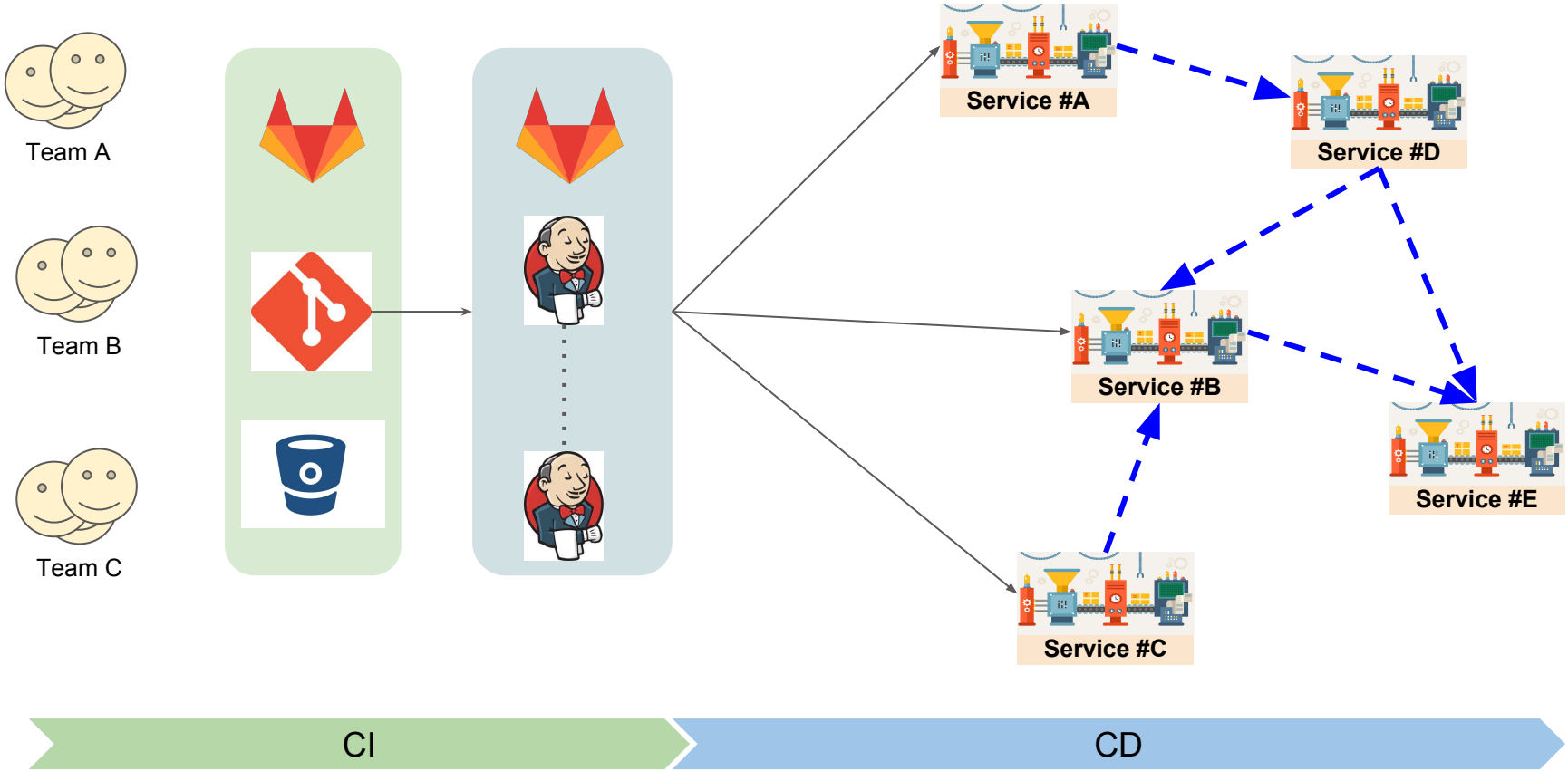


Tag Release

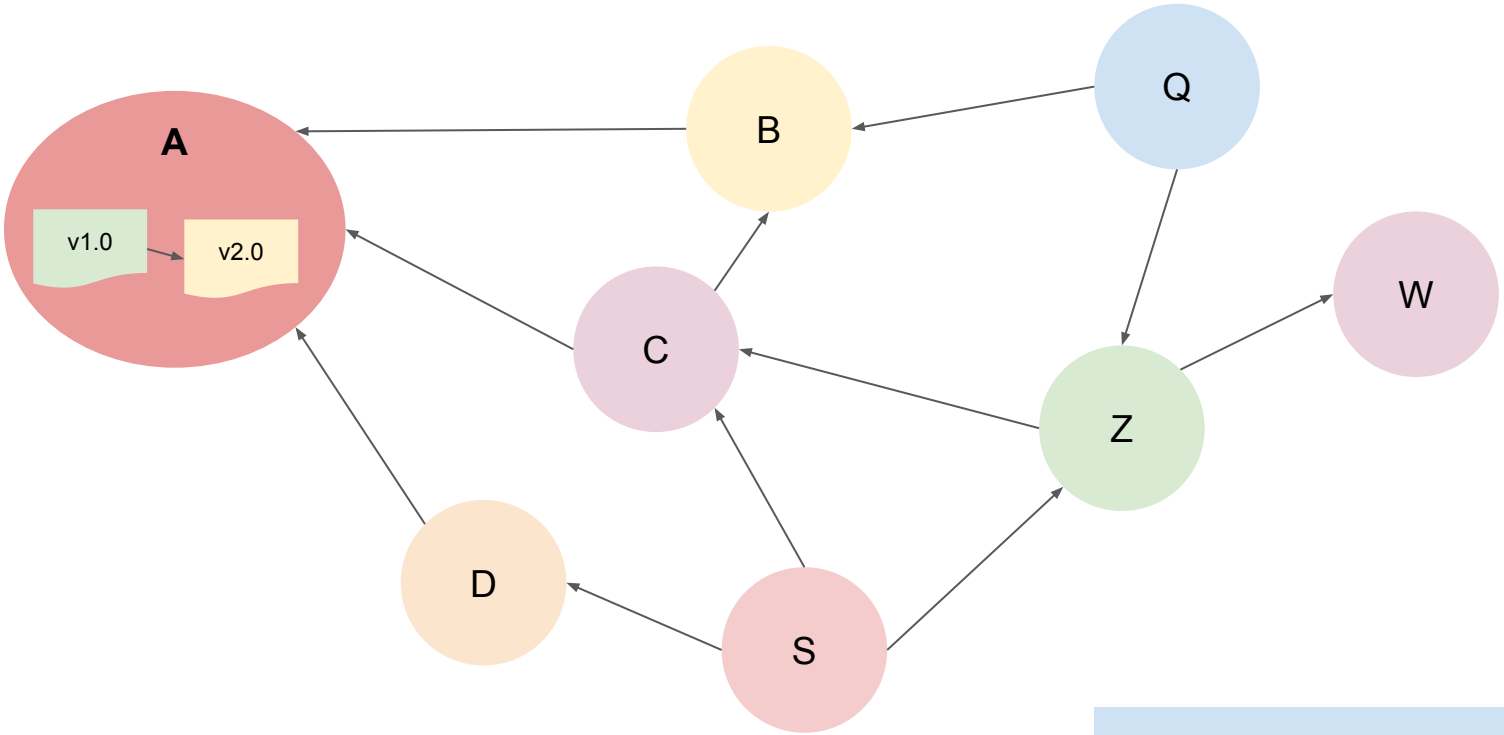

WebAPI_v2.1.0.zip
Version
BuildId
tag release
Deploy
WebAPI_v2.1.0
Sandbox #1
Market #1
Market #2
ARTIFACT
Repositories
(release)
WebAPI_v2.2.0
Source
Build, Code Quality, Unit Test
Test: Function, Regression, Performance
Go Production
CI
CD

部署流水線的複雜度問題

- 軟體有依賴性問題
- 部署流水線也有依賴性問題
  - 在大型組織裡 ....


Team A
Team B
Team C
Service #A
Service #D
Service #B
Service #E
Service #C
CI
CD

連鎖反應



Tips: 每個圈代表 Service or Bug

SRE CH13 - Emergency Response

Again: 統一 Artifact Meta

- Name: 名稱
- Description: 描述
- **Version: 版本, x.y.z, Sementic Versioning**
- BuildId: 時間, YYYYmmdd-HHMM
- Hash: VCS hashcode / Serial Number
- BuildType: dev | rel

Version 的目的:
- 關係管理
- 自己跟自己比

https://rickhw.github.io/2015/02/11/SoftwareEngineering/Version-Control/

產出物管理的實踐

- 定義版本與依賴管理
- 類型:
  - Raw / Docker / npm, NuGet, maven, pip … etc
- 使用 Artifact Management
  - 窮人版: Apache HTTP Index Directory
  - 人模人樣版: **Nexus OSS** (支援很多)
  - 有錢人: JForg Cloud 版

重點

標準化是必要的過程

軟體交付的濫觴
談產出物管理
Artifact Management

Rick Hwang
2019/03/28

DevOps Taiwan Meetup #22

濫觴 ㄌㄢˋ ㄕㄤ

làn shāng

1. 水流發源的地方。因其水量非常淺小，而僅能浮起一個酒杯，故稱為「濫觴」。

《荀子．子道》：「昔者江出於崏山，其始出也，其源可以濫觴。」

《文選．郭璞．江賦》：「惟岷山之導江，初發源乎濫觴。」

2. 比喻事物的開端、起源。

南朝梁．鍾嶸〈詩品序〉：「雖詩體未全，然略是五言之濫觴也。」

似 發軔、先導、先河

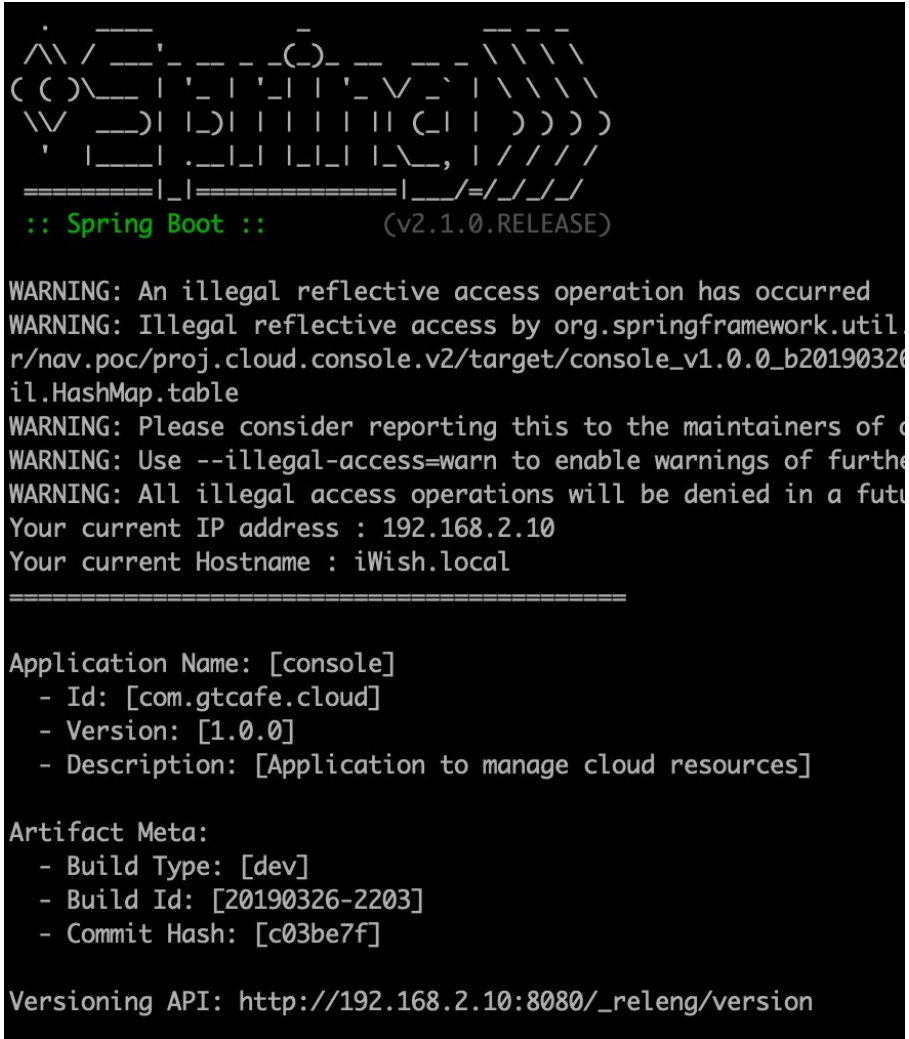
https://www.moedict.tw/%E6%BF%AB%E8%A7%B4

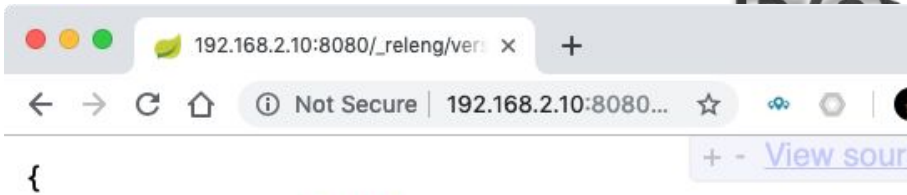
一些思考

- 個人：做 Side Project 的時候，一開始要做哪一些事情？
  - 如何交付給別人使用你的軟體？
  - 拿到你的軟體，要透過什麼方式驗證？
- 加入一家新創團隊，如何確認軟體交付是到位的？
- 加入一家有規模的企業，可能會遇到的交付問題？
- 如果，你自己 Startup 一家公司，你是 CTO，哪些基本功要先做好？

再想想這些問題

- 你的公司是新創草創期（１～３年）？
- 你的公司是新創成長期（３～５年）？
- 你的公司業務有多市場(北美、歐洲、日本)？
- 你的工作有多個環境部署需求？



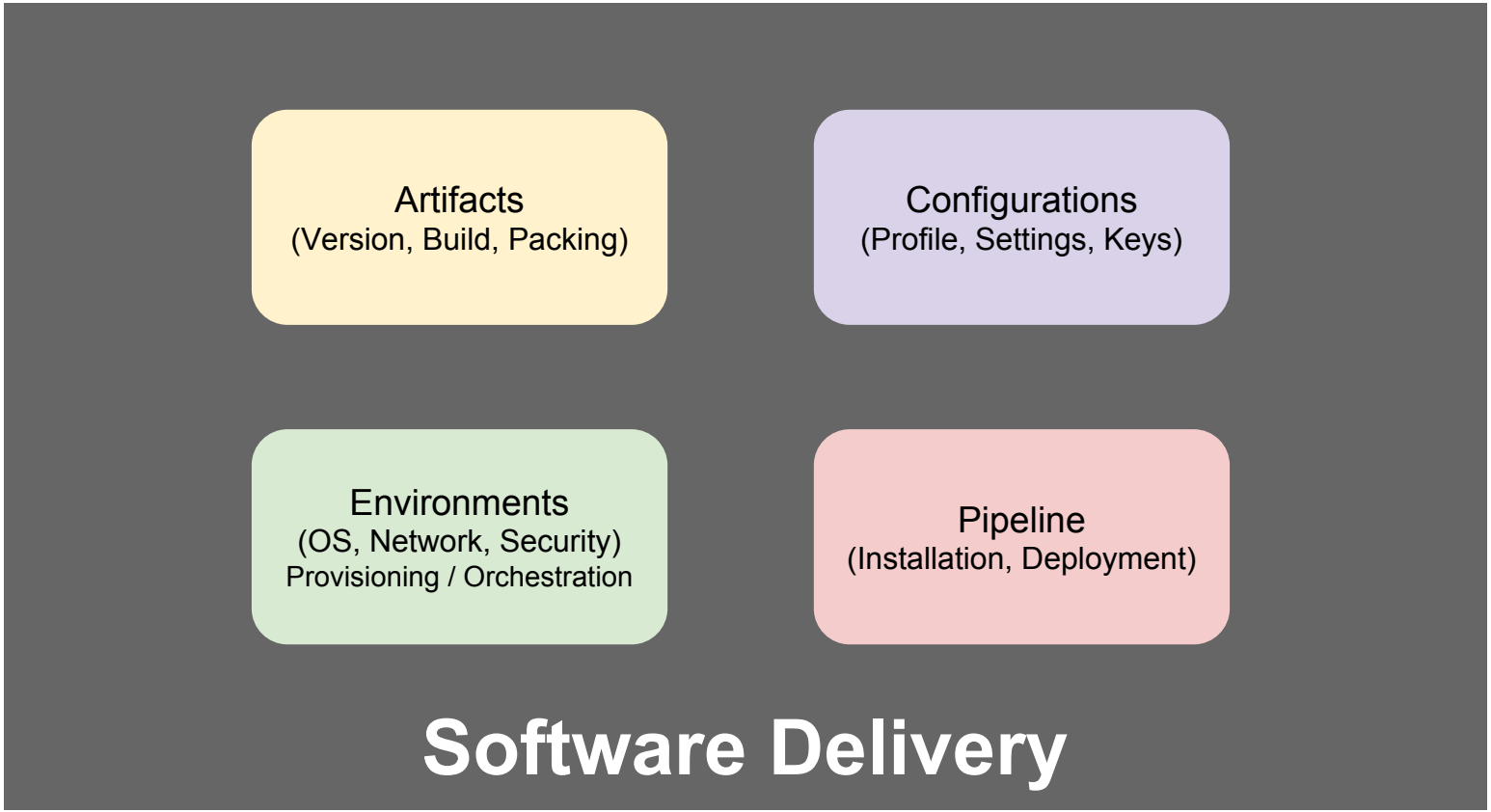



```
{
    buildType: "dev",
    name: "console",
    buildId: "20190326-2203",
    version: "1.0.0",
    hash: "c03be7f"
}
```

軟體交付包涵：

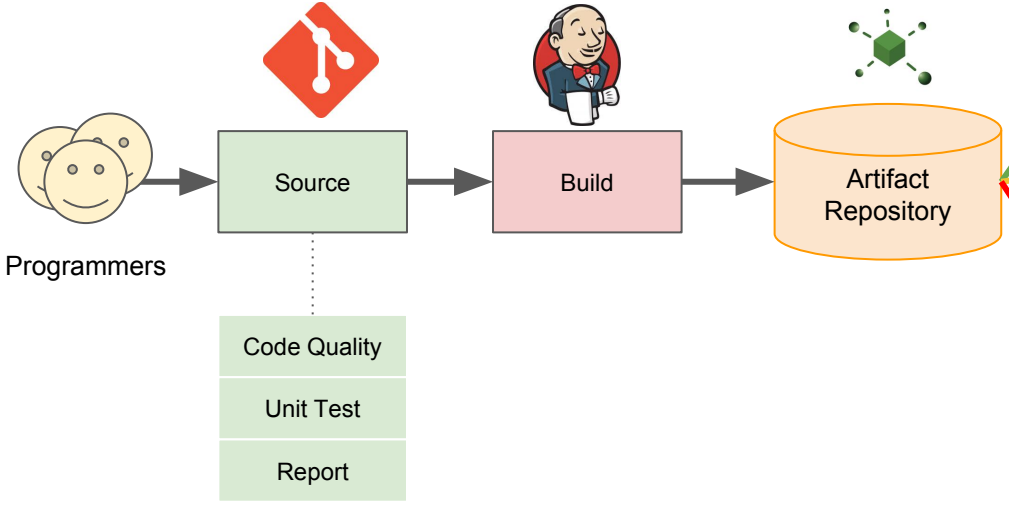
1. 應用程式本身
2. 設定 (config, static) 與 配置 (settings, dynamic)
3. 環境
4. 交付方式

軟體交付的**四大支柱**


Artifacts
(Version, Build, Packing)
Configurations
(Profile, Settings, Keys)
Environments
(OS, Network, Security)
Provisioning / Orchestration
Pipeline
(Installation, Deployment)
Software Delivery

持續交付流水線

(1) 研究與開發:技術選擇與決策
(2) 探索與驗證:功能與非功能
(3) 部署策略與風險控管



(1)
(2)
(3)

00. Evn for **R**esearch and Proof of Concept
01. Env for **D**evelopment

10. Env for Functional **Test**
11. Env for Performance **Test**
12. Env for System, Regression **Test** .. etc

21. Env for Staging or Pre-Production
22. Env for **Production**
23. Env for **Sandbox** (API)
24. Env for DR Site

Continuous Integration

Continuous Delivery

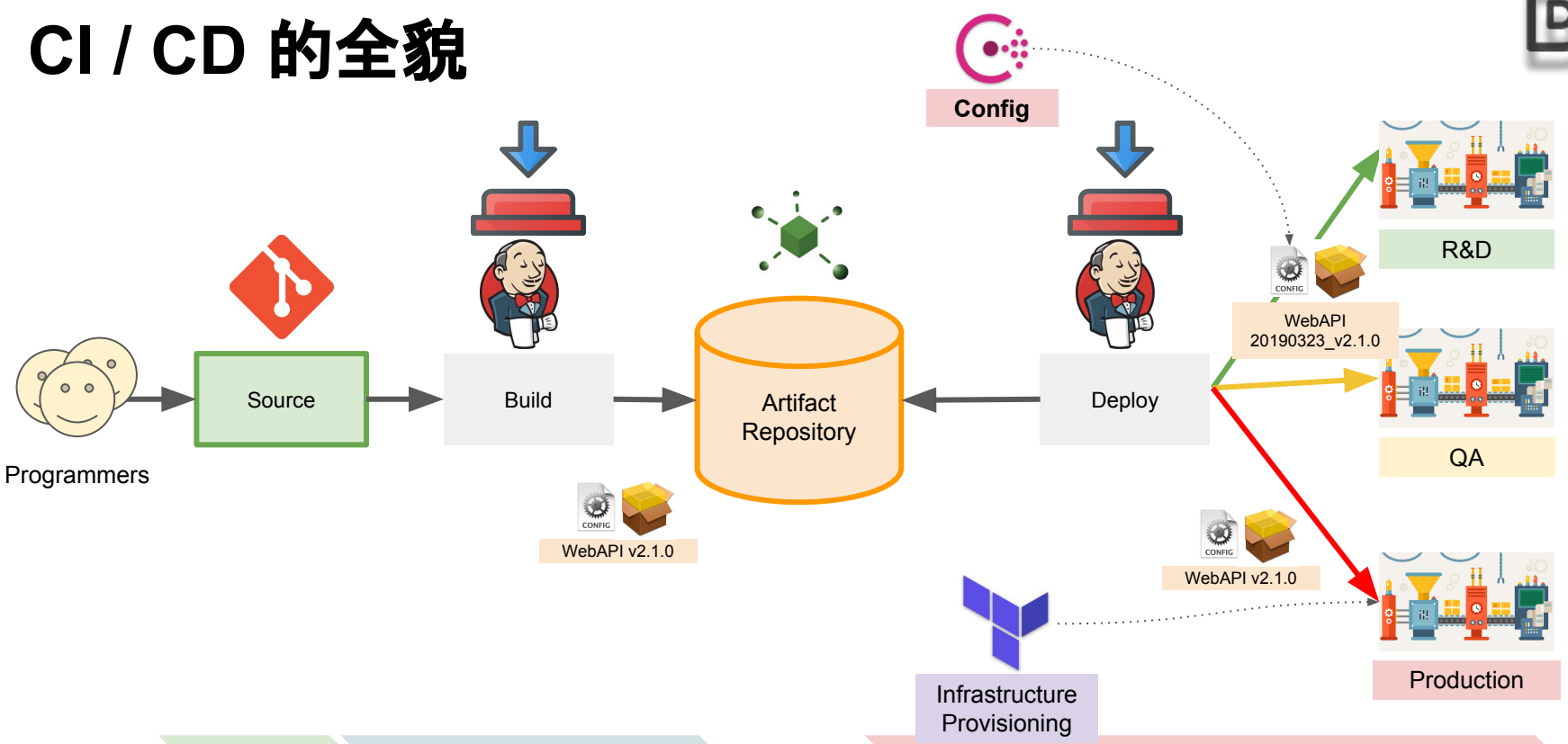
任何人都可以

部署任意版本

到特定環境 (包含建置)

Source: Continuous Delivery - Opening

CI / CD 的全貌


Config
Programmers
Source
Build
Artifact Repository
Deploy
WebAPI 20190323_v2.1.0
R&D
QA
Production
WebAPI v2.1.0
WebAPI v2.1.0
Infrastructure Provisioning
Source
Build, Pack
Code Quality, Unit Test
Continuous Integration
Go Production
Test: Function, Regression, Performance
Continuous Delivery

延伸題目

- Artifact Management: 如何處理跨區域 private docker 的問題？
- 分支策略：Trank Base or Branch Base? Why?
- Infra as Code 也是 Code，所以他會有 Artifact？他有 Config？他的分支策略？
  - Infra as Code 可以利用繼承方式，描述出各個環境的關係
- Build LATEST 的用途？
- Config Management: 如何管理多環境的配置？如何設計繼承架構的配置？

腦力激盪與思考

- 版本號碼應該用 x.y.z 還是像 Chrome 58 那種單一碼？
- 你看過四碼的版本嗎？像是 1.2.3.4, 他們代表什麼意思？
- 每個 Sprint 都要跳版號？誰來做？
- 新創事業需要有 Artifact Repository?
- 你家的 IaC 有 Artifact Repository?
- 分支策略跟 Artifact Management 的關係？

相關文章

- Artifacts Management
- 導入 CI/CD 的第一步
- 怎樣的 CI/CD 才夠 Quality?
- 如何有效的回報問題 (How to Report Problems Effectively)
- Version Control
- 協同合作系統建制與導入 - 以 Redmine 為例
- Resource Provisioning and DevOps
- Continuous Delivery - Opening
- Software Development Lifecycle


תודה
Dankie
Gracias
Спасибо
شكراً
Merci
Takk
Köszönjük
Terima kasih
Grazie
Dziękujemy
Dėkojame
Ďakujeme
Vielen Dank
Paldies
Kiitos
Täname teid
谢谢
Thank You
Tak
感謝您
Obrigado
Teşekkür Ederiz
감사합니다
Σας ευχαριστούμε
ขอบคุณ
Bedankt
Děkujeme vám
ありがとうございます
Tack